Ein Modell zur Untersuchung der Formalisierbarkeit des funktionsorientierten Tests

Vom Fachbereich Informatik der Technischen Universität Kaiserslautern

zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)

genehmigte

Dissertation

von

Dipl.-Math. Benedikte Elbel

Berichterstatter:	Prof. Dr. Peter Liggesmeyer Prof. Dr. Klaus Pohl
Dekan:	Prof. Dr. Reinhard Gotzhein
Datum der wissenschaftlichen Aussprache:	20.06.2007

D 386

Kurzdarstellung

Software stellt ein komplexes Werkzeug dar, das durch seine umfassenden Möglichkeiten die moderne Gesellschaft entscheidend geprägt hat. Daraus ergibt sich eine Abhängigkeit von Funktion und Fehlfunktion der Software, die eine an den funktionalen Anforderungen orientierte Entwicklung und Qualitätssicherung der Software notwendig macht. Die vorliegende Arbeit schafft durch Formalisierung und Systematisierung der Verfahren im funktionsorientierten Test eine fundierte Basis für eine Hinwendung zu den funktionsorientierten Techniken in Softwareentwicklung und –qualitätssicherung.

Hierzu wird in der Arbeit zunächst ein formales Modell für das Vorgehen im dynamischen Test beschrieben, das sich an der Begriffsbildung der Literatur und dem Verständnis der Praxis orientiert. Das Modell beruht auf wenigen zentralen Annahmen, eignet sich für formale Untersuchungen und Nachweise und ist wegen seiner sehr allgemein gehaltenen Definitionen breit anwendbar und einfach erweiterbar. Auf dieser Basis werden Vorgehen und Verfahren zum funktionsorientierten Test analysiert. Zunächst wird dazu das Vorgehen im funktionsorientierten Test im Rahmen des Modells dargestellt. Darauf aufbauend werden zentrale Verfahren des funktionsorientierten Tests analysiert, die zum Gegenstand die systematische Prüfung der Umsetzung von weitgehend informal beschriebenen Anforderungen in einem Softwareprodukt haben. Betrachtet werden Verfahren der funktionalen Partitionierung, der funktionalen Äquivalenzklassenanalyse und Grenzwertbildung, Verfahren zur Prüfung von kausalen Zusammenhängen zwischen Ursachen und Wirkungen, Verfahren zur Prüfung von graphisch spezifizierter Funktionalität in Syntaxdiagrammen, Aktivitätsdiagrammen, Sequenz- und Kollaborationsdiagrammen und Petrinetzen, Verfahren zum Test zustandsbasierter Systeme sowie Ansätze einer funktionalen Dekomposition.

Die Analyse und Diskussion der bekannten Verfahren im formalisierten Rahmenwerk führt zu zahlreichen Ergebnissen und Verfahrensergänzungen. So zeigt sich, dass in den klassischen, informalen Beschreibungen häufig Unklarheiten bestehen. Diese werden hier adressiert und durch Angabe von Kriterien präzisiert, Optimierungsmöglichkeiten werden aufgezeigt. Darüber hinaus wird an der einheitlichen formalen Darstellung der in der Literatur meist separat betrachteten Verfahren deutlich, welche Vergleichbarkeit zwischen den Verfahren besteht, welche Verfahrenskombinationen sinnvoll sind und wie durch ein kombiniert funktions- und strukturorientiertes Vorgehen eine hohe Aussagekraft in der analytischen Qualitätssicherung erreicht werden kann. Bei der Formulierung der Verfahren im Rahmen des Modells wird herausgearbeitet, wo zur Verfahrensdurchführung die kreative Leistung des Testers notwendig ist und welche Anteile formalisiert und damit automatisiert unterstützt werden können.

Diese Betrachtungen bilden die Grundlage für die Skizzierung einer integrierten Entwicklungsumgebung, in der ein funktionsorientiertes Vorgehen in Entwicklung und Qualitätssicherung umgesetzt wird: Hier helfen funktionsorientierte Beschreibungsformen bei der Angabe der Spezifikation, ihrer Verfeinerung und ihrer Vervollständigung, sie unterstützen die Entwicklung durch Modellbildung, sie liefern die Basis für eine funktionsorientierte Testdatenselektion mit Adäquatheitsprüfung, sie können bei geeigneter Interpretierbarkeit über den Datenbereichen zur automatisierten Testfallgenerierung genutzt werden und unterstützen als suboptimale Testorakel eine automatisierte Auswertung des dynamischen Tests. Diese Skizze zeigt die praktische Umsetzbarkeit der vorwiegend theoretischen Ergebnisse dieser Arbeit und setzt einen Impuls für ein verstärktes Aufgreifen funktionsorientierter Techniken in Wissenschaft und Praxis.

Inhalt

1 Einleitung

1.1 Formalisierung und Intuition im funktionsorientierten Test von Software

Man is a tool-making animal.

Benjamin Franklin (1706, † 1790)*

Man mag geteilter Meinung sein, ob dieses Zitat von Benjamin Franklin der menschlichen Natur in vollem Umfang gerecht wird. Lange vor der ersten elektronischen Rechenmaschine entstanden, vermag es jedoch die Faszination von Software und ihren Siegeszug zu erklären, der in den letzten Jahrzehnten zu beobachten war: Software stellt ein mächtiges Werkzeug bereit, das logische Entscheidungen, Berechnungen und Steuerungen übernimmt und so den Menschen unterstützt, entlastet und ihm geistige Freiräume verschafft. Die Mächtigkeit von Software hat jedoch auch eine Kehrseite: Fehlfunktionen können von komplexer Natur sein und katastrophale reale und ökonomische Auswirkungen haben.

Im Zuge der rapiden Entwicklung der Chip- und Speichertechnologie der letzen Jahrzehnte haben sich das nutzbare Datenvolumen und die Auswertungsgeschwindigkeit von Software drastisch erhöht. Mit den technischen Möglichkeiten sind auch die Anforderungen an Software und softwareunterstützte Systeme gestiegen. Wurde Software ursprünglich von wenigen Spezialisten eingesetzt, erweiterte sich nach und nach ihr Einsatzbereich. Heute ist jeder mit Software konfrontiert und auf ihr sicheres und zuverlässiges Funktionieren angewiesen – sei es indirekt als Mitglied einer hochtechnologisierten Industriegesellschaft, oder direkt bei der Nutzung von softwarebasierten Systemen wie Telekommunikationseinrichtungen, medizintechnischen Geräten oder am eigenen Computer. Mit den Einsatzbereichen von Software hat auch ihre gesellschaftliche und wirtschaftliche Bedeutung zugenommen: So wurden die Kosten, die im Jahr 1998 durch Qualitätsmängel von Software in den USA entstanden sind, in der Zeitschrift „Business Week“ mit 85 Milliarden Dollar beziffert [GroEA99]. Das sichere und zuverlässige Funktionieren von Software und softwaregesteuerten Systemen ist damit längst zu einem gesellschaftlichen und wirtschaftlichen Erfolgsfaktor geworden [OstEA96]. Insbesondere für den Wirtschaftsstandort Deutschland wird der Qualität der entwickelten Software eine besondere Bedeutung im globalen Wettbewerb beigemessen [Lig05].

Software kann in ihrer typischen Ausprägung als diskretes mathematisches Objekt angesehen werden, dessen Verhalten im Allgemeinen keinerlei Stetigkeitsbedingungen genügt und daher schwer vorhersagbar ist [Voa99, Ham02]. So stellt die Qualitätssicherung von Softwareprodukten vor dem Hintergrund ihrer vielfältigen Einsatz- und Nutzungsmöglichkeiten ein komplexes Problem dar. Seit einigen Jahrzehnten wird an diesem Thema geforscht, ohne dass ein vollends befriedigender und allgemein akzeptierter Ansatz gefunden werden konnte. Zwar konnten in vielen Bereichen der Softwaretechnik große Fortschritte erzielt werden, doch die Grundprobleme der Erstellung und Qualitätssicherung von Software als ein Werkzeug, das seinen funktionalen und qualitativen Anforderungen gerecht wird, ist bestehen geblieben.

Dieses Problem ist in der Natur der Software selbst begründet. Software wird im Teil 1 der [ISO IEC 2382 93] definiert als Gesamtheit oder Teil der Programme, Prozeduren, Regeln und der zugehörigen Dokumentation eines Informationsverarbeitungssystems, und wird als intellektuelles Produkt unabhängig von seinem Speichermedium gesehen. Damit ist das Produkt „Software“ lediglich eine in großen Teilen formalisierte und damit maschinell interpretierbare Beschreibung der eigenen Funktionali-

tät und ihrer Realisierung im Rechner. Während in den Anfängen der Softwareentwicklung die Realisierung im Rechner einen wichtigen Stellenwert bei der Programmierung einnahm, ist dieser Aspekt nun von zunehmend untergeordneter Bedeutung. Moderne Programmiersprachen und -paradigmen sowie die Entkopplung von Applikationslogik und technischer Realisierung durch Schichtenmodelle der Softwarearchitektur ermöglichen eine Konzentration des Programmierers auf die geforderte Funktionalität. Hierin zeigt sich die größte Herausforderung und das eigentliche Kernproblem der Softwareentwicklung: Die vollständige formale Beschreibung einer zunächst informal vorliegenden Idee der Funktionsweise der Software. Die Gestaltung dieses Formalisierungsprozesses und seine Adaption an die kognitiven Strukturen des Menschen ist eins der fruchtbarsten Arbeitsgebiete der Softwaretechnik. Dennoch beinhaltet der Formalisierungsprozess immer eine intellektuelle Leistung des Menschen, die fehlerbehaftet sein kann und eine stringente Qualitätssicherung des Entwicklungsergebnisses unumgänglich macht.

Mit dem Ziel der Qualitätssicherung sind in den letzten Jahrzehnten zahlreiche konstruktive und analytische Techniken entwickelt worden, die auf eine Fehlervermeidung bei der Erstellung von Software sowie auf ihre umfangreiche Prüfung abzielen. Zur systematischen Prüfung der funktionalen Eigenschaften von Software wurden funktionsorientierte Testverfahren entwickelt, die sich bei der Auswahl der Testfälle und bei der Bewertung ihrer Vollständigkeit an den funktionalen Anforderungen orientieren [Mye79, How80, Lig90, Lig02]. Durch die Orientierung an den funktionalen Anforderungen kommt den funktionsorientierten Testverfahren eine zentrale Rolle bei der Qualitätssicherung zu: So kann nur durch einen systematischen und vollständigen funktionsorientierten Test geprüft werden, ob alle Anforderungen in der Software realisiert wurden.

Die Orientierung an den funktionalen Anforderungen, die die Notwendigkeit und Relevanz des funktionsorientierten Tests begründet, beinhaltet aber gleichzeitig auch seine inhärente Problematik: So ist die Grundlage eine meist informal gegebene Beschreibung der Anforderungen, aus der konkrete Testfälle abgeleitet und in maschineninterpretierbarer Form angegeben werden, oder auf deren Basis Aussagen über die Vollständigkeit bestehender Testfälle getroffen werden. Damit ist auch hier, wie bei der Entwicklung der Software selbst, die Transformation von informal vorliegenden Informationen in ein formales Ergebnis zu leisten. Vor diesem Hintergrund wird deutlich, dass eine Systematisierung, Formalisierung und Automatisierung des funktionsorientierten Testens nicht vollständig erreicht werden kann. Dies mag eine Begründung dafür sein, dass der funktionsorientierte Test eher selten im Interesse der Wissenschaft und Forschung steht, obwohl er für die Praxis von zentraler Bedeutung ist.

Ziel der vorliegenden Arbeit ist es, durch eine Untersuchung der Möglichkeiten zur Formalisierung der bekannten Verfahren des funktionsorientierten Tests eine systematische Basis für ihre Anwendung und Weiterentwicklung zu schaffen. Dabei wird eine weitgehende Beschreibung der Verfahren mit den Mitteln der Prädikatenlogik angestrebt. Gleichzeitig werden Verfahrensaspekte herausgearbeitet, die nicht im Rahmen dieser oder einer erweiterten Terminologie formal beschrieben werden können. Sie zeigen auf, wo und in welchem Rahmen die Kreativität und Intuition des Testers bei der Durchführung des Verfahrens notwendig ist. Vor diesem Hintergrund wird deutlich, dass die Bezeichnung des Testens von Software als eine Kunst, wie sie G. J. Myers mit dem Titel seines Buches „The Art of Software Testing“ [Mye79] geprägt hat, in besonderem Maße auf das funktionsorientierte Testen zutrifft.

1.2 Aufbau der Arbeit

Eine Einführung zum funktionsorientierten Test und eine Motivation der in dieser Arbeit durchgeführten Untersuchung und Formalisierung des Themas wird in Kapitel 2 gegeben. Hier wird zunächst in

einer thematischen Übersicht das funktionsorientierte Testen in den Kontext der Softwarequalitätssicherung einbettet; zentrale Begriffe werden eingeführt. Die anschließende Betrachtung der Entwicklung der Verfahren der Softwarequalitätssicherung und des dynamischen Testens führen auf den Stand der Technik und den Handlungsbedarf zur Formalisierung, Unterstützung und Weiterentwicklung der funktionsorientierten Testverfahren.

Eine formale Grundlage für die Untersuchung des funktionsorientierten Tests wird in Kapitel 3 gegeben. Hier werden zunächst Annahmen getroffen, die für die formale Beschreibung des Vorgehens im dynamischen Test hilfreich und in der einen oder anderen Form unumgänglich sind. Dabei wird eine möglichst allgemeine, an den Belangen der Praxis orientierte Darstellung angestrebt. Unter Vorgabe der getroffenen Annahmen werden anschließend die Begriffe des dynamischen Tests mit Hilfe von Techniken der allgemeinen Mengenlehre und der Prädikatenlogik beschrieben. So wird ein Modell als Basis für die Untersuchung der funktionsorientierten Testverfahren bereitgestellt. Die Tragfähigkeit des Models wird in einer abschließenden Bewertung vor dem Hintergrund verschiedener Ausprägungen des dynamischen Testens in der Praxis beleuchtet.

Die Untersuchung der funktionsorientierten Testverfahren ist Gegenstand des Kapitels 4. Hier werden zunächst die Charakteristika des funktionsorientierten Tests in der Terminologie des in Kapitel 3 bereitgestellten Modells ausgedrückt. Anschließend werden die bekannten und in der Praxis etablierten Verfahren zum funktionsorientierten Test untersucht. Angestrebt wird eine möglichst vollständige Formulierung der Verfahren vor dem Hintergrund der in Kapitel 3 gegebenen Definitionen. Bei der Beschreibung der Verfahren wird deutlich, dass der Bezug zur meist informal gegebenen Darstellung der funktionalen Anforderungen eine vollständige formale Beschreibung der Verfahren unmöglich macht. Es wird analysiert, welche Aspekte der funktionalen Anforderungen relevant und damit aus der Spezifikation herauszuarbeiten sind. Hier wird greifbar, welche geistige Leitung bei der Durchführung des Verfahrens durch den Tester zu erbringen ist und in welchem Rahmen seine Kreativität und Intuition gefordert werden. Unter Benennung der relevanten Aspekte der funktionalen Anforderungen wird eine formale Beschreibung der Verfahren möglich, die in prädikatenlogischer Form gegeben wird. Auf dieser Basis erfolgt eine Diskussion zu möglichen Ergänzungen, Vervollständigungen und Kombinationsmöglichkeiten der Verfahren.

Da eine vollständige Betrachtung aller bekannten funktionsorientierten Testverfahren im Rahmen dieser Arbeit nicht möglich ist, wird die Auswahl der Verfahren vor dem Hintergrund der praktischen Relevanz getroffen. So wird in Kapitel 4 das Augenmerk insbesondere auf Testverfahren zu den in der Praxis verbreiteten Spezifikationsformen gerichtet. Die Arbeit bezieht sich daher auf Verfahren der funktionalen Partitionierung beziehungsweise der Äquivalenzklassenanalyse, auf Verfahren zur Prüfung von Ursachen und ihren Wirkungen, auf Verfahren zur Prüfung von graphisch spezifizierter Funktionalität und auf Verfahren zum Test von Anforderungen an zustandsbehaftete Systeme.

Verfahren, die auf einer formalen Beschreibung von Anforderungen aufsetzen, stehen hingegen nicht im Fokus dieser Arbeit. So werden beispielsweise die Verfahren der Testdatengenerierung und Testauswertung auf Basis algebraischer Spezifikationen von abstrakten Datentypen nicht betrachtet [GanMcMHam81, BerGauMar91, DauGauMar93, DooFra94, GauJam99], auch wenn diese Verfahren auf der Spezifikation der Software operieren. Der wichtigste Unterschied dieser Techniken zu den hier betrachteten Verfahren des funktionsorientierten Tests besteht in ihrem Beitrag zur Qualitätssicherung im Formalisierungsprozess der Softwareentwicklung: So wird bei der Erstellung einer formalen Spezifikation die Transformation der ursprünglichen, informal vorliegenden Anforderungen in eine vollständig definierte, formale Beschreibung der Funktionalität der Software in die Phase der Anforderungsde-

finition vorgezogen. Dieser Ansatz ist zunächst vielversprechend und hilft, viele Fehlerquellen in der weiteren Entwicklung zu umgehen. Hierbei ist jedoch zu berücksichtigen, dass der Prozess der Formalisierung lediglich antizipiert und gleichzeitig auch deutlich komprimiert wird. Dies führt zu extrem hohen Anforderungen an die intellektuellen Fähigkeiten der damit betrauten Personen und macht eine umfassende Qualitätssicherung für den Prozess der Anforderungsdefinition unumgänglich. Die Techniken der Testfallgenerierung auf Basis einer formalen Spezifikation können hierzu einen guten Beitrag leisten. Sie orientieren sich aber nicht an den ursprünglichen, informal formulierten Anforderungen an die Software, sondern vielmehr an der Struktur der Spezifikation und damit des zu prüfenden Artefaktes. Damit unterschieden sie sich in ihrer Intention grundlegend von den hier betrachteten funktionsorientierten Testverfahren, die eine Qualitätssicherung der Transformation der informalen Ideen zu einer formalen Realisierung leisten und dabei eine Vollständigkeit im Hinblick auf die ursprüngliche Idee anstreben.

Aufbauend auf der Formulierung und Untersuchung der wichtigsten funktionsorientierten Testverfahren in Kapitel 4 gibt Kapitel 5 einen Ausblick auf mögliche Verwendungen und Weiterentwicklungen der Ergebnisse dieser Arbeit, die insbesondere in der Automatisierung, Weiterentwicklung und gezielten Kombination der Verfahren bestehen. Konkret wird eine Umsetzung der Ergebnisse in einer integrierten Entwicklungsumgebung angeregt und skizziert.

Im Anhang sind die Verzeichnisse der gegebenen Definitionen, Verfahren und Abbildungen sowie der verwendeten Symbole und der verwendeten Literatur angegeben.

1.3 Danksagung

Herzlich bedanken möchte ich mich bei Herrn Prof. Dr. Liggesmeyer, der die thematische Ausrichtung dieser Arbeit angeregt und mich bei der Erstellung mit wertvollen Ratschlägen unterstützt hat. Ihm verdanke ich insbesondere die Ermunterung zu einer berufsbegleitenden wissenschaftlich Tätigkeit. Weiterhin möchte ich Herrn Prof. Dr. Gilg, dem Leiter des Fachzentrums „Simulation and Risk Management“ der Corporate Technology der Siemens AG, und Herrn Oliver Mäckel, dem Leiter eines dort angesiedelten Forschungsprojekts, für ihre Unterstützung meiner Arbeit danken.

2 Der funktionsorientierte Test und seine Bedeutung in der Softwarequalitätssicherung

Das folgende Kapitel gibt einen Überblick über das Gebiet der Softwarequalitätssicherung als thematisches Umfeld des funktionsorientierten Tests. Abschnitt 2.1 zeigt eine Abgrenzung wesentlicher Begriffe zur Qualitätssicherung von Software und ordnet das funktionsorientierte Testen in diesen Kontext ein. In Abschnitt 2.2 wird die historische Entwicklung und der Stand der Forschung von Prüfverfahren für Software betrachtet. Diesem wird in Abnschnitt 2.3 der Stand der Praxis speziell für den funktionsorientierten Test gegenübergestellt. Der sich daraus ergebende Handlungsbedarf wird in Abschnitt 2.4 abgeleitet. Er stellt Ausgangspunkt und Motivation für die vorliegende Arbeit dar.

Auf eine formale Definition der Begriffe wird in diesem Kapitel verzichtet, da hier lediglich das Umfeld des funktionsorientierten Tests beleuchtet wird. Begriffsdefinitionen zu den hier nur grob umrissenen Themen können in [ISO IEC 2382 93] eingesehen werden. Für umfassende Einführungen zu den Themen sei beispielsweise auf [Bal98, Bal00, Lig02] verwiesen. Eine Darstellung zum Begriff der Produktqualität im Umfeld des Software Engineering liefert [ISO IEC 9126 01]. Definitionen, die im weiteren Verlauf der Arbeit verwendet werden, werden in Kapitel 3 eingeführt.

2.1 Begriffe der Softwarequalitätssicherung

Wie im einleitenden Kapitel 1 beschrieben, kann die Softwareentwicklung als ein Prozess der Formalisierung betrachtet werden, der auf den allgemeinen Anforderungen zur Produktgestaltung aufsetzt und diese bis hin zu ihrer Realisierung in einer vollständig formalen, ausführbaren Form verfeinert. Ein solcher Prozess ist immer durch die intellektuelle Leistung des Menschen bestimmt und wird daher auch immer fehlerbehaftet sein. Dies führt zur Notwendigkeit einer prozessbegleitenden Qualitätssicherung bei der Softwareentwicklung. Ihr Stellenwert wird umso wichtiger, je komplexer die Anforderungen an das Endprodukt der Entwicklung sind und je umfangreicher die direkten, aber auch die indirekten wirtschaftlichen oder gesellschaftlichen Auswirkungen möglicher Qualitätsmängel ausfallen können.

Qualitätstechniken für Software können konstruktiver oder analytischer Natur sein. Konstruktive Qualitätstechniken fördern die Entwicklung qualitativ hochwertiger Systeme, indem Richtlinien, Vorgaben, Methoden und Werkzeuge für den Entwicklungsprozess bereitgestellt werden. Ihre Notwendigkeit und ihr Erfolg sind unumstritten, auch wenn der unmittelbare Einfluss der konstruktiven Methoden auf die Ergebnisqualität nur schwer bewiesen oder quantifiziert werden kann. Analytische Qualitätstechniken beurteilen hingegen direkt die Qualität der Ergebnisse. Sie stellen Prüfverfahren bereit, die auf Zwischen- und Endprodukte des Softwareentwicklungsprozesses angewendet werden. Durch den Vergleich der geplanten und der tatsächlichen Ergebnisse sichern sie den Erfolg der Entwicklung. Konstruktive und analytische Qualitätstechniken sind als komplementäre Verfahren zu betrachten, die sich gegenseitig bedingen und unterstützen und deren kombinierter Einsatz sinnvoll ist. Dies wird beispielsweise daran deutlich, dass konstruktive Techniken die Erstellung von Zwischenprodukten des Entwicklungsprozesses fordern, die eine schrittweise und umfassende analytische Prüfung erst ermöglichen.

Gegenstand der analytischen Qualitätssicherung können das Gesamtsystem, Teilsysteme oder auch Vorprodukte im Entwicklungsprozess sein. So unterliegen in den frühen Phasen der Entwicklung die Analyse- und Designdokumente der analytischen Qualitätssicherung, während der Implementierung

der Quelltext sowie in den Testphasen das ausführbare Programm, das Gesamtsystem oder jeweils Teile davon. Um Prüfverfahren allgemein beschreiben zu können, wird im Folgenden in Anlehnung an [Lig90] der Gegenstand der analytischen Qualitätssicherung unabhängig von seiner Ausprägung als Prüfling bezeichnet.

Ziel der analytischen Qualitätssicherung ist das Auffinden von Abweichungen der tatsächlichen Eigenschaften des Prüflings von den intendierten. So wird eine Korrektur der Abweichungen und damit eine Verbesserung der Qualität des Prüflings möglich. Kann keine Abweichung der Eigenschaften des Prüflings von den intendierten festgestellt werden, steigt das Vertrauen in seine Qualität. Neben den direkten Auswirkungen auf die Qualität des Prüflings und ihre Einschätzung werden durch analytische Qualitätssicherungsmaßnahmen weiterführende Informationen gewonnen. So weisen Schwachstellen am Produkt häufig auf Schwachstellen im Entwicklungsprozess hin. Eine sorgsame Analyse der Ergebnisse ermöglicht es, diese Schwachstellen zu lokalisieren, zu analysieren und zu beheben und somit eine kontinuierliche, von der Produktqualität getriebene Prozessverbesserung zu betreiben.

Referenz der analytischen Qualitätssicherung sind Vorgaben der funktionalen und qualitativen Eigenschaften des Prüflings. Vorgaben der funktionalen und qualitativen Eigenschaften des Endproduktes, die eine Sollvorgabe für die abschließende Qualitätssicherung bilden, sind im Allgemeinen im Rahmen eines Anforderungsdokumentes beziehungsweise einer Spezifikation beschrieben. Diese Spezifikation bildet die Grundlage für den Entwicklungsprozesses [VM97]. Sie liefert eine verbindliche Referenz für den gesamten Entwicklungsprozess und dient als Basis der Kommunikation zwischen Auftraggebern, Entwicklern und Testern. Wichtig ist daher, dass die Spezifikation möglichst vollständig, widerspruchsfrei und eindeutig interpretierbar ist [IEEE 830 98, Rup02]. Da dies mit informalen Methoden nur schwer zu erreichen ist, genügen die Spezifikationen der Praxis diesen Anforderungen meist nicht. Die intendierten Merkmalsausprägungen des Systems sind im Allgemeinen dennoch festgelegt durch implizite Anforderungen an das Endprodukt [Voa99]. Solche impliziten Anforderungen können beispielsweise durch die Erwartungen der Endkunden oder durch Rahmenbedingungen für den späteren Einsatz des Prüflings gegeben sein. Durch diese impliziten Soll-Vorgaben wird ein systematischer Soll-Ist-Vergleich erschwert; die analytische Qualitätssicherung setzt dann umfangreiche, beispielsweise erfahrungsbasierte Kenntnisse der Systemanforderungen voraus. Liegt eine Spezifikation für das Endprodukt vor, können daraus unter Berücksichtigung von Architektur- und Designentscheidungen Teilspezifikationen für enthaltene Teilsysteme abgeleitet werden. Die Ableitung kann in einem wiederholten Prozess von Design- und Anforderungsverfeinerung bis auf Komponentenebene vorgenommen werden [LefWid99]. Eine Spezifikation kann daher für beliebige Teilsysteme oder Komponenten im Softwareentwicklungsprozess formuliert sein.

Verfahren der analytischen Qualitätssicherung können danach unterschieden werden, ob die Software des analysierten Systems ausgeführt wird oder nicht. Zu den Verfahren, die nicht auf der Ausführung der Systemsoftware beruhen, zählen neben Reviews von Ergebnissen oder Zwischenergebnissen des Entwicklungsprozesses auch Verfahren der statischen Analyse, der symbolischen Analyse und der Programmverifikation. Erkenntnisse, die mit Hilfe dieser Verfahren gewonnen wurden, sind unabhängig vom dynamischen Verhalten einzelner Programmläufe gültig. Techniken der statischen Analyse, wie beispielsweise die Prüfung der Einhaltung von Programmierrichtlinien oder die Prüfung auf Anomalien im internen Datenfluss, sind heute weit verbreitet. Ihr Einsatz wird in verschiedenen Anwendungsfeldern durch Normen gefordert [IEC 61508 99]. Auch Techniken der symbolischen Analyse sind heute in eingeschränkter Form in der Praxis verfügbar, wobei der Schwerpunkt auf die Entdeckung von Ursachen potentieller Laufzeitfehler mit Hilfe von Datenabstraktionen gesetzt wird [CouCou79, Deu04]. Der Einsatz der symbolischen Analyse zur Identifikation der Pfadbereiche und

der hierüber berechneten Ausdrücke mit dem Ziel eines Nachweises der Konformität des Prüflings mit der Spezifikation hat sich hingegen als schwer anwendbar erwiesen [How77]. Verfahren der Programmverifikation, wie beispielsweise das Zusicherungsverfahren nach [Flo67], Beweise mit algebraischen Techniken [Mus80] oder Verfahren des Model Checking [ClaGruPel00], werden ebenfalls selten und nur für Kernalgorithmen von Systemen mit hohen Qualitäts- oder Sicherheitsanforderungen eingesetzt. Grund hierfür ist die Komplexität der Verfahren und die Schwierigkeit ihres Einsatzes. So sollte über den Einsatz formaler Verfahren bereits zu Beginn der Entwicklung entschieden werden, damit die Spezifikation in einer formalen Form erarbeitet und die weiteren Artefakte der Entwicklung geeignet gestaltet werden können. Für die Anwendung von Verifikationsverfahren sind spezielle Kenntnisse notwendig, beispielsweise beim Formulieren von Hypothesen und Invarianten. Dies wird in der praxisorientierten Einführung in [Lig02] deutlich. Theoretische Limitierungen für den Einsatz der verifizierenden Verfahren zeigt [ManWal78], eine Übersicht über den praktischen Einsatz formaler Methoden in Industrie und angewandter Forschung gibt [ClaWin96].

Verfahren, bei denen der Prüfling ausgeführt wird, werden als dynamische Testverfahren bezeichnet. Der Begriff „Dynamis“ stammt aus dem Griechischen und bedeutet Kraft, Vermögen oder Möglichkeit im Sinne der einem Gegenstand innewohnenden Anlagen [Kra92]. Dynamische Testverfahren untersuchen das Verhalten und die Eigenschaften des Prüflings durch Prüfung seiner Reaktion unter vorgegebenen Bedingungen. So kann das spätere Verhalten des Prüflings im Feldeinsatz vorweggenommen werden, was direkte Auswirkung auf das Vertrauen in die Komponente hat. Dies begründet den hohen Stellenwert der dynamischen Testverfahren im Rahmen der analytischen Qualitätssicherung. Da eine Ausführbarkeit von Entwicklungsprodukten der frühen Phasen des Systementwicklungsprozesses meist nicht gegeben ist, beschränken sich die dynamischen Verfahren häufig auf die Qualitätssicherung des Systems nach seiner Implementierung. Um die Mächtigkeit der dynamischen Testverfahren auch in frühen Phasen nutzbar zu machen und eine frühzeitige Analyse des Systemverhaltens zu ermöglichen, wird zunehmend die Ausführbarkeit von frühen Produkten des Systementwicklungsprozesses angestrebt [HarEA90, OstEA96, CobClaOst00]. Techniken in Spezifikation und Design, die eine simulative Auswertung ermöglichen, unterstützen diesen Ansatz.

Dynamische Testtechniken können zur Prüfung funktionaler und nicht-funktionaler Anforderungen eingesetzt werden. Funktionale Anforderungen beschreiben das geforderte funktionale Verhalten des Prüflings zu gegebenen Eingabedaten und in vorgegebenen Systemkonstellationen. Sie werden im Allgemeinen durch die Auswahl und Durchführung einzelner Testfälle geprüft. Nicht-funktionale Anforderungen beschreiben über die Funktionalität hinausgehende, generelle Eigenschaften des Prüflings, wie beispielsweise Laufzeitverhalten, Speicherbedarf, aber auch Bedienbarkeit eines Systems oder Portierbarkeit eines Programms [IEEE 830 98]. Soweit sie das intendierte Verhalten zur Laufzeit beschreiben, können auch sie mit speziellen dynamischen Testtechniken geprüft werden. Beispiele sind Last- und Stresstests oder stochastische Tests zur Zuverlässigkeitsbewertung.

Dynamische Testverfahren adressieren zwei unterschiedliche Fragestellungen: Zum einen unterstützen sie die Auswahl der Testdaten, aufgrund derer die Reaktion des Prüflings im dynamischen Test untersucht wird. Dies wird als Testdatenselektion bezeichnet. Zum anderen beurteilen sie die Testdatenadäquatheit, also die Eignung und Vollständigkeit gegebener Testdaten als Grundlage für die Prüfung eines Prüflings als Realisierung seiner Spezifikation. Beide Fragestellungen sind von großem praktischem Interesse und entsprechen zwei unterschiedlichen Blickwinkeln, unter denen dynamische Testverfahren zu betrachten sind. Eine theoretische Diskussion der Testdatenadäquatheit findet sich in [Wey83].

Für die Testdatenselektion und die Beurteilung der Testdatenadäquatheit verwenden die dynamischen Testverfahren unterschiedliche Referenzen. Diese Referenzen sind bestimmend für die Eigenschaften der Testverfahren und ermöglichen ihre Klassifizierung, wie beispielsweise in [Lig90] ausgeführt. Die folgende Zusammenstellung umreißt die Eigenschaften der wichtigsten Klassen von Testverfahren vor dem Hintergrund der verwendeten Referenz:

- Funktionsorientierte Testverfahren orientieren sich bei der Testdatenselektion und der Beurteilung der Testdatenadäquatheit an den funktionalen Anforderungen. Sie sind damit insbesondere für die systematische Prüfung der Umsetzung dieser Anforderungen im Prüfling geeignet. Die Struktur des Prüflings bleibt hierbei unberücksichtigt, so dass Testdaten auch dann als adäquat betrachtet werden können, wenn nicht alle Strukturelemente in der Prüfung betrachtet wurden. Wichtige Quellen zu Verfahren des funktionsorientierten Tests sind [Mye79, How80, Bei90].
- Strukturorientierte Testverfahren verwenden die Elemente der Struktur des Prüflings als Grundlage der Durchführung oder Bewertung des dynamischen Tests. Diese formale, maschineninterpretierbare Operationsbasis begünstigt eine formale Beschreibung der Testtechniken und liefert eine geeignete Grundlage für die Testdatenauswahl und die Formulierung und Quantifizierung der Testvollständigkeit. Dies ermöglicht eine umfassende Werkzeugunterstützung der strukturorientierten Verfahren. Eine systematische Ausrichtung an den zu prüfenden funktionalen Anforderungen wird durch die strukturorientierten Verfahren nicht geleistet: So werden die Anforderungen zwar zur Bewertung des Testergebnisses herangezogen, aber nicht bei der Auswahl und Bewertung der Vollständigkeit von Testdaten berücksichtigt. Quellen der wichtigsten strukturorientierten Verfahren sind [How76, How78, Mye79, WooHedHen80, LasKor83, RapWey85, ClaEA85, TayLevKel92].
- Diversifizierende Testtechniken beschreiben den vergleichenden Test verschiedener Versionen eines Prüflings. Diese können unabhängig voneinander als diversitäre Realisierung entwickelt worden sein, verschiedene Entwicklungsstände desselben Prüflings darstellen oder als künstliche Mutanten des Prüflings erzeugt worden sein. Wichtige Quellen zu diversifizierenden Testtechniken sind [DeMLipSay78, How82, ManKou01, RotEA01].
- Techniken des fehlerbasierten Testens unterstützen die gezielte Suche nach speziellen, oft syntaktisch definierten Fehlerarten und streben den Nachweis ihrer Abwesenheit für den Prüfling an. Sie können direkt die Auswahl der Testfälle steuern oder in einem diversifizierenden Test bei der Erzeugung von Mutanten des Prüflings verwendet werden. Neben der Betrachtung syntaktischer Fehlerklassen existieren auch Ansätze zur Einstreuung von Fehlern in den Datenzustand eines Programms, mit denen in Abhängigkeit von der Testbarkeit eines Programms Rückschlüsse auf die Vollständigkeit der verwendeten Testdaten abgeleitet werden können. Als Basis für die Beurteilung der Adäquatheit einer Menge von Testdaten wird die Unterscheidbarkeit der durch die Fehlerklassen erzeugten Mutanten bezüglich der Testergebnisse verwendet; die Spezifikation wird nur zur Bewertung der Testergebnisse herangezogen. Techniken des fehlerbasierten Testens sichern daher nicht die Vollständigkeit der Prüfung der Umsetzung von Anforderungen in der Software. Quellen zu Ansätzen des fehlerbasierten Testens sind beispielsweise [DeMLipSay78, Mye79, WhiCoh80, Mor90, Voa92, MorMur96].
- Stochastische Testtechniken beschreiben die Auswahl konkreter Testdaten mit stochastischen Verfahren. Ziel ist es, Fehler in Abhängigkeit von verschiedenen Nutzungsprofilen zu finden oder Aussagen zur Zuverlässigkeit von Software zu bestimmten Nutzungsszenarien zu erreichen. Gleichzeitig liefern sie eine Referenz für vergleichende Untersuchungen zur Effizienz und Wirtschaftlichkeit verschiedener Testverfahren. Auch bei den stochastischen Verfahren wird die Spezi-

fikation der funktionalen Anforderungen lediglich zur Beurteilung der Testergebnisse herangezogen, eine systematische Prüfung findet nicht statt. Wichtige Ansätze zum stochastischen Test sind in [ThaLipNel78, DurNta84, MusIanOku90, MilEA92, Voa92, HamVoa93, Gut95, Lyu95, Ham96, Voa99, Nta01] beschrieben.

2.2 Historische Entwicklung der Softwarequalitätssicherung

Seit Beginn der systematischen Nutzung von Software wurden Verfahren zu ihrer Qualitätssicherung entwickelt. Ein wissenschaftliches Interesse am Test von Softwaresystemen zeigte sich in den 70er Jahren in einer Grundsatzdiskussion über die theoretischen Möglichkeiten des Testens: Hier wurde das systematische Testen erstmals als eine eigenständige intellektuelle Tätigkeit verstanden, die über das spontane Ausprobieren der Software hinausging. Die Diskussion war getrieben vom Wunsch, eine möglichst umfassende Lösung des aufkeimenden Softwarequalitätsproblems zu finden. So stand die Suche nach „validen“ und „verlässlichen“ Testverfahren [GooGer75] im Vordergrund, die einen Korrektheitsnachweis unabhängig von Prüfling und Spezifikation durch den dynamischen Test hätten liefern könnten. Dass ein solches Verfahren nicht existieren kann und somit theoretische Grenzen auch für die Aussagekraft von Testverfahren bestehen, wurde bei der Untersuchung der Grenzen allgemeiner Verifikationsverfahren klar. Manna und Waldinger fassen die Ergebnisse wie folgt zusammen [ManWal78]:

„We can never be sure that specifications are correct.“
„No verification system can verify every correct program.“
„We can never be certain that a verification system is correct.“

Um dennoch auf theoretischer Basis die Mächtigkeit dynamischer Testverfahren untersuchen zu können, wurden in Folge dieser Erkenntnisse Aussagen bezüglich eines eingeschränkten Korrektheitsbegriffs angestrebt. So gelang der Nachweis der Abwesenheit von speziellen syntaktischen Fehlern mit dem Mutationentest [DeMLipSay78]. Weiterhin wurde eine Unterteilung des Eingabebereichs in Teilmengen angeregt, über denen eine Aussage bezüglich der Korrektheit oder Fehlerbehaftung des Prüflings durch Auswahl und dynamischen Test einzelner Repräsentanten getroffen werden könnte [WeyOst80]. Eine Definition der Adäquatheit von Testdaten anhand der Eignung ihres Informationsgehalts zur Inferenz des Prüflings scheiterte hingegen an seiner Gleichwertigkeit zum Korrektheitsnachweis [Wey83].

Parallel zu diesen theoretischen Überlegungen wurden praxisorientierte Fragestellungen untersucht. Allgemeine Untersuchungen zur Fehlerbehaftung von Software wurden angestellt [FosOst76, WhiCoh80]. Die Selektion geeigneter Testdaten und die Bewertung ihrer Adäquatheit wurde vor dem Hintergrund von Prüfling, Spezifikation oder Fehlererwartungen diskutiert, wodurch sich zahlreiche Verfahren entwickelten. So entwickelten sich funktionsorientierte Testverfahren ausgerichtet an den Beschreibungsformen der funktionalen Anforderungen an die Software. Eine frühe, aber häufig referenzierte Darstellung dieser Techniken findet sich in [Mye79], eine Untersuchung ihrer Effizienz bei der Fehlerfindung in [How80], Strategien für das zustandsbasierte Testen sind in [Cho78] beschrieben. Strukturorientierte Verfahren, die Elemente der internen Struktur der Software als Referenz für die Prüfstrategie verwendeten, entwickelten sich parallel: So erschien bereits 1976 eine Untersuchung zum Zweig- und Pfadüberdeckungstest [How76]. Der Zweigüberdeckungstest wurde bald in Lehrbüchern als Minimalkriterium im Hinblick auf die Testdatenadäquatheit [Mye79] propagiert. Die generelle Überlegenheit der im Allgemeinen nicht erreichbaren Pfadüberdeckung führte zur Entwicklung von weiteren Strategien zur Auswahl relevanter Teilpfade. Diese Auswahl wurde beispielsweise im Hin-

blick auf den Kontroll- [WooHedHen80] oder Datenfluss [LasKor83, RapWey85] der Software getroffen. Testverfahren, die speziell die in der Software verwendeten komplexen Bedingungen prüfen, wurden ebenfalls bereits Ende der 70er Jahre diskutiert [Mye79] und später auf Betreiben der Luft- und Raumfahrtindustrie weiterentwickelt [ChiMil94]. Als weiteres Adäquatheitskriterium entwickelte sich mit dem Mutationentest das fehlerbasierte Testen, das mit Hilfe syntaktisch erzeugter Varianten des Prüflings unabhängig von seiner Spezifikation und Struktur die Eignung der Testdaten zum Nachweis der Abwesenheit bestimmter syntaktischer Fehlerklassen beurteilt [DeMLipSay78].

Die Verschiedenartigkeit der vorgeschlagenen Ansätze führte zur Diskussion ihrer Eignung, die Entdeckung von Fehlern und den Aufbau von Vertrauen im dynamischen Test zu unterstützen. Empirische, analytische und stochastische Untersuchungen zum Vergleich der Verfahren wurden durchgeführt [DurNta84, BasSel87, HamTay90, WeyJen91, FraWei93, FraWey93]. In den meisten Untersuchungen wurden kontroll- und datenflussorientierte Techniken einander gegenübergestellt, das stochastische Testen mit zumeist gleichverteilten Testdaten diente als Referenz. Hierbei wurde erstmals der Sinn und Nutzen der systematischen Testtechniken gegenüber dem zufälligen Test in Frage gestellt und dem stochastischen Testen ein fester Stellenwert eingeräumt [DurNta84, HamTay90]. Weiterhin wurden in analytischen Untersuchungen im Hinblick auf die verfügbaren Programmierkonstrukte kontroll- und datenflussbasierte Verfahren mit den fehlerbasierten Ansätzen verglichen. Vergleiche mit funktionsorientierten Testtechniken wurden lediglich auf empirischer Basis vorgenommen [BasSel87]. Ein allgemein akzeptiertes Ergebnis der vergleichenden Untersuchungen konnte aufgrund von Problemen der Vergleichbarkeit der Untersuchungsbedingungen, der Annahmen und der Beurteilungskriterien, insbesondere aber wegen der verschiedenartigen Ausprägungen von Software nicht erreicht werden.

Neben der verfahrensorientierten Betrachtungsweise wurde auch die Eignung der Software selbst für einen effektiven Test analysiert. Hierzu wurden die Mechanismen untersucht, die aus inhärenten Fehlern im Code beobachtbare Ausfälle zur Laufzeit entstehen lassen [RicTho88, Mor90, Voa92, OffHay96]. Hieraus entwickelte sich ein Verständnis von Testbarkeit, das bis heute beim Entwurf und der Implementierung von Software berücksichtigt wird. Dies zeigt zunehmende Berücksichtigung von defensiven Programmierstrategien, die Verwendung von Zusicherungen in modernen Programmiersprachen, die zunehmende Forderung von Testbarkeit in Design und Implementierung und die Anerkennung der Testbarkeit als Qualitätsmerkmal für Software [IEC 61508 99, ISO IEC 9126 01].

Mit der Untersuchung der stochastischen Testtechniken, die – gleichverteilt oder einem vorgegebenen Nutzungsprofil folgend – mit zufällig generierten Testdaten operieren, wurde erneut die Frage nach dem primären Ziel des Testens diskutiert: War bisher der Fehlerfindung oberste Priorität zugesprochen [Mye79], so wurde nun der Vertrauensaufbau durch den Test betont [HamTay90, MilEA92, HamVoa93, Ham96, FraEA98]. Hier begann ein inhaltliches Vordringen auf das Gebiet der Zuverlässigkeitsbewertung für Software und Systeme [ThaLipNel78, MusIanOku90, HamVoa93, HowHua95, Lyu95, VoaMil95, BerStr96, Ham96, Voa99]. Die Verbindung zwischen Softwaretest und resultierender Produktqualität, die von besonderer Bedeutung für die industrielle Praxis ist, wurde untersucht. Prüfstrategien wurden vor dem Hintergrund unterschiedlicher Qualitätsanforderungen diskutiert: So wurde stochastisches Testen nach einem operationalen Profil als Technik zum Test von Software mit eher geringen Qualitätsanforderungen vorgeschlagen, während die Nutzung inverser Profile im stochastischen Test zur Absicherung hoher Qualitätsanforderungen angeregt wurde [Voa99]. Ein quantifizierter Zusammenhang zwischen den eingesetzten Techniken und der erreichten Qualität konnte nicht angegeben werden, so dass für sicherheitskritische Applikationen weiterhin die Notwendigkeit zum ergänzenden Einsatz formaler Methoden betont wurde [OstEA96, How98]. Die Wirkungsweise

kombinierter Prüfstrategien wurde auch auf theoretischer Basis untersucht [LitEA00]. Zusätzlich rückten weitere, speziell an den Problemen der Praxis ausgerichtete Optimierungsziele in den Blickpunkt der Studien, wie beispielsweise die aufgrund von Qualitätsmängeln des Softwareproduktes zu erwartenden Kosten [WeiWey88, Gut95] oder die Kosten der Testdurchführung [Nta01]. So reifte die Softwarequalitätssicherung zunehmend zu einer systematischen Ingenieurdisziplin heran, was sich auch in der Erarbeitung und Anwendung von Standards in der industriellen Praxis zeigte [IEC 61508 99].

Zunehmend wurde damit eine Werkzeugunterstützung der Praxis notwendig. Hier wurde zunächst als vordringliches Ziel die Testdatengenerierung im Hinblick auf einzelne Verfahren gesehen, die insbesondere für strukturorientierte und stochastische Techniken vorangetrieben wurde [Kor90]. Heute werden in diesem Bereich erfolgreich evolutionäre Verfahren und genetische Algorithmen verwendet [WegBarSth01, Ost04]. Hierbei ist eine automatisierte Messung der Testdatenadäquatheit notwendig, die für strukturorientierte Techniken durch Instrumentierung der relevanten Strukturelemente einfach erreicht werden kann. Eine automatische Testdatengenerierung für das funktionsorientierte Testen eignet sich zunächst für formale Beschreibungsformen der Spezifikation [DooFra94], weitere Ansätze für semi- oder informal beschriebene Spezifikationen werden punktuell untersucht [GroGriWeg93, LehWeg00, Sne03]. Während die Durchführung von Testfällen für Softwareprodukte und –teilprodukte einfach automatisiert werden konnte, gestaltete sich die automatische Auswertung der Testergebnisse schwierig. So muss die Bewertung eines Testergebnisses immer im Hinblick auf das intendierte Verhalten des Prüflings und damit auf die spezifizieren Anforderungen geschehen. Sie kann daher nur durch ein Werkzeug unterstützt werden, wenn die Spezifikation oder relevante Teile davon in einer maschineninterpretierbaren Form vorliegen [DooFra94]. Gegenstand der Forschung sind suboptimale Testorakel, die Abweichungen von speziellen, teilweise formal beschriebenen Anforderungen erkennen können, allerdings keine vollständige und zuverlässige Auswertung beliebiger Testergebnisse unterstützen [BerStr96, ManKou01]. Als ein fruchtbares Feld für die Werkzeugunterstützung der Praxis zeigte sich der Regressionstest, der die Durchführung und Auswertung bereits aufgezeichneter Testfälle erlaubt. Durch ihn kann die Aufzeichnung einer dynamischen Prüfung für mehrere Versionen eines Systems genutzt werden und eine vergleichende Auswertung durchgeführt werden, was zu signifikanten Einsparungen im dynamischen Test führt. Gegenstand der Forschung ist die zielgerichtete Auswahl von Testfällen zur Regression für eine wirtschaftliche Prüfung nach lokalen Änderungen der Software [HasSne03].

Betrachtet man speziell die Entwicklung des funktionsorientierten Tests, so wird deutlich, dass dieser in der wissenschaftlichen Entwicklung vergleichsweise wenig betrachtet wurde. Hier wird der Stand der Technik maßgeblich durch Ergebnisse vergangener Jahrzehnte und durch Ergebnisse auf anderen Gebieten der Softwaretechnik beeinflusst. Insbesondere die Weiterentwicklung von Techniken in Spezifikation und Design der Software durch geeignete Modellierungstechniken [DeM85, BooRumJac98, OMG03] schuf neue Grundlagen für angepasste Testtechniken. Verbreitete Beispiele sind zustandsbasierte Testtechniken [Cho78, TurRob93, DooFra94] und Testtechniken für Sequenzdiagramme [Bei90, Lig02]. Da beispielsweise auf dem Gebiet der objektorientierten Spezifikation und Modellierung intensive Forschung betrieben wird, kann angenommen werden, dass sich auch das funktionsorientierte Testen speziell auf diesen Gebieten weiterentwickeln wird.

2.3 Stand der Praxis

Betrachtet man den Stand der Praxis zur Softwareentwicklung am Beispiel Deutschlands, so zeigt sich, dass diese zunehmend zu einem wesentlichen Erfolgsfaktor der gesamtwirtschaftlichen Entwick-

lung wird. So hängen inzwischen mehr als 80% der deutschen Exporte von der Imformations- und Kommunikationstechnoligie ab [Sch06]. Umfangreiche Aufwendungen für die Softwareentwicklung sind die Folge, so zum Beispiel im Maschinen- und Anlagenbau, wo bis zu 50% der Entwicklungskosten der Informations- und Kommunikationstechnologie zuzurechnen sind [Sch06]. Betrachtet man im Gegenzug den Stand der industriellen Praxis in der analytischen Qualitätssicherung für Software, so kann beobachtet werden, dass nur wenige der in den letzten Jahrzehnten veröffentlichten Verfahren verbreitet sind und umfassend genutzt werden. Dies mag am oftmals unzureichenden Stellenwert der analytischen Qualitätssicherung in der professionellen Softwareentwicklung liegen, die gelegentlich als unproduktiver Kostentreiber angesehen wird und deren wesentlicher Beitrag zur Produktqualität und damit zum wirtschaftlichen Projekterfolg möglicherweise häufig unterschätzt wird. So kann beobachtet werden, dass in Projekten aufgrund terminlicher und finanzieller Engpässe in der Projektplanung oft eine unverhältnismäßige Kürzung im Bereich der analytischen Qualitätssicherung vorgenommen wird, da Auswirkungen hiervon auf das Produkt und seine Qualität nicht unmittelbar abzuschätzen sind. Als weitere Begründung für die schwache Durchdringung der industriellen Praxis mit den im vorigen Abschnitt beschriebenen Techniken kann der mangelnde Bekanntheitsgrad der Verfahren, ihrer Effizienz und Wirkungsweise sowie eine unzureichende Unterstützung durch Werkzeuge vermutet werden.

Eine verhältnismäßig hohe Verbreitung haben automatisierbare Ansätze und Verfahren zur Qualitätssicherung von Software gefunden. So werden Testfälle heute zunehmend aufgezeichnet und stehen für eine automatisierte Regression zur Verfügung. Auch simulative Ansätze zur Testdatengenerierung werden genutzt, insbesondere in Entwicklungsbereichen mit Modellunterstützung. Ein Beispiel für ihre Anwendung ist die Prüfung von Durchsatz und Verarbeitungsgeschwindigkeit der Software in Last- und Stresstests. Weiterhin werden Ansätze der analytischen Qualitätssicherung genutzt, die auf statischer Basis mit Hilfe von Compilern realisiert werden können. Beispiele hierfür sind die automatisierte Prüfung von Codierrichtlinien, die Datenflussanomalieanalyse, aber auch die Verwendung von Zusicherungen in Testversionen der Software zur Erhöhung ihrer Testbarkeit.

Bezüglich des dynamischen Tests kann eine sehr zögerliche Umsetzung der Ergebnisse der Forschung beobachtet werden. So entsprechen die heute eingesetzten Verfahren oft nicht den vor mehreren Jahrzehnten formulierten Minimalanforderungen [Mye79]. Selten gibt es eine formale Prüfung der Adäquatheit der Testfälle, oft entscheiden die Freigabetermine, das Management oder bestenfalls das intuitive Gefühl der Tester über das Ende einer Testphase. Lediglich in frühen Testphasen werden zunehmend Techniken genutzt, für die eine geeignete Werkzeugunterstützung vorliegt. Beispiele hierfür sind strukturorientierte Verfahren zur Messung der Testabdeckung. Ein weniger systematisches Vorgehen ist bei der Prüfung der funktionalen Eigenschaften von Software zu beobachten. Zwar liegt die Idee des funktionsorientierten Testens fast jedem intuitiven und nicht verfahrensorientierten Test zu Grunde, so dass vermutet werden kann, dass jede Software vor ihrer Freigabe einem wie auch immer gearteten funktionsorientierten Test unterzogen wird [Lig02]. Ein verfahrensorientiertes Vorgehen wird hierbei aber in vielen Fällen nicht verfolgt.

2.4 Handlungsbedarf

Software ist ein zunehmend wichtiger Träger von Innovation und von Differenzierungsmerkmalen zwischen Produkten verschiedener Wettbewerber geworden. Diese Entwicklung zeigt sich beispielsweise im Automobilbereich, wo Neuerungen zu großen Teilen in Software realisiert werden [Saa05]. Sie zielen direkt auf die Kaufentscheidungen technikbegeisterter Kunden und können darüber hinaus als

„Image“-bildende Produkteigenschaften gesehen werden. Für den wirtschaftlichen Erfolg von Entwicklungsprojekten ist es daher zunehmend von Bedeutung, dass die Funktionalität des Produkts exakt am Wunsch und Bedarf der Kunden ausgerichtet ist. Somit kommt der funktionalen Ausrichtung in der Produktentwicklung eine zunehmende Bedeutung für den wirtschaftlichen Erfolg oder Misserfolg zu.

Eine Verankerung der funktionalen Ausrichtung im konstruktiven Vorgehen der Software- und Systementwicklung wird heute zunehmend angestrebt, beispielsweise auch in der Automobilindustrie [LevSch06]: So sind die Techniken der Anforderungsanalyse und der Anforderungsverfolgung Gegenstand wachsenden Interesses. Auch Modellierungstechniken werden zunehmend verfeinert, standardisiert und werkzeugtechnisch unterstützt. In der analytischen Qualitätssicherung steht die Hinwendung zu den funktionsorientierten Techniken jedoch noch aus: Hier werden zunächst Techniken eingeführt, die sich an der Implementierung des Prüflings orientieren und daher einfach zu automatisieren sind. So kann vermutet werden, dass eine verfahrensbasierte Analyse der Anforderungen zur systematischen Prüfung ihrer Realisierung mit Hilfe von funktionsorientierten Testverfahren eher selten durchgeführt wird. Diese mangelnde Umsetzung des funktionsorientierten Tests steht der Ausrichtung auf funktionale Aspekte in der Entwicklung entgegen. Sie stellt ein Risiko für die Qualität der Umsetzung funktionaler Anforderungen und somit für den Gesamterfolg der Entwicklungsprojekte dar.

Konkret formuliert wird die Forderung einer an den funktionalen Anforderungen ausgerichteten Qualitätssicherung für Software und softwarebasierte Systeme in Normen und Standards zur Systementwicklung. So spricht die [IEC 61508 99] für die Entwicklung von Software zur funktionalen Sicherheit starke Empfehlungen für den Einsatz von funktionsorientierten Testtechniken aus. Die [EN 50128 01] dehnt diese Empfehlung auch auf Software ohne Sicherheitsrelevanz aus.

Hieraus ergibt sich die Notwendigkeit, orientiert an den Spezifikationstechniken der Praxis eine umfassende Unterstützung für eine systematische funktionsorientierte Qualitätssicherung bereitzustellen. Diese Unterstützung muss insbesondere dafür geeignet sein, eine Qualitätssicherung der Ergebnisse des Entwicklungsprozesses im Hinblick auf die ursprüngliche, informale Darstellung der Anforderungen an das System zu leisten. Als Grundlage hierfür ist zunächst eine umfassende methodische Beschreibung der funktionsorientierten Testtechniken bereitzustellen. Nur auf einer solchen Basis kann eine aussagekräftige Formulierung von funktionsorientierten Kriterien zur Auswahl von Testdaten und zur Vollständigkeitsbewertung des Tests erfolgen, die sich auch zum Nachweis einer Normkonformität eignet. Eine weitgehend formalisierte Beschreibung der funktionsorientierten Testtechniken ist darüber hinaus notwendige Grundlage einer Werkzeugunterstützung, die ein systematisches und effizientes Vorgehen bei Testdatenauswahl, Testdurchführung und Testvollständigkeitsbewertung unterstützt. Dies stellt die Motivation für die vorliegende Arbeit dar, die die Formalisierungsmöglichkeiten und die nicht formalisierbaren Aspekte des funktionsorientierten Tests aufzeigt und untersucht.

3 Ein Modell zur Beschreibung des dynamischen Tests von Software

Als Grundlage der formalisierten Beschreibung der funktionsorientierten Testverfahren im folgenden Kapitel 4 wird in diesem Kapitel eine formalisierte Beschreibung des allgemeinen Vorgehens im dynamischen Test gegeben. Abschnitt 3.1 trifft hierzu grundsätzliche Annahmen, die es ermöglichen, das dynamische Testen als einen wohldefinierten und endlichen Prozess zu beschreiben. Bei den hierfür zu treffenden Einschränkungen steht die Orientierung am Vorgehen in der Praxis im Vordergrund. Auf dieser Basis kann in Abschnitt 3.2 das typische Vorgehen im dynamischen Test formalisiert werden. Hier werden unter anderem die zentralen Begriffe des dynamischen Tests, des dynamischen Testlaufs und des dynamischen Testverfahrens definiert und mit den Beschreibungstechniken der Mengenlehre und der Prädikatenlogik beschrieben. Abschnitt 3.3 bewertet die so gewonnene formalisierte Beschreibung des dynamischen Tests und zeigt ihre Grenzen und weitere Anwendungsmöglichkeiten auf. Hier werden auch mögliche Lockerungen der in Abschnitt 3.1 getroffenen Annahmen diskutiert, die auf eine Erweiterung des Anwendungsbereichs des in Abschnitt 3.2 definierten Modells des dynamischen Tests zielen.

3.1 Grundlegende Annahmen zum Testprozess

Um ein grundlegendes Verständnis zum Vorgehen im dynamischen Test zu schaffen, sei zunächst angenommen, dass die Anforderungen an den Prüfling im Rahmen einer Spezifikation zusammengestellt sind. Der Prüfling selbst liege als Versuch der Realisierung der in der Spezifikation geforderten Eigenschaften vor und besitze eine ausführbare Semantik, die eine dynamische Prüfung ermöglicht. Im Rahmen des dynamischen Tests wird nun eine Menge von Testdaten ausgewählt, mit denen der Prüfling ausgeführt wird. Die Ergebnisse der Anwendung des Prüflings auf die Testdaten werden auf Konformität mit den Anforderungen der Spezifikation überprüft. Abweichungen zeigen die Notwendigkeit einer Korrektur des Prüflings, Übereinstimmung wird als Indiz für eine Erfüllung der spezifizierten Anforderungen gewertet. Zusätzlich wird bezüglich der Menge der verwendeten Testdaten die Vollständigkeit beziehungsweise Adäquatheit des Tests beurteilt.

Dieses intuitiv umrissene Vorgehen fasst den dynamischen Test als Untersuchung einer durch den Prüfling beschriebenen Zuordnung an gegebenen Datenpunkten auf Übereinstimmung mit der Spezifikation auf. Implizit ist hierin die Annahme einer Ein-Schritt-Verarbeitungssemantik enthalten, wie sie bereits in den 70er und frühen 80er Jahren formuliert wurde. Grundidee ist, dass in einem Verarbeitungsschritt einem Eingabedatum genau ein Ausgabedatum durch den Prüfling zugeordnet wird, soweit das Programm terminiert. Die Annahme einer Ein-Schritt-Verarbeitungssemantik bringt einen großen Vorteil mit sich: Sie ermöglicht eine einfache formale Beschreibung des dynamischen Tests mit Hilfe von Zuordnungen und Relationen. Gravierender Nachteil ist hingegen die enthaltene Einschränkung bezüglich des Prüflings: So eignet sich diese Semantik insbesondere für Programmsysteme mit einfacher Benutzerschnittstelle. Als problematisch erweist sie sich jedoch bei der Beschreibung von Programmsystemen mit einem „Gedächtnis“, das Informationen über mehrere Ausführungen sammelt, sowie zur Beschreibung von Programmsystemen mit komplexen Schnittstellen wie beispielsweise einer Dateischnittstelle oder Datenbankanbindung, und zur Beschreibung von interaktiven Programmen und zur Beschreibung von reaktiven technischen Systemen, die Signale über eine beliebige Dauer hinweg auswerten und generieren. Für solche Systeme ist die Anwendbarkeit der auf

Basis der Ein-Schritt-Verarbeitungssemantik definierten Verfahren im Einzelfall zu prüfen. Ist es für ein interaktives Programm beispielsweise möglich, die Bedienung auf eine endliche Zahl von Eingaben einzuschränken, können die endlichen Folgen der Ein- und Ausgabewerte als Vektoren aufgefasst werden, deren programmtechnische Verarbeitung der Ein-Schritt-Verarbeitungssemantik genügt. Praktikabler mag es sein, den Zustand des Systems als einen weiteren Parameter der Eingabedaten aufzunehmen, so dass eine Beschreibung des Prüflings und der Spezifikation im Sinne der Ein-Schritt-Verarbeitungssemantik möglich wird [TurRob93, Voa99]. Auch durch eine hinreichend allgemeine Definition der möglichen Ein- und Ausgaben des Prüflings, die beispielsweise auch Signale über einem Kontinuum zulässt, kann möglicherweise eine Modellierung entsprechend der Annahme erreicht werden. Im Folgenden wird zugunsten der Einfachheit der Beschreibung die Ein-Schritt-Verarbeitungssemantik vorausgesetzt. Diese stellt auch in vielen funktionsorientierten Testverfahren eine Grundlage dar, wie die Abschnitte 4.1 bis 4.3 des folgenden Kapitels zeigen werden. Bei der Betrachtung von Prüflingen mit Gedächtnis oder internem Zustand in Abschnitt 4.4 werden Gedächtnis beziehungsweise Zustand dann als Bestandteil der Ein- und Ausgabedaten erfasst, so dass auch hier die Annahme der Ein-Schritt-Verarbeitungssemantik zugrunde gelegt werden kann.

Annahme 3-1: Ein-Schritt-Verarbeitungssemantik

Für den Prüfling wird die Ein-Schritt-Verarbeitungssemantik vorausgesetzt.

Um die Entscheidbarkeit und Durchführbarkeit des oben beschriebenen Vorgehens im dynamischen Test zu garantieren, sind weitere Annahmen bezüglich Spezifikation und Prüfling zu treffen. So ist vorauszusetzen, dass die Spezifikation vollständig und eindeutig interpretierbar ist. Diese Annahme ist unumgänglich, da nur so eine Entscheidbarkeit der Testauswertung garantiert werden kann. Die Vollständigkeit der Spezifikation stellt sicher, dass für jedes Testdatum mindestens eine intendierte Reaktion des Prüflings spezifiziert ist. Die eindeutige Interpretierbarkeit garantiert, dass die Konformität des Testergebnisses mit der Spezifikation objektiv überprüft werden kann. Dies ist nicht zu verwechseln mit einem eindeutig vorgeschriebenen Testausgang: Zu einem Testdatum können mehrere unterschiedliche Testergebnisse der Spezifikation entsprechen, die jedes für sich ein zulässiges Ergebnis darstellen. Vollständigkeit und eindeutige Interpretierbarkeit der Spezifikation sind in der Praxis oft nicht gewährleistet, da sie mit informalen Spezifikationstechniken kaum zu erreichen sind, formale Techniken jedoch selten genutzt werden. Vor dem Hintergrund des abstrakten Spezifikationsbegriffs aus Kapitel 2.1, der die Spezifikation als Gesamtheit der intendierten Merkmalsausprägungen umfasst, kann diese Annahme als Grundlage für die Entscheidbarkeit im Testprozess jedoch idealisiert getroffen werden.

Annahme 3-2: Eindeutigkeit und Vollständigkeit der Spezifikation

Die Spezifikation beschreibt vollständig und eindeutig interpretierbar die Anforderungen an den Prüfling.

Um neben der Entscheidbarkeit auch die Durchführbarkeit des Testprozesses in endlicher Zeit zu gewährleisten, ist die Beschränkung auf eine endliche Menge von Testdaten im dynamischen Test notwendig. Da dies eine realistische Annahme für jeden Testprozess in der Praxis ist, scheint sie zunächst natürlich und somit allgemein akzeptabel. Bei der Anwendung von Verfahren zur Testdatenselektion oder zur Testdatenadäquatheit muss diese Annahme allerdings explizit beachtet werden – man

betrachte als Beispiel die in der Praxis meist nicht gegebene Durchführbarkeit des Verfahrens der Pfadabdeckung und verwandter Verfahren.

Annahme 3-3: Endlichkeit der Menge der Testdaten

Der dynamische Test beschränkt sich auf eine endliche Menge von Testdaten.

Weiterhin wird zur Garantie der Durchführbarkeit des dynamischen Tests in endlicher Zeit die Angabe eines Performanzkriteriums im Rahmen der Spezifikation angenommen, das für jeden Eingangsdatensatz eine maximal akzeptable Systemlaufzeit bis zur eindeutigen Feststellung der Systemreaktion vorgibt. Diese Annahme stellt die Durchführbarkeit der einzelnen Testfälle und damit des dynamischen Tests in endlicher Zeit sicher. Als alternative Bedingung dazu scheint sich die Einschränkung auf Programme anzubieten, die in endlicher Zeit terminieren. Da das Problem der Programmtermination jedoch nicht entscheidbar ist und da aufgrund der Bedürfnisse der Praxis die Einschränkung des dynamischen Tests auf eine solche Klasse von Programmen nicht sinnvoll ist, scheint die obige Annahme eines Performanzkriteriums im Rahmen der Spezifikation eher akzeptabel und wird im Folgenden vorausgesetzt.

Annahme 3-4: Performanzkriterium

Zu jedem Eingangsdatensatz existiert ein Performanzkriterium in der Spezifikation, das eine maximal akzeptable Reaktionszeit des Prüflings festlegt.

Um dem Test eine grundsätzliche Aussagekraft zu verleihen, wird weiterhin ein deterministisches Programmverhalten vorausgesetzt. Diese Annahme ermöglicht es, aus einem bestandenen Test auch für zukünftige Anwendung des Prüflings auf den getesteten Daten ein spezifikationskonformes Verhalten vorauszusetzen und somit eine eindeutige Qualitätsaussage zu treffen. Gleichzeitig stellt die Annahme im Hinblick auf die Praxis eine echte Einschränkung dar: So wird ein Nicht-Determinismus häufig dann angenommen, wenn die Ursache eines variablen Verhaltens der Software nicht grundlegend geklärt, modelliert oder systematisch nachgebildet werden kann. Nicht-Determinismus kann zum Beispiel bei der Nutzung geteilter Ressourcen zwischen Prozessen oder der Einbindung externer Einflüsse und menschlicher Interaktion angenommen werden. Für solche Systeme ist die Aussagekraft des dynamischen Tests beschränkt: Zu ihrem Test können simulative Methoden zur Erzeugung einer Stichprobe genutzt werden, die eine Einschätzung des Verhaltens des Prüflings in Form einer statistischen Aussage ermöglicht. Deterministische Aussagen über das Verhalten des Prüflings auf einzelnen Testdaten können jedoch nicht gewonnen werden. Um für die folgende Untersuchung eine einfach handhabbare Beschreibung des Vorgehens im dynamischen Test in Anlehnung an die klassische Betrachtungsweise zu erreichen, wird im Folgenden ein deterministisches Verhalten des Prüflings vorausgesetzt.

Annahme 3-5: Deterministisches Verhalten des Prüflings

Für den Prüfling wird ein deterministisches Verhalten vorausgesetzt, dass von Eingabedaten beziehungsweise Eingangskonstellationen eindeutig bestimmt wird.

Die Annahme der Ein-Schritt-Verarbeitungssemantik ermöglicht eine einfache Beschreibung der funktionalen und relationalen Aspekte von Prüfling und Spezifikation auf Basis der Ein- und Ausgangsda-

ten. Die Annahmen der vollständigen und eindeutig interpretierbaren funktionalen Spezifikation, der endlichen Menge an Testdaten, des impliziten Performanzkriteriums und des deterministischen Programmverhaltens garantieren darüber hinaus die Durchführbarkeit und Entscheidbarkeit des dynamischen Tests. Diese Annahmen werden daher im Folgenden für die Beschreibung des Testprozesses vorausgesetzt. Weitere Annahmen, mit Ausnahme der hier immer zugrunde gelegten Axiome der Mengenlehre und der natürlichen Zahlen [ReiSoe01], werden nicht benötigt.

3.2 Formale Beschreibung des dynamischen Tests

Die mathematische Formulierung und Untersuchung des Testprozesses wurde besonders in den 70er und 80er Jahren vorangetrieben, als die theoretischen Möglichkeiten des Testens sowie die Mächtigkeit spezieller Verfahren untersucht wurde [GooGer75, How80, WeyOst80]. Eine umfassende theoretische Aufbereitung und Erweiterung der Ergebnisse findet sich in [Gou83]. Die im Folgenden eingeführte Notation nutzt den von Gourlay vorgeschlagenen Formalismus, der die früheren Ansätze von Goodenough und Gerhart, Howden sowie Weyuker und Ostrand verallgemeinert. Die bei Gourlay sehr allgemein gehaltenen Definitionen werden hier für den in Abschnitt 3.1 umrissenen Testprozesses präzisiert. Ziel ist die konsistente Definition der zentralen Begriffe des dynamischen Tests mit Hilfe von Beschreibungstechniken der Mengenlehre und Prädikatenlogik. In Abschnitt 3.2.1 werden die Grundlagen des dynamischen Tests eingeführt. Der Begriff des Prüflings und der Spezifikation über einem gegebenen Ein- und Ausgabedatenbereich wird definiert, die Korrektheit des Prüflings bezüglich der Spezifikation wird beschrieben. Abschnitt 3.2.2 führt die zentralen Begriffe des dynamischen Testfalls, des dynamischen Testlaufs und des dynamischen Testverfahrens eines Prüflings bezüglich seiner Spezifikation ein und definiert ihr Bestehen. Auch Kombination und Subsumption dynamischer Testverfahren werden beschrieben. In Abschnitt 3.2.3 wird ein erweitertes Vorgehen zur Auswertung des dynamischen Tests untersucht und formalisiert.

3.2.1 Prüfling und Spezifikation über dem Ein- und Ausgabebereich

Aus Grundlage für eine Beschreibung der funktionalen Eigenschaften von Prüfling und Spezifikation werden zunächst die zugrunde gelegten Ein- und Ausgabebereiche eingeführt.

Definition 3-1: Ein- und Ausgabebereich, Ein- und Ausgabeparameterbereich, Ein- und Ausgabeparameter

(a) Als Eingabebereich I und Ausgabebereich O sind beliebige Mengen zugelassen.

(b) Können Ein- und Ausgabebereich als kartesische Produkte

$$I = I_1 \times I_2 \times \ldots \times I_k \qquad (k \in \aleph)$$
$$O = O_1 \times O_2 \times \ldots \times O_l \qquad (l \in \aleph)$$

einer endlichen Zahl beliebiger Mengen angegeben werde, so werden die Mengen I_1, I_2, …, I_k und O_1, O_2, …, O_l als Eingabe- beziehungsweise Ausgabeparameterbereiche bezeichnet. In diesem Fall werden die Elemente $i \in I$ und $o \in O$ als Ein- beziehungsweise Ausgabedatensätze bezeichnet und als Tupel

$$i = (i_1, i_2, \ldots, i_k) \qquad (i_1 \in I_1, i_2 \in I_2, \ldots, i_k \in I_k, k \in \aleph)$$
$$o = (o_1, o_2, \ldots, o_l) \qquad (o_1 \in O_1, o_2 \in O_2, \ldots, o_l \in O_l, l \in \aleph)$$

notiert. Die Elemente $i_1, i_2, \ldots, i_k$ und $o_1, o_2, \ldots, o_l$ werden als Ein- beziehungsweise Ausgabeparameter bezeichnet.

Für Systeme, deren Schnittstellen innerhalb eines digitalen Rechnersystems liegen, kann für den Ein- und Ausgabebereich zusätzlich gefordert werden, dass lediglich im Rechner darstellbare Daten enthalten sind. Zur Betrachtung technischer Systeme, die außer elektronischen auch noch weitere Komponenten wie beispielsweise mechanische, pneumatische oder pyrotechnische enthalten können, ist die oben gegebene, allgemein gehaltene Formulierung besser geeignet. Die Darstellung des Ein- und Ausgabebereichs in Produktform ermöglicht die Festlegung von Schnittstellen mit endlich vielen Parametern für Prüfling und Spezifikation. Sie ist notwendig für die Formulierung spezieller dynamischer Testverfahren, die sich auf Parameter und deren Kombinationen beziehen. Die Parameter entstammen beliebigen, festen Parameterbereichen, bezüglich derer keine weiteren Annahmen getroffen werden.

Auf Basis dieser Definition für den Ein- und Ausgabebereich kann der Begriff der Spezifikation wie folgt definiert werden.

Definition 3-2: Spezifikation, Menge der Spezifikationen über einem Ein- und Ausgabebereich

(a) Eine Spezifikation s über dem Eingabebereich I und Ausgabebereich O wird durch eine dreistellige Relation über Eingabe- und Ausgabebereich und den positiven reellen Zahlen, also durch eine Teilmenge des kartesischen Produktes $I \times O \times \Re^+$, angegeben, die der Bedingung

$$\forall(i \in I)\ \exists(o \in O)\ \exists(r \in \Re^+)\ (i, o, r) \in s \qquad \text{(Vollständigkeit)}$$

genügt.

(b) $\Sigma_{I, O}$ bezeichnet im Folgenden die Menge aller Spezifikationen über Eingabebereich I und Ausgabebereich O.

Diese Definition der Spezifikation ist so zu interpretieren, dass $(i, o, r) \in s$ genau dann gilt, wenn die Ausgangsdaten o als Systemreaktion auf die Eingangsdaten i innerhalb der durch r vorgegebenen Zeit den Anforderungen beziehungsweise dem intendierten Systemverhalten entsprechen. Die in Annahme 3-1 geforderte Ein-Schritt-Verarbeitungssemantik ermöglicht hierbei die Beschreibung der Spezifikation als Relation. Die in Annahme 3-2 formulierte Forderung der Vollständigkeit an die Spezifikation s wird hier durch die Vollständigkeitsbedingung gewährleistet. Sollen, beispielsweise aus technischen Gründen, Elemente des Eingabebereichs aus der Betrachtung ausgeschlossen werden, so kann dies über eine Einschränkung der Menge der Eingabedaten erreicht werden. Die eindeutige Interpretierbarkeit ermöglicht die Beschreibung in Form einer Relation. Die Berücksichtigung der Performanz in der Spezifikation ist auf Annahme 3-4 zurückzuführen, die eine Durchführung des dynamischen Tests in endlicher Zeit ermöglicht.

Da eine Spezifikation nach dieser Definition wegen der möglichen Mehrdeutigkeit keine Abbildung im mathematischen Sinn ist, können die für Abbildungen definierten Begriffe des Bildes und Urbildes nicht verwendet werden. Um dennoch eine komfortable Notation zur Bezeichnung der zu einem gegebenen Eingabedatum spezifizierten Ausgabedaten und ihrer Parameter zu ermöglichen, wird der Begriff des Wertebereichs wie folgt eingeführt.

Definition 3-3: Wertebereich einer Spezifikation, Wertebereich einer Spezifikation zu einem Eingabedatum, Wertebereich einer Spezifikation über dem Ausgabebereich, Wertebereich einer Spezifikation über einem Ausgabeparameterbereich

Gegeben sei eine Spezifikation $s \in \Sigma_{I, O}$.

(a) Die Menge

$$W^s := \{ (o, r) \in O \times \Re^+ : \exists (i \in I)\ (i, o, r) \in s \}$$

wird im Folgenden als Wertebereich der Spezifikation s über $O \times \Re^+$ bezeichnet, die Menge

$$W^{s, i} := \{ (o, r) \in O \times \Re^+ : (i, o, r) \in s \}$$

als Wertebereich der Spezifikation s über $O \times \Re^+$ zum Eingabedatum i.

(b) Die Menge

$$W^{O, s} := \{ o \in O : \exists (i \in I)\ \exists (r \in \Re^+)\ (i, o, r) \in s \}$$

wird im Folgenden als Wertebereich der Spezifikation s über dem Ausgabebereich O bezeichnet, die Menge

$$W^{O, s, i} := \{ o \in O : \exists (r \in \Re^+)\ (i, o, r) \in s \}$$

als Wertebereich der Spezifikation s über O zum Eingabedatum i.

(c) Die Menge

$$W^{O_w, s} := \{ x \in O_w : \exists (i \in I)\ \exists (r \in \Re^+)\ \exists (o \in O)\ ((o_w = x) \wedge ((i, o, r) \in s)) \}$$

wird im Folgenden als Wertebereich der Spezifikation s über dem Parameterbereich O_w bezeichnet, die Menge

$$W^{O_w, s, i} := \{ x \in O_w : \exists (r \in \Re^+)\ \exists (o \in O)\ ((o_w = x) \wedge ((i, o, r) \in s)) \}$$

als Wertebereich der Spezifikation s über dem Parameterbereich O_w zum Eingabedatum i.

Der relationale Charakter der hier verwendeten Definition der Spezifikation erleichtert eine Formulierung des Vorgehens in einem dynamischen Test mit funktionaler Auswertung und damit die Formalisierung funktionsorientierter Verfahren. Weniger geeignet ist er zur Formulierung von nicht funktionalen Aspekten der Spezifikation wie beispielsweise Zuverlässigkeits-, Verfügbarkeits- und Sicherheitsanforderungen. Mögliche Erweiterungen zur Berücksichtigung nicht funktionaler Anforderungen werden in Abschnitt 3.3.3 diskutiert. Da für eine Diskussion des funktionsorientierten Tests die relationale Darstellung ausreichend ist, wird Definition 3-2 als Grundlage der folgenden Betrachtungen verwendet.

Entsprechend der Definition der Spezifikation als Relation kann der Prüfling als Zuordnung über dem Ein- und Ausgabebereich und dem Bereich der positiven reellen Zahlen beschrieben werden. Diese Form der Definition setzt die in Annahme 3-1 geforderte Ein-Schritt-Verarbeitungssemantik und das in Annahme 3-5 geforderte deterministische Verhalten des Prüflings voraus.

Definition 3-4: Prüfling, Menge der Prüflinge über einem Ein- und Ausgabebereich

(a) Ein Prüfling p über dem Eingabebereich I und dem Ausgabebereich O wird beschrieben durch eine partielle Funktion des Eingabebereichs auf das kartesische Produkt des Ausgabebereichs und der positiven reellen Zahlen

$$p: \quad I \quad \rightarrow \quad (O \times \Re^+) \cup \perp$$

$$i \quad \mapsto \quad \begin{cases} (p_{out}(i), p_{perf}(i)) & \text{falls p für i terminiert,} \\ \perp & \text{sonst.} \end{cases}$$

Dabei bezeichnet $p_{out}(i) \in O$ die vom Prüfling berechneten Ausgabedaten und $p_{perf}(i) \in \Re^+$ die benötigte Berechnungszeit, falls p für i terminiert.

(b) $\Pi_{I, O}$ bezeichnet im Folgenden die Menge aller Prüflinge über Eingabebereich I und Ausgabebereich O.

Durch einen Prüfling p wird jedem Element i des Eingabebereichs, für das der Prüfling terminiert, das berechnete Ausgabeelement $p_{out}(i)$ sowie die benötigte Rechenzeit $p_{perf}(i)$ zugeordnet. Die Funktion p ergibt sich so aus den Komponenten $p = (p_{out}, p_{perf})$. Sie kann nicht als totale Funktion über ganz I angenommen werden, da auch die Betrachtung von Programmen möglich sein soll, die für gewisse Eingabedaten nicht terminieren oder fehlerhaft abbrechen.

Wichtig zur Betrachtung von Prüfverfahren ist weiterhin die Formulierung der Korrektheit eines Prüflings $p \in \Pi_{I, O}$ im Hinblick auf eine Spezifikation $s \in \Sigma_{I, O}$, die entsprechend der gängigen Anschauung wie folgt formuliert werden kann:

Definition 3-5: Korrektheit von Prüflingen bezüglich Spezifikationen

Die Korrektheit $corr_{I, O} \subseteq \Pi_{I, O} \times \Sigma_{I, O}$ wird definiert durch die zweistellige Relation

$\forall(p \in \Pi_{I, O}) \; \forall(s \in \Sigma_{I, O})$

$(((p, s) \in corr_{I, O}) :\Leftrightarrow \forall(i \in I) \; ((p(i) \neq \perp) \wedge ((i, p_{out}(i), p_{perf}(i)) \in s)))$

über der Menge der Prüflinge und der Spezifikationen.

Ein Prüfling wird bezüglich einer Spezifikation demnach genau dann als korrekt bezeichnet, wenn p für alle Eingabedaten terminiert und Ausgabedatum und Performanz der Spezifikation entsprechen. Dies impliziert, dass ein Prüfling, der für gewisse Eingabedaten nicht terminiert, bezüglich keiner Spezifikation korrekt sein kann.

Die Herkunft eines Prüflings $p \in \Pi_{I, O}$ oder einer Spezifikation $s \in \Sigma_{I, O}$ wird hier nicht näher beleuchtet, da im Vordergrund des Interesses Prüfverfahren und ihre Eigenschaften stehen, die zum Test vorgegebener Prüflinge und Spezifikationen verwendet werden. Auch die formale Gestaltung von Spezifikation und Prüfling wird im Rahmen der hier gegebenen grundlegenden Definitionen nicht eingeschränkt, da nur den funktionalen Aspekt über Ein- und Ausgabedaten und das Verhalten bezüglich der Performanz betrachtet werden. So lassen die Definitionen im Rahmen der zugrunde gelegten Annahmen größtmögliche Freiheit, ermöglichen aber gleichzeitig eine Betrachtung der für den funktionsorientierten Test relevanten funktionalen Aspekte.

3.2.2 Dynamisches Testen eines Prüflings im Bezug auf seine Spezifikation

Um den dynamischen Test eines Prüflings bezüglich einer Spezifikation über einem Ein- und Ausgabebereich zu erfassen, muss das Vorgehen der Testdurchführung und –auswertung beschrieben werden. Dazu wird zunächst der Begriff eines dynamischen Testfalls eingeführt und seine Bewertung vor dem Hintergrund der Spezifikation beschrieben. Darauf aufbauend werden die zentralen Begriffe des dynamischen Testlaufs und des dynamischen Testverfahrens definiert.

Definition 3-6: Dynamischer Testfall, Bestehen eines dynamischen Testfalls

(a) Ein dynamischer Testfall für einen Prüfling $p \in \Pi_{I, O}$ bezüglich einer Spezifikation $s \in \Sigma_{I, O}$ besteht in der Anwendung des Prüflings p auf ein Testdatum $t \in I$. Das Ergebnis $p(t) = (p_{out}(t), p_{perf}(t))$ setzt sich zusammen aus dem Ergebnis über dem Ausgabebereich $p_{out}(t)$ und der erreichten Performanz $p_{perf}(t)$.

(b) Ein dynamischer Testfall gilt dann als bestanden, wenn das Ergebnis der Testdurchführung der Spezifikation entspricht. Das Bestehen von dynamischen Testfällen wird durch die dreistellige Relation $ok_{I, O} \subseteq I \times \Pi_{I, O} \times \Sigma_{I, O}$ mit

$$\forall(t \in I)\ \forall(p \in \Pi_{I, O})\ \forall(s \in \Sigma_{I, O})$$
$$(((t, p, s) \in ok_{I, O}) :\Leftrightarrow ((p(t) \neq \perp) \wedge ((t, p_{out}(t), p_{perf}(t)) \in s)))$$

beschrieben.

Ein dynamischer Testfall für einen gegebenen Prüfling und eine gegebene Spezifikation wird nach dieser Definition vollständig durch die Angabe des Testdatums beschrieben. Im Folgenden wird daher das Testdatum $t \in I$ als Bezeichner für den Testfall verwendet.

Aus den Definitionen für die Korrektheit und für das Bestehen von dynamischen Testfällen folgt, dass Prüflinge $p \in \Pi_{I, O}$, die korrekt bezüglich einer Spezifikation $s \in \Sigma_{I, O}$ sind, jeden beliebigen dynamischen Testfall bestehen:

$$(p, s) \in corr_{I, O} \Rightarrow \forall(t \in I)\ ((p(t) \neq \perp) \wedge ((t, p_{out}(t), p_{perf}(t)) \in s)) \quad \text{(Definition 3-5)}$$
$$\Rightarrow \forall(t \in I)\ ((t, p, s) \in ok_{I, O}). \quad \text{(Definition 3-6)}$$

Auf Basis einzelner dynamischer Testfälle kann die Durchführung eines dynamischen Tests auf einer gegebenen Menge von Testdaten beschrieben werden, die im Folgenden als dynamischer Testlauf bezeichnet wird.

Definition 3-7: Dynamischer Testlauf, Bestehen dynamischer Testläufe

Im Folgenden bezeichnet $\wp^{fin}(I)$ die Menge aller nichtleeren, endlichen Teilmengen des Eingabebereichs I.

(a) Ein dynamischer Testlauf für den Prüfling $p \in \Pi_{I, O}$ bezüglich einer Spezifikation $s \in \Sigma_{I, O}$ besteht aus einer nichtleeren, endlichen Menge dynamischer Testfälle. Er wird durch die zugrunde gelegte Menge der Testdaten $T \in \wp^{fin}(I)$ beschrieben.

(b) Ein dynamischer Testlauf T gilt dann als bestanden, wenn alle dynamischen Testfälle $t \in T$ bestanden wurden. Das Bestehen von dynamischen Testläufen durch einen Prüfling bezüglich einer Spezifikation wird durch die dreistellige Relation $ok'_{I, O} \subseteq \wp^{fin}(I) \times \Pi_{I, O} \times \Sigma_{I, O}$ mit

$$\forall(T \in \wp^{fin}(I))\ \forall(p \in \Pi_{I, O})\ \forall(s \in \Sigma_{I, O})$$
$$(((T, p, s) \in ok'_{I, O}) :\Leftrightarrow (\forall(t \in T)\ ((t, p, s) \in ok_{I, O})))$$

beschrieben.

Dynamische Testläufe bestehen nach dieser Definition aus einer nichtleeren, endlichen Menge von dynamischen Testfällen. Diese mengenbasierte Definition impliziert, dass einzelne dynamische Testfälle während eines Testlaufs nicht wiederholt werden. Auch die Reihenfolge der Durchführung der

einzelnen Testläufe ist unerheblich. Dies ist sinnvoll, da in Abschnitt 3.1 mit Annahme 3-5 ein deterministisches Verhalten und mit Annahme 3-1 die Ein-Schritt-Verarbeitungssemantik des Prüflings vorausgesetzt wurde. Eine Wiederholung oder Änderung der Reihenfolge von dynamischen Testfällen für denselben Prüfling liefert daher keine zusätzlichen Informationen. Die Menge der dynamischen Testfälle eines dynamischen Testlaufs darf nach der obigen Definition nicht leer sein, um die Durchführung mindestens eines dynamischen Testfalles im Rahmen eines Testlaufs zu garantieren. Gleichzeitig muss die Menge endlich sein, um die Durchführbarkeit des Tests in endlicher Zeit zu gewährleisten und damit der Annahme 3-3 zu genügen. Die Endlichkeit der dynamischen Testläufe ist bei endlichem Eingabebereich I immer gewährleistet, da einzelne Testfälle nicht wiederholt werden. Beim Test technischer Systeme mit analogen Schnittstellen zur Umwelt kann dies nicht vorausgesetzt werden, weswegen die Endlichkeit in der Definition explizit gefordert wird. Ein dynamischer Testlauf eines vorgegebenen Prüflings bezüglich einer Spezifikation ist vollständig durch die Menge $T \subseteq I$ der Testdaten beschrieben, die im Folgenden als Bezeichner für den dynamischen Testlauf verwendet wird.

Dynamische Testläufe gelten immer dann als bestanden, wenn alle enthaltenen dynamischen Testfälle bestanden wurden. Auch für dynamische Testläufe folgt aus der Korrektheit des Prüflings bezüglich der Spezifikation das Bestehen jedes beliebigen Testlaufs:

$$
\begin{aligned}
(p, s) \in corr_{I, O} &\Rightarrow \forall(t \in I)\ ((p(t) \neq \perp) \wedge ((t, p_{out}(t), p_{perf}(t)) \in s)) && \text{(Definition 3-5)}\\
&\Rightarrow \forall(t \in I)\ ((t, p, s) \in ok_{I, O}) && \text{(Definition 3-6)}\\
&\Rightarrow \forall(T \in \wp^{fin}(I))\ \forall(t \in T)\ ((t, p, s) \in ok_{I, O}) && \text{(allg. Mengenlehre)}\\
&\Rightarrow \forall(T \in \wp^{fin}(I))\ ((T, p, s) \in ok'_{I, O}). && \text{(Definition 3-7)}
\end{aligned}
$$

Auf der Basis dieser Definitionen kann nun der Begriff des dynamischen Testverfahrens definiert werden. Eine leistungsfähige Definition muss zwei Sichtweisen vereinen: Einerseits muss die konstruktive Sicht eines dynamischen Testverfahrens unterstützt werden, die die Auswahl der Testdaten in den Vordergrund stellt. Andererseits ist der wertende Charakter eines dynamischen Testverfahrens zu erfassen, der eine Beurteilung der Eignung und Vollständigkeit durchgeführter dynamischer Testläufe leistet. Diese beiden Aspekte werden in der Literatur als Testdatenselektion beziehungsweise Testdatenadäquatheit bezeichnet. Die folgende, rein mengenorientierte Beschreibung lässt Spielraum für beide Interpretationen:

Definition 3-8: Dynamisches Testverfahren, Bestehen eines dynamischen Testverfahrens, Adäquatheit eines dynamischen Testlaufs

Im Folgenden bezeichnet $\wp(M)$ die Potenzmenge einer Menge M, also die Menge aller Teilmengen von M.

(a) Ein dynamisches Testverfahren X beschreibt eine Zuordnung

$$
\begin{aligned}
X: \quad & \Pi_{I, O} \times \Sigma_{I, O} && \rightarrow && \wp(\wp^{fin}(I))\\
& (p, s) && \mapsto && X(p, s),
\end{aligned}
$$

die einem beliebigen Prüfling $p \in \Pi_{I, O}$ und einer beliebigen Spezifikation $s \in \Sigma_{I, O}$ eine Menge dynamischer Testläufe $X(p, s) \in \wp(\wp^{fin}(I))$ zuordnet und gleichzeitig die Bedingung

$$
\begin{aligned}
&\forall(s \in \Sigma_{I, O})\ \forall(p \in \Pi_{I, O})\ \forall(T \in X(p, s))\ \forall(M \in \wp^{fin}(I))\\
&((M \supseteq T) \Rightarrow (M \in X(p, s))) \qquad \text{(Monotonie)}
\end{aligned}
$$

erfüllt.

(b) Im Folgenden bezeichnet $\Xi_{I, O} := \{ X : \Pi_{I, O} \times \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I)) \}$ die Menge aller dynamischen Testverfahren, die Prüflingen aus $\Pi_{I, O}$ und Spezifikationen aus $\Sigma_{I, O}$ eine Menge von dynamischen Testläufen zuordnet.

(c) Ein dynamisches Testverfahren $X \in \Xi_{I, O}$ gilt dann vom Prüfling $p \in \Pi_{I, O}$ und der Spezifikation $s \in \Sigma_{I, O}$ als bestanden, wenn es ein mindestens einen dynamischen Testlauf $T \in X(p, s)$ gibt, der bestanden wird. Das Bestehen von dynamischen Testverfahren durch einen Prüfling bezüglich einer Spezifikation wird daher durch die dreistellige Relation $ok''_{I, O} \subseteq \Xi_{I, O} \times \Pi_{I, O} \times \Sigma_{I, O}$ mit

$$\forall(X \in \Xi_{I, O}) \, \forall(p \in \Pi_{I, O}) \, \forall(s \in \Sigma_{I, O})$$
$$((X, p, s) \in ok''_{I, O}) :\Leftrightarrow (\exists(T \in X(p, s)) \, ((T, p, s) \in ok'_{I, O}))$$

beschrieben.

(d) Alle dynamischen Testläufe, die durch das Verfahren einem Prüfling und einer Spezifikation zugeordnet werden, werden als adäquat für Prüfling und Spezifikation bezüglich des Verfahrens bezeichnet.

Der so definierte Begriff des dynamischen Testverfahrens beschreibt das gängige Prinzip im dynamischen Test: Für einen Prüfling und eine Spezifikation wird keine feste Teilmenge des Eingabebereichs zur Überprüfung vorgeschrieben, sondern eine ganze Menge von nichtleeren, endlichen Teilmengen als mögliche dynamische Testläufe angeboten. Jede dieser Mengen kann als dynamischer Testlauf verwendet werden; sie sind im Hinblick auf das dynamische Testverfahren gleichwertig.

Die in Definition 3-8 (a) für dynamische Testverfahren geforderte Monotonie verlangt, dass zu jedem dynamischen Testlauf auch alle endlichen Obermengen als dynamische Testläufe akzeptiert werden. Dies entspricht dem in der Praxis verbreiteten Vorgehen bei der Testdatenselektion und der Bewertung der Testdatenadäquatheit, das die Hinzunahme beliebiger dynamischer Testfälle zu einem dynamischen Testlauf erlaubt, ohne dass die Adäquatheit des dynamischen Testlaufs verloren gehen kann. Diese Bedingung entspricht der in [Wey86] geforderten Monotonie für Adäquatheitskriterien. Grundsätzlich ist es auch denkbar, auf die Forderung der Monotonie zu verzichten; dies entspricht jedoch nicht dem gängigen Verständnis und Vorgehen in der Praxis.

Der konstruktive Aspekt eines dynamischen Testverfahrens, der die Selektion der dynamischen Testfälle beschreibt, schlägt sich in der obigen Definition in der Zuordnung der dynamischen Testläufe zu Prüfling und Spezifikation nieder. Hierbei ist zu beachten, dass für die Zuordnung einer Menge von Testläufen zu einem gegebenen Prüfling oder einer Spezifikation auch weitere Parameter angegeben werden können. Dies können zum Beispiel Fehlerklassen sein, deren Abwesenheit durch das dynamische Testverfahren nachgewiesen wird. Auch die Angabe von Überdeckungszielen in Form eines Parameters ist denkbar. Für die funktionsorientierten Testverfahren, die auf einer informalen Spezifikation operieren, ist die Beschreibung in Form einer Zuordnung der dynamischen Testläufe zu Prüfling und Spezifikation nicht unproblematisch. Wie bereits in Kapitel 1 beschrieben, ist für die Bestimmung der Testdaten zu einem funktionsorientierten Testverfahren oft zunächst eine Aufarbeitung der informal in der Spezifikation beschriebenen Informationen durch den Tester zu leisten. Diese Informationen können dann als zusätzliche Parameter in der Verfahrensbeschreibung angegeben werden. Hier wird deutlich, dass kreative Leistung und systematisches Verfahren bei den funktionsorientierten Testverfahren eng verwoben sind.

Der analytische Aspekt der Adäquatheitsprüfung von dynamischen Testläufen wird durch die mengenorientierte Beschreibung gestützt: So werden die zum dynamischen Testverfahren gehörenden Testläufe als adäquat bewertet, alle anderen nicht. Die Adäquatheit bezüglich eines Testverfahrens X(p, s) kann beispielsweise beschrieben werden durch die Indikatorfunktion $1_{X(p, s)}$ über der Menge der möglichen Testläufe $\wp^{fin}(I)$. Sie ordnet den durch X akzeptierten Testläufen den Wert 1 zu, allen anderen Testläufen hingegen den Wert 0.

Die Menge der durch das Testverfahren vorgeschlagenen beziehungsweise akzeptierten dynamischen Testläufe kann – im Gegensatz zu den dynamischen Testläufen selbst – leer oder unendlich sein. Dies ist vor dem Hintergrund des praktischen Testgeschehens notwendig: So kann kein Testlauf durch das dynamische Testverfahren der Zweigabdeckung akzeptiert werden, wenn der vorgelegte Prüfling p nichtausführbare Zweige enthält. In diesem Fall ordnet das zugehörige dynamische Testverfahren X dem Prüfling und seiner Spezifikation die leere Menge $\varnothing$ zu. Weiterhin ist es möglich, dass ein dynamisches Testverfahren auf einem Prüfling oder einer Spezifikation nicht sinnvoll operieren kann. So kann ein Verfahren zur Prüfung von Syntaxgraphen aus einer in Fließtext gehaltenen Spezifikation keine Vorgaben für den dynamischen Test ableiten. In diesem Fall werden keine Forderungen an die dynamischen Testläufe gestellt, beliebige dynamische Testläufe werden als adäquat akzeptiert. Unendlich viele akzeptierte dynamische Testläufe können durch ein dynamisches Testverfahren nur dann vorgeschlagen werden, wenn der Eingabebereich I von Prüfling und Spezifikation eine unendliche Mächtigkeit hat. Dies ist für reine Softwareprogramme aufgrund der endlichen Zahlendarstellung im Rechner nicht relevant, für die Beschreibung des Tests technischer Systeme mit kontinuierlichen Schnittstellen zur Außenwelt jedoch denkbar.

Die Relation $ok''_{I, O}$ des Bestehens eines dynamischen Testverfahrens wird in Definition 3-8 bewusst umfangreich definiert. So gilt ein dynamisches Testverfahren bereits dann als bestanden, wenn nur einer der möglichen dynamischen Testläufe das Verfahren besteht. Dies ermöglicht eine kritische Beurteilung eines dynamischen Testverfahrens im Hinblick auf die Möglichkeit des Bestehens eines dynamischen Testlaufs durch einen bezüglich seiner Spezifikation nicht korrekten Prüfling. Würde eine alternative Beschreibung des Bestehens eines dynamischen Testverfahrens das Bestehen eines jeden zugehörigen dynamischen Testlaufs fordern, so würde dies in Verbindung mit der Monotonie einem Korrektheitsnachweis gleichkommen. Eine solche Definition des Bestehens eines dynamischen Testverfahrens ist zwar theoretisch möglich, praktisch aber unbrauchbar: Kein dynamisches Testverfahren könnte durch einen bezüglich seiner Spezifikation nicht korrekten Prüfling bestanden werden. Die Umkehrung dieses Sachverhaltes gilt jedoch auch für die hier getroffene Definition des Bestehens eines dynamischen Testverfahrens: Ist ein Prüfling hinsichtlich seiner Spezifikation korrekt, so folgt daraus das Bestehen eines jeden dynamischen Testverfahrens. Der Nachweis hierfür ergibt sich direkt aus den Definitionen und kann analog zu den Nachweisen des Bestehens beliebiger dynamischer Testfälle und Testläufe durch einen korrekten Prüfling geführt werden.

Für die Praxis ist es eine wünschenswerte Eigenschaft eines dynamischen Testverfahrens, wenn zu jedem Prüfling und jeder Spezifikation mindestens ein adäquater dynamischer Testlauf existiert. Dies wird in Anlehnung an [FraWey88] mit dem im Folgenden definierten Prädikat der Anwendbarkeit ausgedrückt.

Definition 3-9: Anwendbarkeit eines dynamischen Testverfahrens

Ein dynamisches Testverfahren $X \in \Xi_{I, O}$ wird als anwendbar bezeichnet, wenn sichergestellt ist, dass es für jeden Prüfling $p \in \Pi_{I, O}$ und jede Spezifikation $s \in \Sigma_{I, O}$ mindestens einen adäquaten dynamischen Testlauf gibt. Die Anwendbarkeit kann als einstelliges Prädikat

$$\forall(X \in \Xi_{I, O})\ (\text{Anwendbar}(X) :\Leftrightarrow (\forall(p \in \Pi_{I, O})\ \forall(s \in \Sigma_{I, O})\ (X(p, s) \neq \varnothing)))$$

über $\Xi_{I, O}$ beschrieben werden.

In [FraWey88] wird insbesondere die Anwendbarkeit datenflussorientierter Testverfahren für beliebige Prüflinge diskutiert. Zu beachten bei der Verwendung dieser Definition ist, dass die Prüfung der Anwendbarkeit beliebiger dynamischer Testverfahren ein unentscheidbares Problem ist, dass es also keinen Algorithmus gibt, der für ein beliebiges Testverfahren entscheidet, ob es anwendbar ist oder nicht [FraWey88].

Als einfaches Beispiel für ein dynamisches Testverfahren soll an dieser Stelle das vollständige Testen beschrieben werden, dass in Anlehnung an die englische Bezeichnung „exhaustive testing“ auch als ausschöpfendes Testen bezeichnet werden kann. Beim vollständigen Test wird unabhängig vom Prüfling $p \in \Pi_{I, O}$ und der Spezifikation $s \in \Sigma_{I, O}$ der gesamte Eingabebereich I im dynamischen Test überprüft. Dieser darf gemäß Definition 3-7 nur endliche Mächtigkeit besitzen, damit Annahme 3-3 erfüllt und die Durchführbarkeit des dynamischen Tests in endlicher Zeit sichergestellt ist.

Verfahren 3-1: Vollständiges Testen

(a) Voraussetzung:

Der Eingabebereich I besitzt endliche Mächtigkeit.

(b) Verfahrensbeschreibung:

Der vollständige dynamische Test wird beschrieben durch

$$\begin{array}{lllll} X^{vollst}: & \Pi_{I, O} \times \Sigma_{I, O} & \rightarrow & \wp(\wp^{fin}(I)) \\ & (p, s) & \mapsto & \{ I \}. \end{array}$$

Die Verfahrensbeschreibung zeigt, dass das vollständige Testen bei endlichem Eingabebereich immer anwendbar für beliebige Prüflinge und Spezifikationen im Sinne der Definition 3-9 ist. Die praktische Umsetzung des vollständigen Tests ist jedoch sehr beschränkt: Selbst wenn für reine Softwaresysteme aufgrund der endlichen Darstellung der Daten im Rechner von einem endlichen Eingabebereich ausgegangen werden kann, ist die Anzahl der im vollständigen Test zu prüfenden Eingabedaten viel zu groß. Zur exemplarischen Anwendung der oben gegebenen Definitionen soll hier jedoch die bekannte Eigenschaft gezeigt werden, dass ein bestandener vollständiger Test einen Nachweis für die Korrektheit des Prüflings bezüglich der Spezifikation bedeutet:

$$((X^{vollst}, p, s) \in ok''_{I, O}) \Rightarrow \quad \text{(Definition 3-8)}$$

$$((\exists T \in X^{vollst})\ ((T, p, s) \in ok'_{I, O})) \Rightarrow \quad \text{(Verfahren 3-1)}$$

$$((I, p, s) \in ok'_{I, O}) \Rightarrow \quad \text{(Definition 3-7)}$$

$$(\forall t \in I)\ ((t, p, s) \in ok_{I, O}) \Rightarrow \quad \text{(Definition 3-6)}$$

$$(\forall t \in I)\ ((p(t) \neq \perp) \wedge ((t, p_{out}(t), p_{perf}(t)) \in s)) \Rightarrow \quad \text{(Definition 3-5)}$$

$((p, s) \in corr_{I, O})$.

Aus der Beweisführung wird deutlich, dass der Grund für die Aussagekraft des vollständigen Tests in der Vollständigkeit der Prüfung des Eingabebereichs liegt. Diese Eigenschaft wird bei allen anderen Testverfahren verletzt, da dort im Allgemeinen nur die Überprüfung von echten Teilmengen des Eingabebereichs gefordert wird.

Zum Vergleich dynamischer Testverfahren ist die Relation der Subsumption in der Literatur häufig diskutiert worden [ClaHasRic82, FraWey88]. Diese Relation vergleicht die dynamischen Testverfahren im Hinblick auf die als adäquat akzeptierten dynamischen Testläufe: Falls für einen beliebigen Prüfling und eine beliebige Spezifikation die Adäquatheit eines dynamischen Testlaufs bezüglich eines dynamischen Testverfahrens immer auch seine Adäquatheit bezüglich eines zweiten impliziert, so sagt man, dass das erste dynamische Testverfahren das zweite subsumiert. Die mengenorientierte Gestaltung der Definition 3-8 ermöglicht eine einfache Beschreibung dieser Relation zwischen dynamischen Testverfahren.

Definition 3-10: Subsumptionsrelation zwischen dynamischen Testverfahren

Für zwei dynamische Testverfahren aus $\Xi_{I, O}$ wird die Relation

$\forall(X_1 \in \Xi_{I, O}) \forall(X_2 \in \Xi_{I, O})$
$((X_1 \text{ subsumiert } X_2) :\Leftrightarrow (\forall(s \in \Sigma_{I, O}) \forall(p \in \Pi_{I, O}) (X_1(p, s) \subseteq X_2(p, s))))$

über $\Xi_{I, O} \times \Xi_{I, O}$ als Subsumption bezeichnet. Die Subsumption ist identitiv, reflexiv und transitiv, so dass ($\Xi_{I, O}$, subsumiert) eine geordnete Menge bildet.

Identitivität, Reflexivität und Transitivität der Subsumptionsrelation folgen unmittelbar aus der Identitivität, Reflexivität und Transitivität der Relation $\subseteq$. Hervorzuheben ist, dass die Subsumptionsrelation eine Ordnung auf der Menge der dynamischen Testverfahren $\Xi_{I, O}$ definiert. Diese ist nicht linear, so dass die Vergleichbarkeit zweier dynamischer Testverfahren nicht immer gegeben ist. Als Beispiel für die Vergleichbarkeit im Sinne der Subsumption kann der vollständige Test gemäß Verfahren 3-1 betrachtet werden, der jedes im Sinne von Definition 3-9 anwendbare Testverfahren subsumiert.

Oftmals werden einzelne dynamische Testverfahren miteinander kombiniert. Eine solche Kombination von Testverfahren kann bei der Erzeugung von Testdaten genutzt werden, aber auch zu ihrer Bewertung herangezogen werden. Nahe liegendes Beispiel für eine kombinierte Strategie ist die Bewertung der Adäquatheit von Testdaten im Hinblick auf die erreichte strukturelle und funktionale Überdeckung von Prüfling und Spezifikation. Bei der praktischen Umsetzung eines solchen kombinierten Testverfahrens werden dynamische Testläufe, die im Hinblick auf das eine Verfahren gewählt wurden, durch einzelne dynamische Testfälle oder Testläufe bezüglich des anderen Verfahrens ergänzt. Die mengenbasierten Definitionen dieser Begriffe ermöglichen es, dieses Vorgehen als Vereinigung von dynamischen Testläufen zu beschreiben.

Definition 3-11: Kombination dynamischer Testläufe

Die Vereinigung zweier dynamischer Testläufe $T_1 \in \wp^{fin}(I)$ und $T_2 \in \wp^{fin}(I)$ über demselben Eingabebereich I und Ausgabebereich O bildet wegen $T_1 \cup T_2 \in \wp^{fin}(I)$ einen dynamischen Testlauf, der als Kombination der Testläufe T_1 und T_2 bezeichnet wird.

Es ist intuitiv verständlich und anhand von Definition 3-7 leicht nachzuweisen, dass ein Prüfling p bezüglich einer Spezifikation s genau dann die Kombination zweier dynamischer Testläufe T_1 und T_2 besteht, wenn p bezüglich s sowohl T_1 als auch T_2 besteht.

Bei der Definition der Kombination zweier dynamischer Testverfahren muss berücksichtigt werden, dass dynamische Testläufe, die lediglich eines der beiden Verfahren erfüllen, das andere aber nicht, das kombinierte Verfahren nicht erfüllen dürfen. Akzeptiert werden nur dynamische Testläufe, die von beiden Verfahren akzeptiert werden.

Definition 3-12: Kombination dynamischer Testverfahren

Gegeben seien zwei dynamische Testverfahren $X_1 \in \Xi_{I, O}$ und $X_2 \in \Xi_{I, O}$. Dann wird die Zuordnung

$$\begin{array}{lccl} (X_1 \bullet X_2): & \Pi_{I, O} \times \Sigma_{I, O} & \rightarrow & \wp(\wp^{fin}(I)) \\ & (p, s) & \mapsto & X_1(p, s) \cap X_2(p, s) \end{array}$$

als Kombination der dynamischen Testverfahren X_1 und X_2 bezeichnet.

Satz:

(a) $X_1 \bullet X_2$ ist ein dynamisches Testverfahren über I und O.

(b) Die Kombination dynamischer Testverfahren ist kommutativ und assoziativ, d. h. es gilt

$$X_1 \bullet X_2 = X_2 \bullet X_1, \qquad \text{(Kommutativität)}$$
$$(X_1 \bullet X_2) \bullet X_3 = X_1 \bullet (X_2 \bullet X_3). \qquad \text{(Assoziativität)}$$

(c) Die Kombination endlich vieler dynamischer Testverfahren $X_1, X_2, \ldots, X_n$ ($n \in \aleph$) kann durch schrittweise Kombination der einzelnen Testverfahren X_i in beliebiger Reihenfolge bestimmt werden. Es ergibt sich ein dynamisches Testverfahren, dass mit

$$\circledcirc_{(i=1, \ldots, n)} X_i := X_1 \bullet X_2 \bullet \ldots \bullet X_n$$

bezeichnet wird.

Beweis:

(a) Zum Nachweis, dass die Kombination $X_1 \bullet X_2$ ein dynamisches Testverfahren ist, muss die Monotonie gezeigt werden. Dazu sei für beliebige $p \in \Pi_{I, O}$ und $s \in \Sigma_{I, O}$ ein dynamischer Testlauf $T \in (X_1 \bullet X_2)(p, s)$ gewählt. Dieser ist nach Definition 3-12 sowohl in $X_1(p, s)$ als auch in $X_2(p, s)$ enthalten. Für jedes $U \in \wp^{fin}(I)$ mit $U \supseteq T$ gilt wegen der Monotonie der dynamischen Testverfahren X_1 und X_2 ebenfalls $U \in X_1(p, s)$ und $U \in X_2(p, s)$ und damit $U \in X(p, s)$.

(b) Die Kommutativität und Assoziativität der Kombination dynamischer Testverfahren folgt direkt aus der Kommutativität und Assoziativität der Schnittmengenbildung in Definition 3-12.

(c) Der Nachweis, dass $\circledcirc_{(i=1, \ldots, n)} X_i$ ein dynamisches Testverfahren ist, folgt einfach durch Induktion aus (a) unter Verwendung von (b). ■

Als Beispiel sei ein dynamisches Testverfahren betrachtet, das aus der Kombination eines dynamischen Verfahrens zur Strukturüberdeckung und einem weiteren dynamischen Verfahren zur funktionalen Überdeckung entstanden ist. Dieses kombinierte Verfahren darf einen dynamischen Testlauf zur Überdeckung der Struktur, der jedoch spezifizierte funktionale Aspekte nicht prüft, nicht als adäquat akzeptieren. Akzeptiert werden jedoch beispielsweise Testläufe, die aus der Vereinigung von Testläufen zur Erfüllung des strukturorientierten beziehungsweise des funktionsorientierten Verfahrens entstanden sind. Eine solche Vereinigung, die möglicherweise zu einem sehr umfangreichen dynamischen Testlauf führt, ist jedoch oft nicht notwendig. So prüft ein Testlauf zur Überdeckung der Struktur im Allgemeinen bereits wesentliche funktionale Aspekte. Zur Erfüllung des kombinierten Verfahrens müssen nur einzelne dynamische Testläufe hinzugenommen werden, die die bislang noch nicht überprüften funktionalen Aspekte prüfen. Dieser Zusammenhang spiegelt sich in den obigen Definitionen wider.

3.2.3 Die Auswertung des dynamischen Tests

Die im vorigen Abschnitt eingeführten Begriffe des dynamischen Testfalls, dynamischen Testlaufs und dynamischen Testverfahrens prüfen durch die zugehörige Bestehensrelation die exakte Übereinstimmung des Prüflings mit der Spezifikation auf den Testdaten. Eine entsprechende Auswertung der Testdaten ist in der Praxis nicht immer möglich und auch nicht immer in voller Schärfe gewünscht. Der folgende Abschnitt ergänzt daher Definitionen zur Auswertung des dynamischen Tests, die durch eine gezielte Modifikation der Bestehensrelation Raum für eine Anpassung des beschriebenen Vorgehens an die Gegebenheiten der Praxis lassen.

In Definition 3-6 wird die Spezifikation zur Prüfung des Testergebnisses verwendet. Die Spezifikation wird damit als ein Orakel im Sinne der Definition von Howden [How86] aufgefasst. Nach Howden wird eine beliebiges Programm, ein beliebiger Prozess oder ein beliebiger Datensatz als Orakel bezeichnet, der den zu erwartenden Ausgang eines beliebigen dynamischen Testfalls – unabhängig von seiner Durchführung – festlegt. Ein Orakel kann also, ebenso wie die Spezifikation selbst, als eine Relation zwischen Eingabe-, Ausgabe- und Performanzbereich verstanden werden. Eine exakte Übereinstimmung des Testorakels mit der Spezifikation wird nicht gefordert, da Testorakel in realen Testprozessen häufig fehlerbehaftet oder auf Teilaspekte eingeschränkt sind, wie beispielsweise in [BerStr96] ausgeführt ist.

Definition 3-13: Testorakel, Bestehen eines dynamischen Testfalls bezüglich eines Testorakels, suboptimales Testorakel

(a) Ein Testorakel wird beschrieben durch eine dreistellige Relation über Eingabe- und Ausgabebereich und den positiven reellen Zahlen or $\subseteq I \times O \times \Re^+$, die der Bedingung

$$\forall(i \in I)\ \exists(o \in O)\ \exists(r \in \Re^+)\ (i, o, r) \in s \qquad \text{(Vollständigkeit)}$$

genügt. Die Menge aller Testorakel über $I \times O \times \Re^+$ wird im Folgenden mit $\Omega_{I, O}$ bezeichnet.

(b) Ein dynamischer Testfall wird von einem Testorakel akzeptiert, wenn das Ergebnis der Testdurchführung dem Testorakel entspricht. Die Akzeptanz von dynamischen Testfällen eines Prüflings bezüglich eines Testorakels wird somit durch die dreistellige Relation $or\_ok_{I, O} \subseteq I \times \Pi_{I, O} \times \Omega_{I, O}$ mit

$$\forall(t \in I)\ \forall(p \in \Pi_{I, O})\ \forall(or \in \Omega_{I, O}):$$
$$((t, p, or) \in or\_ok_{I, O}) :\Leftrightarrow ((p(t) \neq \perp) \wedge ((t, p_{out}(t), p_{perf}(t)) \in or)$$

beschrieben.

(c) Ein Testorakel $or \in \Omega_{I, O}$ wird als optimal bezüglich einer Spezifikation $s \in \Sigma_{I, O}$ bezeichnet, falls

$$or = s$$

gilt. Andernfalls wird es als suboptimal bezeichnet.

Formal unterscheiden sich die Definition 3-2 (a) und Definition 3-13 (a) nicht, so dass eine Spezifikation immer als Testorakel verwendet werden kann. Fallen Spezifikation und Testorakel zusammen, so stimmen die Relationen $or\_ok_{I, O}$ und $ok_{I, O}$ überein. Definition 3-6 (b) ergibt sich dann als Sonderfall von Definition 3-13 (b). Auch hier stellt die geforderte Vollständigkeit eines Testorakels keine echte Einschränkung dar, da den Eingabedaten, bezüglich derer das Testorakel keine explizite Prüfung durchführen soll, der gesamte Ausgabebereich und ein allgemein akzeptabler Performanzbereich zugeordnet werden kann.

Um die Übereinstimmung eines Testorakels mit einer Spezifikation zu betrachten, kann die Mächtigkeit der Menge der Eingabedaten ermittelt werden, bezüglich derer Spezifikation und Testorakel voneinander abweichen. Diese kann als Unschärfe des Testorakels oder als Abstand des Testorakels von der Spezifikation bezeichnet werden. Zusammenhänge zwischen einem funktionalen und einem strukturellen Abstandsbegriff zwischen Spezifikationen werden beispielsweise in [JilDesMil01] diskutiert und können auf das Problem der Bewertung eines suboptimalen Orakels übertragen werden. Interessant für die Praxis ist der Fall, dass das Testorakel eine Obermenge der Spezifikation darstellt. Dies kann beispielsweise gegeben sein, wenn Prüfinstanzen lediglich die Einhaltung kritischer Bedingungen überwachen.

Auch bei der Beurteilung des Ergebnisses von dynamischen Testläufen und dynamischen kann ein Testorakel verwendet werden. So kann die Relation $or\_ok_{I, O}$ analog zur Bestehensrelation dynamischer Testfälle in Definition 3-7 (b) zur Relation $or\_ok'_{I, O}$ über $\wp^{fin}(I) \times \Pi_{I, O} \times \Omega_{I, O}$ erweitert werden. Diese Relation beschreibt genau die dynamischen Testläufe, deren dynamische Testfälle bezüglich des Testorakels bestanden werden. Analog zu Definition 3-8 (c) kann daraus die Relation $or\_ok''_{I, O}$ abgeleitet werden, die das Bestehen dynamischer Testverfahren im Hinblick auf ein Testorakel bestimmt.

Nach Definition 3-7 gilt eine dynamischer Testlauf genau dann als bestanden, wenn für alle enthaltenen dynamischen Testfälle die Übereinstimmung des Prüflings mit der Spezifikation beziehungsweise mit einem Testorakel nachgewiesen werden kann. In der Praxis führen jedoch manchmal deutlich schwächere Kriterien zur Akzeptanz des Prüflings, die gewisse Abweichungen in den Reaktionen des Prüflings von Spezifikation oder Testorakel tolerieren. Beispiel ist die Akzeptanz und Freigabe eines Prüflings, der eine vorgegebene Anzahl weniger relevanter Abweichungen im Test nicht überschritten hat, oder auch die Freigabe eines Prüflings, dessen stochastischer Test zwar Abweichungen von der Spezifikation gezeigt hat, der aber einen Hypothesentest für den Nachweis seiner Zielzuverlässigkeit bestanden hat. Aus diesem Grund wird an dieser Stelle Raum gegeben für eine Definition eines adaptierbaren Kriteriums, das als Akzeptanz dynamischer Testläufe bezeichnet wird. Für die Akzeptanz wird lediglich gefordert, dass das Bestehen eines dynamischen Testlaufs entsprechend Definition 3-7 seine Akzeptanz impliziert.

Definition 3-14: Akzeptanz dynamischer Testläufe

Eine beliebige dreistellige Relation $ak'_{I, O} \subseteq \wp^{fin}(I) \times \Pi_{I, O} \times \Sigma_{I, O}$ wird als Akzeptanz bezeichnet, wenn sie die Bedingung

$ak'_{I, O} \supseteq ok'_{I, O}$

erfüllt.

Die Akzeptanz dynamischer Testläufe kann analog zu Definition 3-8 zu einer Akzeptanzrelation $ak''_{I, O}$ für dynamische Testverfahren erweitert werden. Diese Relation ist wegen der in Definition 3-14 verankerten Akzeptanz bestandener dynamischer Testläufe eine Obermenge zur Bestehensrelation $ok''_{I, O}$. Ein dynamisches Testverfahren, das gemäß $ok''_{I, O}$ bestanden wird, wird daher auch immer durch $ak''_{I, O}$ akzeptiert.

Der erweiterte Akzeptanzbegriff für dynamische Testläufe und –verfahren ist nicht zu verwechseln mit der nachträglichen Akzeptanz eines Testlaufs bezüglich eines abgeschwächten Adäquatheitskriteriums, wie er beispielsweise in der Praxis bei der Akzeptanz eines Testlaufs mit 80% Zweigüberdeckung anstelle von ursprünglich geforderten 90% zu beobachten sein könnte. Eine solche Erweiterung kann nur durch Modifikation des zur Begutachtung der Testfälle herangezogenen Adäquatheitskriteriums und damit nur im Hinblick auf das dynamische Testverfahren formuliert werden.

Es ist denkbar, dass dynamische Testläufe und Testverfahren, die durch ein Testorakel beurteilt wurden, mit einem Akzeptanzkriterium gemäß Definition 3-14 ausgewertet werden. Hierbei sind Abweichungen von einem exakten, an der Erfüllung der Spezifikation orientierten Verfahren sowohl bei der Beurteilung der Ergebnisse einzelner Testfälle durch das Testorakel als auch bei der Akzeptanz ganzer dynamischer Testläufe möglich. Der Vergleich dieses im Hinblick auf die Praxis realistischen Auswertungsmodells mit der an Übereinstimmung mit der Spezifikation orientierten Definition 3-7 macht deutlich, dass in einem realen Testprozess die tatsächliche Aussagekraft des Testergebnisses durch vielfältige Einflüsse geschwächt werden kann. Die hier gegebenen Definitionen des suboptimalen Testorakels, der erweiterten Akzeptanzrelation und der Anpassung des Adäquatheitskriteriums helfen, diese Einflüsse analytisch erfassbar zu machen.

3.3 Bewertung des Modells und Diskussion von Erweiterungsmöglichkeiten

Das im vorigen Abschnitt eingeführte Modell zum Vorgehen im dynamischen Test beschreibt Prüfling und Spezifikation vor dem Hintergrund ihres Ein- und Ausgabebereichs und definiert die zentralen Begriffe des dynamischen Testfalls, des dynamischen Testlaufs und des dynamischen Testverfahrens. Diese gestatten mit Hilfe der Relation des Bestehens die Analyse der Übereinstimmung von Prüfling und Spezifikation über ihrem Eingabebereich. Das Modell ermöglicht die mengen- und zuordnungsbasierte Beschreibung und Analyse von dynamischen Testverfahren im Hinblick auf Prüfling und Spezifikation und kann um weitere Parameter ergänzt werden. Aspekte einer gegenüber der Spezifikation suboptimalen Auswertung der dynamischen Testfälle werden durch ein Testorakel modelliert, Raum für Toleranzen bei der Auswertung dynamischer Testläufe gibt eine gestaltbare Akzeptanzrelation, die Abweichungen von der Spezifikation explizit zulassen kann.

Darüber hinaus werden an dem Modell spezielle Eigenschaften des dynamischen Tests deutlich. Wie in [FraEA98] diskutiert, werden durch den dynamischen Test die Ziele verfolgt, Fehler im Prüfling zu

finden und Vertrauen in seine Funktionsweise aufzubauen. Diese Ziele stehen im Modell der punktuellen Analyse der Übereinstimmung von Prüfling und Spezifikation über dem Eingabebereich gegenüber. Es wird deutlich, dass die Aussagekraft des dynamischen Tests auf die Funktionsweise des Prüflings zu den durchgeführten Testfälle beschränkt ist: Keine noch so große Zahl „bestandener" Testfälle kann den Erfolg eines weiteren, noch nicht durchgeführten Testfalls garantieren. Dies ist darin begründet, dass die Auswertung einen stichprobenhaften Charakter hat und keinerlei Bedingungen an den Prüfling gestellt werden, die einen Rückschluss auf seine Ausprägung für nicht getestete Eingabedaten gestatten [Ham02]. Hieraus wird deutlich, dass mit Hilfe eines unvollständigen dynamischen Tests ein Korrektheitsnachweis nicht erbracht werden kann. Weiterhin wird durch die Beschreibung des suboptimalen Testorakels und der erweiterten Akzeptanzrelation deutlich, wie durch Ungenauigkeiten bei der Auswertung die Effektivität eines dynamischen Testverfahrens bei der Fehlerfindung und seine Aussagekraft für den Vertrauensaufbau geschwächt werden können. Auf dieser Basis kann abgeleitet werden, welche Einflüsse bei einer Quantifizierung dieser Effekte berücksichtigt werden müssen.

Damit liefert das Modell eine geeignete Grundlage für eine allgemeine Untersuchung des dynamischen Tests und für die Analyse spezieller dynamischer Testverfahren. In Kapitel 4 werden daher die hier eingeführten Definitionen als Grundlage für eine Untersuchung des funktionsorientierten Tests verwendet. An dieser Stelle muss jedoch auch auf Beschränkungen hingewiesen werden, die aus den Annahmen zur Modellbildung resultieren. Diese können eine Anwendung in der Praxis erschweren und die Übertragbarkeit auf erweiterte Fragestellungen einschränken. Sie werden in den folgenden Abschnitten aufgezeigt und untersucht; ein knapper Ausblick auf mögliche Erweiterungen des Modells wird gegeben.

3.3.1 Abschwächung der Annahmen zum Testprozess

Die in Abschnitt 3.1 getroffenen Annahmen wurden als Grundlage der Definitionen des Abschnitts 3.2 eingeführt und haben es ermöglicht, den dynamischen Test als einen wohldefinierten, endlichen Prozess zu beschreiben. Bei ihrer Einführung wurde darauf hingewiesen, dass sie idealisiert sind und Einschränkungen der Praxis darstellen. Soll nun das in den vorangegangenen Abschnitten eingeführte Modell bei der Analyse eines realen Testprozesses verwendet werden, muss zunächst die Erfüllung der Annahmen untersucht werden, bevor die formal definierten Begriffe im realen Kontext interpretiert werden. Falls Annahmen nicht erfüllt sind, kann eine Erweiterung des Modells erwogen werden. Die Auswirkungen einer Abschwächung der Annahmen auf die Durchführbarkeit und Entscheidbarkeit des dynamischen Tests werden im Folgenden anhand des Modells diskutiert.

Annahme 3-1 fordert die Ein-Schritt-Verarbeitungssemantik, die Basis der relationalen und zuordnungsbasierten Definitionen von Prüfling und Spezifikation ist. Kann diese Annahme nicht getroffen werden, so sind sämtliche Definitionen des vorangegangenen Abschnitts neu zu fassen. Die Flexibilität bei der Festlegung der Ein- und Ausgabebereiche führt jedoch dazu, dass die Annahme weniger einschränkt, als es zunächst scheint. So ist es möglich, Ein- und Ausgabebereich vektoriell oder als Signal über dem Kontinuum zu beschreiben. Die Definitionen werden hiervon nicht berührt. Dynamische Testfälle, Testläufe und Testverfahren beziehen sich in diesem Fall auf Daten des entsprechenden Darstellungsformats. Unproblematisch wird die Datenbestimmung immer dann sein, wenn Ein- und Ausgangsgrößen separat betrachtet werden können. Realisiert der Prüfling jedoch ein Regelsystem mit Rückkopplung zwischen Ein- und Ausgabedaten, ergeben sich Probleme im Hinblick auf die Plausibilität der Daten. So wird beispielsweise ein Klimagerät die Raumtemperatur auswerten und

gleichzeitig auch beeinflussen. Diese ist somit gleichermaßen Ein- und Ausgangsdatum. Hier sind nur wenige der denkbaren Temperaturkurven realistisch, die zudem von der Realisierung im Prüfling abhängen. Zu ihrer Auswahl und zur Prüfung ihrer Plausibilität kann in diesem Fall die in der Spezifikation beschriebene Relation verwendet werden. An diesen Überlegungen wird deutlich, dass das in diesem Kapitel entwickelte Modell speziell auf den dynamischen Test transformativer Systeme ausgerichtet ist und sich für den dynamischen Test reaktiver Systeme weniger eignet. Eine wichtige Ausnahme hiervon bilden die zustandsbasierten Systeme, die in der Praxis für die Realisierung von reaktiven Systemen verwendet werden. Ihr dynamischer Test kann im Rahmen des hier entwickelten Modells abgebildet werden, indem der Systemzustand als zusätzlicher Ein- und Ausgangsparameter aufgenommen wird [TurRob93, Voa99]. Eine solche Anpassung des Modells wird bei der Diskussion der zustandsbasierten Testverfahren in Abschnitt 4.4 beschrieben.

Annahme 3-2 fordert die Eindeutigkeit und Vollständigkeit der Spezifikation. Diese ist essentiell wichtig, um die Korrektheit von Prüflingen und Auswertung von Testergebnissen über dem gesamten Eingabebereich formulieren zu können. Kann in der Realität eine Vollständigkeit und Eindeutigkeit der Spezifikation nicht vorausgesetzt werden, so kann ein erweiterter Spezifikationsbegriff verwendet werden, der neben den tatsächlich spezifizierten auch die intendierten Erwartungen an das Verhalten des Prüflings umfasst. Ein solches Verständnis von Spezifikation wird beispielsweise in [Voa99] angeregt. Unsicherheiten, die daraus bei der Auswertung des Tests entstehen, können mit Hilfe des suboptimalen Testorakels modelliert werden. Die Auswirkungen auf die Aussagekraft des Testergebnisses können dann im Hinblick auf Abweichungen zwischen Spezifikation und Orakel vor dem Hintergrund der Auswahl beziehungsweise der statistischen Verteilung der Testdaten beurteilt werden.

Die Annahme 3-3 der Endlichkeit der Menge der Testdaten ist in realen Testprozessen im Allgemeinen erfüllt. Bei der Definition von dynamischen Testverfahren ist sie allerdings explizit zu betrachten. Eine Vernachlässigung dieser Annahme kann zur Forderung dynamischer Testläufe mit unendlicher Mächtigkeit führen und damit zu Problemen mit der Anwendbarkeit des dynamischen Testverfahrens, wie es beim Verfahren der Pfadabdeckung für Prüflinge mit einer endlichen Zahl von möglichen Schleifendurchläufen der Fall ist.

Das in Annahme 3-4 verankerte Performanzkriterium wird in realen Testprozessen im Allgemeinen durch die für einzelne Testfälle beschränkte Ausführungszeit erfüllt. Es kann beispielsweise einfach durch Angabe eines einheitlichen Maximalwerts für die Ausführungszeit des Prüflings erreicht werden. In Bereichen, in denen Performanzanforderungen eine wichtige Rolle spielen, kann die Performanzangabe zu einzelnen Testfällen ein geeignetes Mittel des Spezifikation sein; hier werden Performanzangaben jedoch häufig in Form des Durchsatzes angegeben. Eine – wenn auch nicht explizit angegebene – Beschränkung der Ausführungszeit wird dennoch in den meisten Testprozessen gegeben sein und ist essentiell wichtig, um die Durchführung des dynamischen Tests in endlicher Zeit sicherzustellen.

Die Annahme 3-5 fordert das deterministische Verhalten des Prüflings. Es kann sinnvoll sein, diese Annahme zu lockern und ein zufälliges Verhalten des Prüflings zuzulassen. Dies ist dann hilfreich, wenn die Reaktion des Prüflings von Gegebenheiten abhängt, die nicht genau beschrieben oder modelliert werden können. Als Beispiel hierfür sei der Zugriff auf gemeinsam genutzte Ressourcen durch den Prüfling und weitere Softwarekomponenten im selben System genannt, der einen Einfluss auf die Antwortzeiten des Prüflings haben kann. Die Definitionen des dynamischen Testlaufs und des dynamischen Testverfahrens sind für einen Prüfling mit nicht-deterministischem Verhalten anzupassen: So ist eine Wiederholung einzelner dynamischer Testfälle zuzulassen, da die mehrfache Durchführung

eines Testfalls zusätzliche Informationen erzeugt. Eine mengenorientierte Beschreibung ist hierfür ungeeignet und muss zu einer sequenz- oder verteilungsbasierten Definition erweitert werden. Nahe liegend ist in diesem Fall die stochastische Auswahl der Testdaten. Zur Modellierung des Prüflings entsprechend Definition 3-4 müssen in diesem Fall ebenfalls stochastische Techniken verwendet werden. So kann dieser beispielsweise durch Angabe einer Zufallsvariablen und ihrer Verteilung beschrieben werden. Aussagen des dynamischen Tests können unter solchen Bedingungen nur stochastischer Natur sein und beispielsweise in Form von Hypothesen, Momenten- und Verteilungsschätzungen angegeben werden. Dadurch ergeben sich erweiterte Möglichkeiten für die Auswertung der dynamischen Testläufe: Qualitätsvorgaben können in Form von Hypothesen formuliert und mit statistischen Tests auf den Ergebnissen des dynamischen Tests überprüft werden. Dieses Vorgehen kann bei einer Vorgabe statistischer Kenngrößen zur Qualität und ihrer Überprüfung sinnvoll eingesetzt werden.

3.3.2 Betrachtung wirtschaftlicher Aspekte des dynamischen Tests

Die Betrachtung wirtschaftlicher Aspekte ist bei der Auswahl dynamischer Testtechniken von großem praktischem Interesse. Das hier vorgestellte Modell kann für solche Betrachtungen erweitert werden: So ermöglicht es die Betrachtung der Kosten eines dynamischen Testverfahrens für einen gegebenen Prüfling und eine gegebene Spezifikation durch Analyse der Mächtigkeit der akzeptierten dynamischen Testläufe, ihres Minimalwerts und ihres Durchschnitts. Mittelwertbetrachtungen für spezielle Klassen von Prüflingen oder Spezifikationen sind denkbar. Weiterhin ist es denkbar, bei der Formulierung von Prüfstrategien die Auswahl von dynamischen Testläufen mit einer möglichst geringen Anzahl von dynamischen Testfällen zu fordern. Zur Betrachtung der potentiellen Fehlleistungskosten durch Abweichungen von der Spezifikation kann zu jedem Eingabedatum ein Wert angegeben werden, der die Kosten bei Abweichung von den spezifizierten Ausgaben angibt. Es können auch mehrere Werte für Abweichungen verschiedener Schwere angegeben werden. Zusätzlich kann der Ausgabebereich in Teilmengen unterteilt werden, über denen das Schadensausmaß quantifiziert wird. Weitere Möglichkeiten für die Angabe von Kostenfunktion werden in [WeiWey88] und [Gut95] angeregt. Die Gegenüberstellung der durch ein dynamisches Testverfahren verursachten Kosten mit der Anzahl der gefundenen Fehler und den dadurch vermiedenen Fehlleistungskosten ermöglichen die Entwicklung einer kostenoptimierten Teststrategie auf theoretischer Basis im Modell.

Grenzen dieser Betrachtung, etwa bei der Angabe realer Kenngrößen, entstehen durch der Unkenntnis der tatsächlichen relationalen Struktur der Spezifikation, des Verhaltens des Prüflings, der mit Fehlverhalten verbundenen Kosten und nicht zuletzt durch die Unmöglichkeit einer algorithmischen Beschreibung der Testverfahren, da die Zuordnung von dynamischen Testläufen zu beliebigen Prüflingen und Spezifikationen im Allgemeinen ein nicht berechenbares Problem darstellt. Letzteres kann einfach am Beispiel der Anweisungsüberdeckung nachvollzogen werden, bei dem sich die Ermittlung von Testdaten zur Überdeckung der finalen Endpunktes des Programms als äquivalent zum Halteproblem erweist. Eine analytische Betrachtung der Wirtschaftlichkeit wird damit eher theoretischer Natur sein, kann aber durch eine empirische Ermittlung der betrachteten Größen ergänzt werden.

Wie bereits in Abschnitt 3.3.1 bei der Diskussion einer Erweiterung von Annahme 3-5 angeregt, ist es weiterhin möglich, eine stochastische Beschreibung des Prüflings in Form einer Zufallsvariable zu geben, die der Unsicherheit über die tatsächlich realisierte Funktionalität Rechnung trägt. Vor diesem Hintergrund können statistische Verfahren zur Schätzung der oben diskutierten Kennwerte erarbeitet werden. Ein entsprechender Ansatz wurde bereits in [Nel73] bei der Entwicklung einer statistischen

Theorie zur Schätzung der Zuverlässigkeit von Software und der Ableitung der optimalen Stichprobengrößen über einer Partitionierung des Eingabebereichs verwendet, in [WeiWey88] im Hinblick auf die Betrachtung von Fehlleistungskosten erweitert und [Gut95] zu einer statistischen Theorie mit dem Ziel der Optimierung des Testprofils im Hinblick auf die Fehlleistungskosten ausgearbeitet.

3.3.3 Prüfung nicht-funktionaler Anforderungen

Das gewählte Modell beschreibt ein Vorgehen im dynamischen Test, das als Zielsetzung eine Prüfung der Umsetzung der spezifizierten Anforderungen im Prüfling verfolgt. Diese Anforderungen werden hier relational als Zuordnung von Daten des Ein- und Ausgabebereichs mit zusätzlicher Angabe einer Performanzanforderung in der Spezifikation beschrieben und haben damit eine stark funktionale Prägung. Nicht-funktionale Anforderungen, die über die Performanzangabe hinausgehen, können in der gegebenen Darstellung der Spezifikation nicht erfasst werden und finden daher im hier beschriebenen Modell zum dynamischen Test keinen Eingang. Sie müssen separat formuliert und geprüft werden. Beispiele für solche Anforderungen sind Sicherheits-, Zuverlässigkeits- oder Verfügbarkeitsanforderungen sowie Anforderungen an Reaktionszeiten unter Last und Anforderungen an einen zu leistenden Durchsatz.

Das hier beschriebene Modell berücksichtigt solche nicht-funktionalen Anforderungen nicht, bietet aber verschiedenen Möglichkeiten zur Erweiterung im Hinblick auf ihre Prüfung: So können Sicherheitsanforderungen beispielsweise durch eine zusätzliche Prüfinstanz in Form eines Testorakels geprüft werden, das die Einhaltung gewisser Sicherheitsvorgaben überwacht. Darüber hinaus können Zuverlässigkeits- und Verfügbarkeitsanforderungen mit einem beispielsweise am Nutzerprofil ausgerichteten stochastischen Test geprüft werden, für den lediglich die mengenbasierte Beschreibung des dynamischen Testlaufs zu einer verteilungsbasierten erweitert werden muss. Die Auswertung eines solchen Tests ist ebenfalls auf stochastischer Basis zu beschreiben. Zur Analyse des Verhaltens des Prüflings unter Last kann eine Zeitbetrachtung bei der Testdurchführung ergänzt werden. Wird zusätzlich eine sukzessive Weiterentwicklung des Prüflings durch Korrektur der gefundenen Fehler zugelassen, kann eine Zuverlässigkeitsanalyse mit Hilfe von Zuverlässigkeitswachstumsmodellen durchgeführt werden [MusIanOku90, Lyu95, ElbMäc02].

Bei der Prüfung von Sicherheits- und Zuverlässigkeitsanforderungen ist zu beachten, dass dynamische Testverfahren nur bedingt für ihre Prüfung geeignet sind. In [MilEA92] wird die Aussagekraft von stochastischen Tests mit gegebener Eingabeverteilung auf statistischer Basis diskutiert. Die Betrachtungen in [How98] machen deutlich, dass eine immense Menge von bestandenen dynamischen Testfällen in einem stochastischen Test erreicht werden muss, um belastbare Zuverlässigkeitsaussagen zu erhalten. Dennoch besteht die Notwendigkeit eines dynamischen Tests insbesondere auch in einem sicherheits- oder verfügbarkeitskritischen Umfeld, was beispielsweise in [Voa99] durch die Forderung eines dynamischen Tests mit einem abnormalen, zur typischen Nutzung inversen Profil unterstrichen wird. Beim Einsatz des dynamischen Tests zur Prüfung von Sicherheits- oder Zuverlässigkeitsanforderungen ist daher eine möglichst weitgehende automatisierte Unterstützung bei der Testdatengenerierung, der Testdurchführung und der Testauswertung nach funktionalen Aspekten von zentraler Bedeutung. Eine Grundlage für eine solche Unterstützung wird im folgenden Kapitel 4 erarbeitet.

Zur Absicherung von Anforderungen bezüglich Sicherheit und hoher Zuverlässigkeit wird jedoch der Einsatz weiterer Techniken der konstruktiven und analytischen Qualitätssicherung gefordert [IEC 61508 99]. Da das in diesem Kapitel beschriebene Modell speziell auf den dynamischen Test ausgerichtet wurde, ist es für eine Betrachtung solcher Techniken nicht geeignet.

3.3.4 Überwachung des internen Datenzustands

Die Beschreibung von Prüfverfahren, die über das Ausführen des Prüflings auf beliebigen Eingabedaten und die Beobachtung der Ausgabedaten hinausgehen, kann nur mit Hilfe spezieller Anpassungen vorgenommen werden. So ist eine Überwachung der internen Zustände des Prüflings, wie sie beispielsweise zur Erhöhung der Testbarkeit von Software [Voa92] oder zur Zertifizierung sicherheitskritischer Applikationen gefordert werden [BerStr96, Voa99], nicht direkt mit dem Modell vereinbar. Bei der Überwachung des internen Datenzustands werden fehlerhafte interne Zustände von einer Prüfinstanz beobachtet und gemeldet. Um dies im obigen Rahmen zu berücksichtigen, kann die Prüfinstanz als Teil des Prüflings betrachtet werden, ihre Ausgaben als weiterer Ausgabeparameterbereich. Die Spezifikation ist dann um den Sachverhalt zu ergänzen, dass eine entsprechende Warnung der Prüfinstanz niemals ausgegeben werden darf. Die Betrachtung des um die Prüfinstanz erweiterten Prüflings kann bei dynamischen Testverfahren, die die interne Struktur des Prüflings berücksichtigen, allerdings zu Abweichungen vom ursprünglichen Verfahren führen.

3.3.5 Perturbation des internen Datenzustands

Ebenso schwierig wie die Modellierung von Techniken zur Überwachung des internen Datenzustands gestaltet sich die Modellierung des Tests von Software durch das Einfügen von Perturbationen, also von gezielten oder zufälligen Veränderungen des internen Datenzustands des Prüflings. Solche Techniken werden ebenfalls für den Test sicherheitskritischer und fehlertoleranter Software diskutiert [MorMur96, Voa99], da durch sie beispielsweise die Funktionsweise der internen Sicherheitsmechanismen beobachtet werden kann. Auch hier ist eine Modellierung durch eine entsprechende Erweiterung des Prüflings und seines Eingabebereichs denkbar. Die Spezifikation muss beim Einfügen von Perturbationen die Wahrung eines als sicher angenommenen Zustands fordern. Auch bei einer solchen Erweiterung des Prüflings sind die Auswirkungen im Hinblick auf dynamische Testverfahren, die die Struktur des Prüflings verwenden, zu prüfen.

3.3.6 Einbettung in den Gesamtkontext der Qualitätssicherung

Das hier entwickelte Modell ist speziell ausgerichtet auf den dynamischen Test funktionaler Anforderungen. Aus diesem Grund ist es ungeeignet für eine Betrachtung anderer Verfahren der konstruktiven und der analytischen Qualitätssicherung, die nicht auf einer dynamischen Ausführung des Prüflings beruhen. So ist das Modell beispielsweise ungeeignet für eine Beschreibung des Vorgehens zur statischen Analyse des Prüflings oder für eine Beschreibung konstruktiver Verfahren zur Qualitätssicherung bei der Erstellung des Prüflings. Da das hier beschriebene Modell jedoch auf die wesentlichen Aspekte des dynamischen Tests reduziert ist und keinerlei Annahmen über die Gestalt und Herleitung von Prüfling und Spezifikation macht, kann es vermutlich problemlos in eine Gesamtbetrachtung der Qualitätssicherung zur Softwareentwicklung integriert werden. Ein Beispiel für eine solche Integration in eine integrierte Entwicklungsumgebung wird im Ausblick in Abschnitt 5.4 skizziert.

4 Beschreibung, Analyse und Ergänzung der funktionsorientierten Testverfahren

Das folgende Kapitel analysiert und beschreibt die wichtigsten funktionsorientierten Testverfahren mit Hilfe der in Kapitel 3 eingeführten Terminologie. Zunächst wird dazu das Modell zum dynamischen Testen um einige Definitionen zum funktionsorientierten Test ergänzt, die in der darauf folgenden Betrachtung der einzelnen Verfahren verwendet werden. Ziel des Kapitels ist es, eine weitgehend formalisierte Beschreibung der bisher meist nur informal beschriebenen funktionsorientierten Testverfahren zu erreichen, die eine systematische Untersuchung der Verfahren ermöglicht und eine Grundlage für ihre Weiterentwicklung oder Formalisierung sein kann. Gleichzeitig wird herausgestellt, welche Aspekte der Verfahren nicht formalisiert werden können, da eine informale Beschreibung als Grundlage dient. Hier wird deutlich, dass Kreativität und Intuition des Testers bei der analytischen Qualitätssicherung im Bezug auf eine informale Beschreibung der Anforderungen einen wichtigen Stellenwert haben.

Wie in Kapitel 3 bezeichnet I auch im Folgenden den Eingabebereich, O den Ausgabebereich, p einen Prüfling aus der Menge aller Prüflinge $\Pi_{I, O}$ über I und O und s eine Spezifikation aus der Menge aller Spezifikationen $\Sigma_{I, O}$ über I und O. Die Bezeichner t und i werden für dynamische Testfälle beziehungsweise Eingabedaten aus I verwendet, T für dynamische Testläufe über I und X für dynamische Testverfahren aus $\Xi_{I, O}$.

Die funktionsorientierten Testverfahren verwenden zur Auswahl und Bewertung von Testdaten ausschließlich die in der Spezifikation festgehaltenen funktionalen Anforderungen an den Prüfling. Alleinige Grundlage für die Selektion beziehungsweise Adäquatheit von dynamischen Testläufen ist damit die in Definition 3-2 eingeführte Spezifikation s. Die Eigenschaften des Prüflings p bleiben hierbei unberücksichtigt. Der Begriff des funktionsorientierten Testverfahrens kann daher wie folgt definiert werden.

Definition 4-1: Funktionsorientiertes Testverfahren, funktionales Testverfahren

(a) Ein dynamisches Testverfahren $X^{funk} \in \Xi_{I, O}$ wird als funktionsorientiertes Testverfahren bezeichnet, wenn es unabhängig vom Prüfling ist, wenn also die Bedingung

$$\forall(s \in \Sigma_{I, O})\ \forall(p_1 \in \Pi_{I, O})\ \forall(p_2 \in \Pi_{I, O})\ (X^{funk}(p_1, s) = X^{funk}(p_2, s))$$

erfüllt ist. In diesem Fall kann vereinfachend

$$X^{funk}(s) := X^{funk}(p, s)$$

geschrieben werden.

(b) Alternativ zum Begriff des funktionsorientierten Testverfahrens wird der des funktionalen Testverfahrens verwendet.

In der Praxis bestehen Spezifikationen oft aus einer endlichen Menge funktionaler Einzelanforderungen, die Teilaspekte der geforderten Funktionalität des Prüflings beschreiben und die teils in Prosa, teils mit semiformalen oder formalen Methoden beschrieben sind. Diese funktionalen Einzelanforderungen können als Teilspezifikationen betrachtet werden, die nur zu speziellen Eingabewerten Zuordnungen festlegen. Um eine Teilspezifikation zu einer Spezifikation entsprechend Definition 3-2 zusammenfassen zu können, werden die Zuordnungen in einer Relation aufgenommen. Zur Erreichung der Vollständigkeit dieser Relation werden zu Eingabedaten, die in einer Teilspezifikation nicht explizit

betrachtet werden, alle Ausgabe- und Performanzdaten akzeptiert. Somit können Teilspezifikationen durch Verknüpfung ihrer logischen Aussagen mit dem Operator „und“ beziehungsweise durch Schnitt ihrer Zuordnungsrelationen zu einer Gesamtspezifikation verbunden werden.

Definition 4-2: Teilspezifikation, Zusammenfassung von Teilspezifikationen, Beschreibung einer Spezifikation durch eine Menge von Teilspezifikationen

(a) Eine Teilbeschreibung der Spezifikation $s \in \Sigma_{I, O}$ wird als Teilspezifikation $s_1 \in \Sigma_{I, O}$ bezeichnet, wenn sie als Spezifikation entsprechend Definition 3-2 notiert werden kann und

$$s \subseteq s_1$$

gilt.

(b) Eine Spezifikation $s \in \Sigma_{I, O}$ wird als vollständig durch eine Menge von Teilspezifikationen $\{ s_1, s_2, \ldots, s_n \}$ $(n \in \aleph)$ beschrieben bezeichnet, wenn

$$s = \bigcap_{k \in \{1, \ldots, n\}} s_k$$

gilt.

Der Schnitt von Teilspezifikationen einer Spezifikation stellt immer eine Spezifikation dar. Hierfür ist wegen Definition 3-2 lediglich die Vollständigkeit des Schnitts zu zeigen. Diese ergibt sich wegen $s \subseteq \bigcap_{k \in \{1, \ldots, n\}} s_k$ direkt aus der Vollständigkeit von s.

Teilspezifikationen geben nach dieser Definition eine reduzierte Sicht auf die Gesamtspezifikation. Sie dürfen die relationale Beschreibung nicht weiter einschränken als die Gesamtspezifikation. Diese berücksichtigt dann die Gesamtheit der relationalen Aspekte und damit alle Anforderungen der Teilspezifikationen.

Werden in einem funktionsorientierten Test für jede Teilspezifikation s_k $(k \in \{ 1, \ldots, n \})$ geeignete funktionsorientierte Testverfahren X^{funk_k} gewählt, so ist es nahe liegend, diese im Hinblick auf die Gesamtspezifikation zu kombinieren. Somit bietet sich das dynamische Testverfahren

$$X^{funk} := \circledcirc_{k \in \{ 1, \ldots, n \}} X^{funk_k}$$

zur Prüfung der Gesamtspezifikation an. Dieses Verfahren ist wegen der Unabhängigkeit der Verfahren X^{funk_k} vom Prüfling ebenfalls ein funktionsorientiertes Testverfahren nach Definition 4-1. Wegen der Definition der Kombination in Definition 3-12 ordnet das kombinierte Testverfahren X^{funk} zur Gesamtspezifikation nur diejenigen dynamischen Testläufe zu, die von allen Einzelverfahren als adäquat bewertet werden. Unter der Voraussetzung, dass für jede Teilspezifikation s_k mindestens ein geeigneter dynamischer Testlauf T_k existiert, ist wegen der Monotonie der dynamischen Testverfahren beispielsweise der dynamische Testlauf

$$\bigcup_{k \in \{1, \ldots, n\}} T_k \in X^{funk}(s)$$

für die Prüfung der Gesamtspezifikation adäquat. Ein solcher dynamischer Testlauf ergibt sich im dynamischen Test bei der sukzessiven Prüfung der in den Teilspezifikationen beschriebenen Funktionalität. Es können jedoch noch weitere, weniger umfangreiche dynamische Testläufe in $X_{funk}(s)$ enthalten sein, in denen einzelne Testfälle die Verfahren zu mehreren Teilspezifikationen befriedigen.

Diese Definitionen beschreiben zusätzlich zum in Kapitel 3 eingeführten Modell des dynamischen Tests die Aspekte, die sich speziell auf den funktionsorientierten Test beziehen. So legt Definition 4-1 fest, was unter einem funktionsorientierten Testverfahren zu verstehen ist. Definition 4-2 zeigt die

Ableitung einer relationalen Spezifikation aus einer Zusammenstellung heterogener Anforderungen, die als Teilspezifikationen verstanden und relational zu einer Gesamtspezifikation verknüpft werden. Auf dieser Basis wird das typische Vorgehen im funktionsorientierten Test beschrieben, bei dem eine umfangreiche Spezifikation sukzessive analysiert wird und dynamische Testverfahren separat auf Teilanforderungen angewendet werden. Hierbei ist sicherzustellen, dass jedes auf eine Teilspezifikation angewendete funktionsorientierte Testverfahren durch einen adäquaten dynamischen Testlauf befriedigt wird. Dies kann formal durch die Kombination der Einzelverfahren zu einem Gesamtverfahren beschrieben werden.

Auf Basis des so ergänzten Modells zum funktionsorientierten Test werden im Folgenden die wichtigsten funktionsorientierten Testverfahren formuliert und analysiert. Hierbei werden zunächst die Verfahren der funktionalen Partitionierung betrachtet, anschließend die Verfahren zur Betrachtung von Ursachen und ihren Wirkungen analysiert, weiterhin die auf graphischen Spezifikationsformen operierenden Verfahren untersucht und anschließend die Verfahren zum Test zustandsbasierter Spezifikationen und Systeme beschrieben. Abschließend werden die erweiterten Ansätze von Howden zum funktionsorientierten Test diskutiert.

4.1 Verfahren der funktionalen Partitionierung

Dem Ziel einer vollständigen Überprüfung der in der Spezifikation festgeschriebenen Funktionalität im dynamischen Test steht im Allgemeinen die Menge und Komplexität der zu berücksichtigenden Ein- und Ausgabekonstellationen entgegen. Ein vollständiger Test, wie in Verfahren 3-1 beschrieben, ist aufgrund des Umfangs der zu prüfenden Eingabedaten normalerweise nicht möglich. Daher ist es notwendig, eine geeignete Stichprobe für den dynamischen Test auszuwählen. Diese muss zwei konkurrierenden Zielen genügen [Mye79]: Zum einen ist eine umfassende Prüfung der relevanten Betriebssituationen anzustreben, um die Umsetzung der funktionalen Anforderungen der Spezifikation im Prüfling beurteilen zu können. Zum anderen ist die Anzahl der dynamischen Testfälle gering zu halten, um den Umfang des dynamischen Tests beherrschbar zu machen. Dieser Zielkonflikt führt zur Idee einer Aufteilung des Ein- und Ausgabebereichs in Teilbereiche, über denen eine funktional äquivalente Behandlung gefordert ist. Aus den Teilbereichen werden mit unterschiedlichen Methoden Repräsentanten ausgewählt, die im Rahmen des Tests als dynamische Testfälle verwendet werden. Die Idee einer solchen funktionalen Partitionierung des Ein- und Ausgabebereichs liegt verschiedenen dynamischen Testverfahren zugrunde, die in den folgenden Abschnitten diskutiert werden.

4.1.1 Funktionale Partitionierung des Eingabebereichs

Der funktionale Partitionentest ist im eigentlichen Sinn kein eigenständiges dynamisches Testverfahren, er fasst vielmehr die Gemeinsamkeiten einer ganzen Klasse von dynamischen Testverfahren zusammen. In allgemeiner Form wird er in der Literatur als „partition testing“ bezeichnet und häufig referenziert, so zum Beispiel als Vergleichsverfahren für theoretische Untersuchungen [WeyOst80, HamTay90, Ham00] oder als Basis für vergleichende Analysen der Effizienz von Testverfahren [DurNta84, FraWey93, Nta01]. Aufgrund des Stellenwerts des „partition testing“ wird daher im Folgenden die auf einer funktionalen Betrachtung beruhende Variante als eigenständiges Verfahren eingeführt.

In der Literatur werden unterschiedliche Methoden zur Festlegung der Partitionen angegeben. Sie beziehen sich auf Prüfling oder Spezifikation oder nutzen weitere Ansätze, wie beispielsweise den

Ansatz der Fehlererwartungen. Der allgemeine Partitionentest muss also nicht funktional geprägt sein. Eine funktionale Partitionierung wird jedoch häufig empfohlen. So wird beispielsweise in [WeyOst80] wird die Ableitung einer Partitionierung des Eingabebereichs aus der Spezifikation empfohlen und in [Ham00] explizit auf die Eignung einer stückweise stetig definierten Spezifikation hierfür verwiesen.

Meist wird unter einer Partitionierung eine überlappende Aufteilung des Eingabebereichs verstanden, die diesen voll ausschöpft, deren Vereinigung also den Eingabebereich selbst ergibt. Selten liegt eine Partitionierung im mathematischen Sinn vor, also eine Aufteilung in disjunkte, die Grundmenge voll ausschöpfende Teilmengen. Die folgende, hier verwendete Definition erlaubt daher ein Überlappen der Teilmengen.

Definition 4-3: Partitionierung einer Menge, endliche Partitionierung, Partition

Zu einer beliebigen Menge M wird ein System von Teilmengen $Part_M \subseteq \wp(M)$ als Partitionierung bezeichnet, falls die Vereinigung aller Mengen des Teilmengensystems $Part_M$ die Menge M selbst ergibt. Eine Partitionierung wird als endlich bezeichnet, wenn sie nur endlich viele Teilmengen umfasst. Die Teilmengen $A \in Part_M$ werden als Partitionen bezeichnet.

Grundidee des funktionsorientierten Tests auf Basis einer Partitionierung des Eingabebereichs ist die Zusammenfassung von Elementen, die eine gleichartige funktionale Behandlung erfordern. Dieser Gedanke führt zur folgenden Definition einer funktionalen Partitionierung.

Definition 4-4: Funktionale Partitionierung des Eingabebereichs zu einer Spezifikation

Eine endliche Partitionierung $Part_I$ des Eingabebereichs I wird als funktionale Partitionierung $Fpart_I$ zu einer Spezifikation $s \in \Sigma_{I, O}$ bezeichnet, wenn für jede Partition $A \in Fpart_I$ die folgende Bedingung erfüllt ist:

In der Spezifikation s wird für alle Eingabedaten $i \in A$ eine gleichartige funktionale Behandlung gefordert. (funktionale Äquivalenz)

Die Partitionen $A \in Fpart_I$ werden als funktionale Partitionen des Eingabebereichs bezeichnet.

Zum dynamischen Test wird aus jeder funktionalen Partition jeweils ein Repräsentant ausgewählt, dessen funktionale Behandlung durch den Prüfling im dynamischen Test überprüft wird. Dieses Vorgehen beschreibt das folgende Verfahren.

Verfahren 4-1: Funktionaler Partitionentest über dem Eingabebereich

(a) Voraussetzung:

Gegeben sei die Zuordnung

$$Fpart_I: \quad \Sigma_{I, O} \rightarrow \wp^{fin}(\wp(I)) \qquad s \mapsto Fpart_I(s),$$

einer funktionalen Partitionierung des Eingabebereichs zu einer Spezifikation.

(b) Verfahrensbeschreibung:

Der funktionale Partitionentest über dem Eingabebereich wird beschrieben durch

$$X^{part}: \quad \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(A \in Fpart_I(s))\ \exists(t \in T)\ (t \in A).$$

Verfahren 4-1 setzt die Ableitung einer funktionalen Partitionierung aus der Spezifikation voraus, geht aber nicht auf dazu notwendigen Schritte ein. Wird in der Spezifikation eine konkrete Partitionierung angegeben, kann die Zuordnung übernommen und das Verfahren vollständig beschrieben werden. Ist dies aufgrund der informalen Beschreibung der Anforderungen nicht der Fall, ist es Aufgabe des Testers, eine geeignete Partitionierung zu erarbeiten. Hierbei sind Kreativität und Intuition des Testers wichtige Faktoren. Es ist denkbar, dass eine Spezifikation ungeeignet zur Ableitung einer funktionalen Partitionierung ist. In diesem Fall wird die funktionale Partitionierung $Fpart_I(s)$ lediglich eine Partition enthalten, nämlich den Eingabebereich, für den die funktionale Äquivalenz in der Behandlung durch den Prüfling besteht. Somit wird das Prädikat P_s für beliebige dynamische Testläufe $T \in \wp^{fin}(I)$ erfüllt, dass dynamische Testverfahren hat dann keine Aussagekraft.

Auf Basis einer vorgegebenen Partitionierung kann der Tester bei der Auswahl der Testdaten, beispielsweise durch zufällige Auswahl, unterstützt werden. Eine entsprechende Unterstützung bei der Testdatengenerierung leisten beispielsweise die in [GroGriWeg93, LehWeg00, MeyGrüSpa03] beschriebenen Werkzeuge. Grundlage ist hier eine manuelle Einteilung der Parameterbereiche, aufgrund derer der Tester mit Hilfe graphischer Methoden bei der Testfallgenerierung unterstützt wird. Eine automatisierte Testdatenbeschreibung und auch –generierung wird angestrebt und ist bereits teilweise umgesetzt. Eine automatisierte Vollständigkeitsbewertung zu gewählten Testdaten ist möglich, soweit die Zugehörigkeit der Testdaten zu den funktionalen Partitionen automatisiert ermittelt werden kann.

Das Vorgehen zum dynamischen Test auf Basis einer funktionalen Partitionierung des Eingabebereichs kann auch im Hinblick auf eine Partitionierung des Ausgabebereichs erweitert werden. Hierbei ist zu beachten, dass eine Partitionierung des Ausgabebereichs, die nach Definition 4-3 den gesamten Ausgabebereich ausschöpfen muss, auch Ausgabedaten enthalten kann, die nicht im Wertebereich der Spezifikation liegen. Bei einer Partitionierung nach funktionalen Aspekten sollten solche Ausgabedaten in einer speziellen Partition zusammengefasst werden, deren Prüfung im dynamischen Test nicht verlangt wird.

Auch kann in Anlehnung an die Ideen in [Mye79] im Rahmen einer fehlerbasierten Teststrategie versucht werden, im dynamischen Test Ausgaben dieser Partition zu erreichen. Von diesen Anpassungen abgesehen kann die Betrachtung der funktionalen Ausgabeäquivalenzklassen analog zur Betrachtung der funktionalen Eingabeäquivalenzklassen in Verfahren 4-1 ergänzt werden.

4.1.2 Funktionale Äquivalenzklassenanalyse

Meist setzen sich Ein- und Ausgabebereich, wie in Definition 3-1 beschrieben, aus mehreren Parameterbereichen zusammen. Die funktionalen Anforderungen der Spezifikation werden in diesem Fall meist mit Bezug auf die Ausprägung einzelner Parameter formuliert. Das Verfahren des dynamischen Tests auf Basis einer funktionalen Äquivalenzklassenanalyse nutzt diese spezielle Gestaltung der Spezifikation. Es wurde bereits 1979 von Myers veröffentlicht [Mye79], wird aber auch heute noch als universell einsetzbares Verfahren für den funktionsorientierten Test empfohlen [Lig02]. Grundlage ist

eine Partitionierung der Parameterbereiche nach funktionalen Aspekten, die im dynamischen Test durch Repräsentanten zu überprüfen ist. Das Verfahren folgt somit einem ähnlichen Grundgedanken wie der dynamische Test auf Basis einer funktionalen Partitionierung des Eingabebereichs.

Formale Voraussetzung für den dynamischen Test auf Basis einer funktionalen Äquivalenzklassenanalyse ist die Darstellung des Ein- und Ausgabebereiches als kartesisches Produkt einzelner Parameterbereiche.

Voraussetzung 4-1: Darstellung des Ein- und Ausgabebereichs als kartesische Produkte von Parameterbereichen

Eingabebereich I und Ausgabebereich O können als kartesische Produkte

$I = I_1 \times I_2 \times \ldots \times I_k \qquad (k \in \aleph)$

$O = O_1 \times O_2 \times \ldots \times O_l \qquad (l \in \aleph)$

von Ein- beziehungsweise Ausgabeparameterbereichen dargestellt werden.

Zur funktionalen Äquivalenzklassenbildung wird auf Basis der gegebenen Spezifikation s eine endliche Partitionierung der Parameterbereiche entsprechend Definition 4-3 vorgenommen. Im Vordergrund steht dabei wiederum der Gedanke der funktionalen Äquivalenz der Elemente einer Partition: Zusammengefasst werden Eingabeparameter, deren äquivalente Behandlung durch die Spezifikation gefordert wird, sowie Ausgabeparameter, die laut Spezifikation durch eine äquivalente Behandlung zu erzeugen sind. Die Partitionen werden als funktionale Äquivalenzklassen bezeichnet.

Es werden gültige und ungültige funktionale Äquivalenzklassen unterschieden. Als gültig wird eine funktionale Äquivalenzklasse über dem Eingabeparameterbereich bezeichnet, wenn die Spezifikation eine gleichartige funktionale Behandlung für ihre Elemente fordert. Als ungültig gilt sie, wenn eine Fehlerbehandlung für die enthaltenen Parameter gefordert ist. Diese Ideen zum Begriff der funktionalen Äquivalenzklassen über den Eingabeparameterbereichen und zu ihrer Gültigkeit werden in der folgenden Definition zusammengefasst.

Definition 4-5: Funktionale Äquivalenzklassen eines Eingabeparameterbereichs, Gültigkeit von funktionalen Äquivalenzklassen eines Eingabeparameterbereichs, Gültigkeit von Eingabeparametern

Eine endliche Partitionierung $Part_{I_v}$ des Eingabeparameterbereichs I_v wird genau dann als System funktionaler Äquivalenzklassen $FÄq_{I_v,s}$ zu einer Spezifikation $s \in \Sigma_{I,O}$ bezeichnet, wenn für jede Partition $A \in Part_{I_v}$ eine der folgenden Bedingungen erfüllt ist:

(a) In der Spezifikation s wird für alle Eingabedaten $i \in I$ mit Eingabeparameter $i_v \in A$, die keine Fehlerbehandlung erfordern, eine gleichartige funktionale Behandlung gefordert.

(b) In der Spezifikation s wird für alle Eingabedaten $i \in I$ mit Eingabeparameter $i_v \in A$ eine spezielle Fehlerbehandlung gefordert.

Die Partitionen $A \in FÄq_{I_v,s}$ werden als funktionale Äquivalenzklassen des Eingabeparameterbereichs bezeichnet.

Falls für eine funktionale Äquivalenzklasse A die Bedingung (a) gilt, wird sie als gültige funktionale Äquivalenzklasse bezeichnet. Dies wird im Folgenden mit dem Prädikat $\text{Gültig}_{s,I_v}(A)$ beschrieben. Parameter i_v, die in einer gültigen funktionalen Äquivalenzklasse enthalten sind, werden ebenfalls als gültig bezeichnet, was mit dem Prädikat $\text{gültig}_{s,I_v}(i_v)$ ausgedrückt wird. Falls eine funktionale Äquivalenzklasse nicht gültig ist, wird sie und werden die enthaltenen Parameter als ungültig bezeichnet.

Für funktionale Äquivalenzklassen der Ausgabeparameterbereiche muss, den Darstellungen [Mye79, Lig02] folgend, bei der Definition der Gültigkeit ihre spezifizierte Erreichbarkeit berücksichtigt werden: Als gültig gilt eine funktionale Äquivalenzklasse, wenn die enthaltenen Ausgabeparameter laut Spezifikation durch eine gleichartige Behandlung erzeugt werden sollen. Dies kann auch eine Fehlerbehandlung sein. Als ungültig wird sie bezeichnet, wenn sie nicht in dem in Definition 3-3 wie eingeführten Wertebereich der Spezifikation über dem Ausgabeparameterbereich enthalten ist. Eine Beschreibung der Gültigkeit einzelner Ausgabeparameter wird für die Verfahrensbeschreibung nicht benötigt.

Definition 4-6: Funktionale Äquivalenzklassen eines Ausgabeparameterbereichs, Gültigkeit von funktionalen Äquivalenzklassen eines Ausgabeparameterbereichs

Eine endliche Partitionierung Part_{O_w} eines Ausgabeparameterbereichs O_w wird genau dann als System funktionaler Äquivalenzklassen $\text{FÄq}_{O_w,s}$ zu einer Spezifikation $s \in \Sigma_{I,\,O}$ bezeichnet, wenn für jede Partition $A \in \text{Part}_{O_w}$ eine der folgenden Bedingungen erfüllt ist:

(a) Für die Partition A gilt $A \subseteq W^{O_w,s}$, das heißt sie ist vollständig im Wertebereich der Spezifikation über dem Parameterbereich enthalten. Weiterhin wird in der Spezifikation s für alle Ausgabedaten $o \in O$ mit Ausgabeparameter $o_w \in A$ die Erzeugung durch eine gleichartige Behandlung gefordert. Dies kann eine funktionale Behandlung oder eine Fehlerbehandlung sein.

(b) Für die Partition A gilt $A \subseteq (O_w \setminus W^{O_w,s})$, das heißt sie liegt vollständig außerhalb des Wertebereichs der Spezifikation über dem Parameterbereich.

Die Partitionen $A \in \text{FÄq}_{O_w,s}$ werden als funktionale Äquivalenzklassen des Ausgabeparameterbereichs bezeichnet. Falls für eine funktionale Äquivalenzklasse A die Bedingung (a) gilt, wird sie als gültige funktionale Äquivalenzklasse bezeichnet. Dies wird mit dem Prädikat $\text{Gültig}_{s,O_w}(A)$ beschrieben. Andernfalls wird sie als ungültige funktionale Äquivalenzklasse bezeichnet.

Zur Beschreibung des dynamischen Testverfahrens auf Basis einer funktionalen Äquivalenzklassenanalyse wird im Folgenden die Existenz einer Zuordnung funktionaler Äquivalenzklassensysteme der Parameterbereiche zur Spezifikation angenommen. Eine solche Zuordnung kann immer angegeben werden: Im ungünstigsten Fall wird auf Basis der Informationen aus der Spezifikation keine Einteilung der Parameterbereiche erreicht; die Parameterbereiche selbst stellen die funktionalen Äquivalenzklassen dar und die funktionale Äquivalenz bezieht sich lediglich auf die Behandlung gemäß der Spezifikation. Das Verfahren liefert in diesem Fall keinen Beitrag zur Auswahl und Bewertung der Testdaten. Ist die Spezifikation jedoch zur Ableitung funktionaler Äquivalenzklassen geeignet, empfiehlt es sich, eine möglichst feine Einteilung der Klassen vorzunehmen, um eine fundierte Prüfung der Funktionalität

durch das Verfahren sicherzustellen. Auf dieser Basis kann das Verfahren wie folgt beschrieben werden.

Verfahren 4-2: Dynamischer Test auf Basis einer funktionalen Äquivalenzklassenbildung

(a) Voraussetzungen:

Der Eingabebereich I und Ausgabebereich O können als kartesische Produkte

$$I = I_1 \times I_2 \times \ldots \times I_k \qquad (k \in \aleph)$$
$$O = O_1 \times O_2 \times \ldots \times O_l \qquad (l \in \aleph)$$

von Parameterbereichen dargestellt werden.

Für die Eingabeparameterbereiche I_v ($1 \leq v \leq k$) und Ausgabeparameterbereiche O_w ($1 \leq w \leq l$) können durch

$$FÄq_{I_v} : \Sigma_{I,O} \rightarrow \wp^{fin}(\wp(I_v)), \quad s \mapsto FÄq_{I_v}(s)$$

und

$$FÄq_{O_w} : \Sigma_{I,O} \rightarrow \wp^{fin}(\wp(O_w)), \quad s \mapsto FÄq_{O_w}(s)$$

Systeme funktionaler Äquivalenzklassen über den Parameterbereichen aus der Spezifikation abgeleitet werden, deren Gültigkeit durch die Prädikate $Gültig_{s,I_v}$, $gültig_{s,I_v}$ und $Gültig_{s,O_w}$ ausgedrückt wird.

(b) Verfahrensbeschreibung:

Der dynamische Test auf Basis einer funktionalen Äquivalenzklassenanalyse wird beschrieben durch

$$X^{äquiv} : \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I)), \quad s \mapsto \{ T \in \wp^{fin}(I) : P_{s,1}(T) \wedge P_{s,2}(T) \}$$

mit

$$P_{s,1}(T) := \forall(v \in \{1, \ldots, k\})\, \forall(A \in FÄq_{I_v}(s))\, \exists(t \in T)\, ((t_v \in A) \wedge (\forall(x \in \{1, \ldots, k\} \setminus \{v\})\, gültig_{s,I_x}(t_x)))$$

$$P_{s,2}(T) := \forall(w \in \{1, \ldots, l\})\, \forall(A \in FÄq_{O_w}(s))\, (Gültig_{s,O_w}(A) \Rightarrow (\exists(t \in T)\, \forall(o \in W^{O,s,t})\, (o_w \in A))).$$

Das Verfahren der funktionalen Äquivalenzklassenbildung stellt zwei Bedingungen an die Auswahl und Bewertung von dynamischen Testläufen. Zunächst wird durch das Prädikat $P_{s,1}$ gefordert dass jede funktionale Äquivalenzklasse eines Eingabeparameterbereichs durch mindestens einen dynamischen Testfall t überprüft wird. Dadurch wird eine breite funktionale Abdeckung der Spezifikation durch geeignete Repräsentanten angestrebt. Die weiteren Eingabeparameter dürfen ausschließlich aus gültigen funktionalen Äquivalenzklassen stammen, damit die Reaktion des Prüflings nicht durch die Fehlerbehandlung eines anderen Eingabeparameters maskiert wird. Durch $P_{s,2}$ wird sichergestellt, dass die gültigen funktionalen Äquivalenzklassen des Ausgabebereichs im dynamischen Testlauf mindestens einmal erreicht werden. Diese Bedingung ist auch dann sicherzustellen, wenn zu einem Eingabedatum mehrere Ausgabedaten der Spezifikation entsprechen. Deshalb wird in $P_{s,2}$ eine Bedingung für alle Parameter im Wertebereich des dynamischen Testfalls gestellt. Hierzu ist auch eine

alternative Definition denkbar, die für jede gültige funktionale Äquivalenzklasse des Ausgabebereichs fordert, dass das Ergebnis mindestens eines dynamischen Testfalls mit ausschließlich gültigen Eingabeparametern in der Ausgabeäquivalenzklasse liegt. Anstelle von $P_{s,2}$ könnte in diesem Fall das Prädikat

$$\tilde{P}_{s,2}(T) := \forall(w \in \{1, \ldots, l\})\ \forall(A \in FÄq_{O_w}(s))\ (Gültig_{s,O_w}(A) \Rightarrow (\exists(t \in T)\ ((p(t))_w \in A))$$

verwendet werden. Dabei ist jedoch zu beachten, dass das resultierende dynamische Testverfahren nicht mehr allein im Hinblick auf die Spezifikation definiert ist, sondern den Ausgang des Testfalls für den Prüfling mit berücksichtigt. Dieser ist jedoch zum Zeitpunkt der Testdatenselektion nicht bekannt und sollte auch nicht in die Adäquatheitsprüfung mit einfließen, da er möglicherweise fehlerbehaftet ist. Aus diesem Grund wird in der Verfahrensbeschreibung das Prädikat $P_{s,2}$ verwendet.

Zusätzlich zu den in Verfahren 4-2 geforderten dynamischen Testfällen werden in dem in [Mye79] beschriebenen Verfahren noch weitere Testfälle gefordert, mit deren Ausführung durch den Prüfling Ausgaben provoziert werden sollen, die ungültigen funktionalen Äquivalenzklassen der Ausgabeparameterbereiche zuzurechnen sind. Diese Forderung bezieht sich über den Inhalt der Spezifikation hinaus auf erwartete Irrtümer bei der Realisierung des Prüflings. Das Verfahren ist damit nicht spezifikationsbasiert, sondern bezieht eine Fehlererwartung des Testers mit ein. Da diese Fehlererwartung und die intuitive Auswahl der Testfälle nicht formal beschrieben werden kann, wird sie in der obigen Verfahrensbeschreibung nicht berücksichtigt. Myers selbst unterstreicht den intuitiven Charakter, in dem er die von ihm beschriebene funktionale Äquivalenzklassenanalyse als Geisteshaltung, nicht als Verfahren bezeichnet.

In [Lig02] werden die funktionalen Ausgabeäquivalenzklassen zur Ableitung funktionaler Äquivalenzklassen über den Eingabeparameterbereichen verwendet: Für jede der hier angegebenen funktionalen Ausgabeäquivalenzklassen werden diejenigen Elemente eines Eingabeparameterbereichs zusammengefasst, die laut Spezifikation die Ausgabe eines Parameters einer funktionalen Ausgabeäquivalenzklasse erfordern. Für gültige funktionale Ausgabeäquivalenzklassen ist eine solche Zuordnung durchführbar, soweit die Spezifikation eine derartige Analyse erlaubt. Für ungültige funktionale Ausgabeäquivalenzklassen, wie sie in [Mye79] beschrieben werden, ist diese Betrachtungsweise jedoch problematisch, da eine Spezifikation nach Definition 4-6 mit einer ungültigen funktionalen Ausgabeäquivalenzklasse keine Eingabedatensätze verbindet. In der obigen Beschreibung des dynamischen Testverfahrens auf Basis der funktionalen Äquivalenzklassenanalyse wurden daher die funktionalen Äquivalenzklassen über dem Ein- und Ausgabebereich referenziert, was bei gleicher Aussagekraft die Formalisierung vereinfacht.

Im Rahmen der obigen Verfahrensbeschreibung ist das in [Mye79] und [Lig02] genannte Ziel nicht berücksichtigt, die gültigen funktionalen Äquivalenzklassen der Ein- und Ausgabeparameterbereiche mit möglichst wenigen Testfällen abzudecken. Grund ist die Definition 3-8 des dynamischen Testverfahrens, die alle einem Prüfling und einer Spezifikation zugeordneten dynamischen Testläufe als gleichwertig betrachtet werden. Die Bevorzugung von Testläufen mit geringem Umfang kann durch Einführung einer Strategie bei der Auswahl der dynamischen Testläufe berücksichtigt werden, die die Anzahl der dynamischen Testfälle im gewählten dynamischen Testlauf minimiert. Entsprechende Erweiterungen des Modells wurden in Abschnitt 3.3.2 angesprochen.

Der dynamische Test auf Basis einer funktionalen Äquivalenzklassenanalyse kann als Spezialfall eines funktionalen Partitionentests aufgefasst werden: Wird zu jeder funktionalen Äquivalenzklasse eine Partition gebildet, die alle Daten mit Parameter aus der funktionalen Äquivalenzklasse zusammen-

fasst, so stellt die Durchführung eines dynamischen Tests auf Basis der funktionalen Äquivalenzklassenanalyse einen vollständigen Test der so gewonnenen Partitionierung im Sinne von Verfahren 3-1 sicher. Umgekehrt stellt aber ein funktionaler Partitionentest auf Basis der so gewonnen Partitionierung nicht die Erfüllung des dynamischen Testverfahrens auf Basis der funktionalen Äquivalenzklasse sicher, da hierfür die Gültigkeit der funktionalen Äquivalenzklassen, der Parameter und der Eingabedaten sowie die auf dieser Basis geforderten Parameterkombinationen berücksichtigt werden müssen.

Zur Bildung der funktionalen Äquivalenzklassen sind noch einige Anmerkungen zu machen. Zunächst fällt auf, dass, ähnlich wie bei der Definition der Partitionen in Abschnitt 4.1.1, eine nicht-disjunkte Aufteilung eines Parameterbereichs zulässig ist. Die funktionalen Äquivalenzklassen eines Parameterbereichs bilden daher keine Äquivalenzklassen im mathematischen Sinn: In der Mathematik wird eine zweistellige Relation über einer Menge als Äquivalenzrelation bezeichnet, wenn sie reflexiv, transitiv und symmetrisch ist. Fasst man die Elemente zusammen, die in einer solchen Relation zueinander stehen, erhält man ein System von disjunkten Teilmengen der ursprünglichen Menge, dessen Vereinigung die Menge selbst ergibt. Diese Mengen werden als Äquivalenzklassen bezeichnet [ReiSoe01]. Die Möglichkeit der überlappenden Einteilung eines Parameterbereichs in funktionale Äquivalenzklassen entstammt der Formulierung des Verfahrens in der Literatur [Mye79, Lig02].

Grundvoraussetzung für den sinnvollen Einsatz des Verfahrens ist die Eignung der Spezifikation zur Ableitung funktionaler Äquivalenzklassen über den Ein- und Ausgabeparameterbereichen. Die Funktionalität sollte daher in erster Linie im Hinblick auf die Ausprägung einzelner Parameter beschrieben sein und weniger auf einer Verknüpfung der Parameter beruhen. Falls die Äquivalenzklassenbildung für einzelne Parameterbereiche nicht geeignet ist, kann es sinnvoll sein, mehrere Parameterbereiche zu einem einzelnen, dann mehrdimensionalen Parameterbereich zusammenzufassen. Dies entspricht zwar nicht den ursprünglichen Darstellungen des Verfahrens, ist aber auf Basis der hier verwendeten Definitionen ohne Einschränkung möglich, da in Definition 3-1 keine Annahme über die interne Struktur eines Parameterbereichs gemacht wird. Das in der Literatur oft verwendete Beispiel der Dreiecksklassifikation [z. B. Mye79] macht deutlich, wie mit Hilfe der Betrachtung eines zusammengesetzten Parameterbereichs funktionale Äquivalenzklassen bestimmt werden können: Hier beschreiben drei numerische Eingabeparameter die Seitenlängen a, b und c eines Dreiecks, das durch den Prüfling als gültig oder ungültig, und im Falle der Gültigkeit als ungleichseitig, gleichschenklig oder gleichseitig zu charakterisieren ist. Betrachtet man die numerischen Eingabeparameter getrennt, so ergibt sich nur sehr beschränkte Möglichkeit für eine funktionale Äquivalenzklassenanalyse. Erst die gemeinsame Betrachtung der Eingabeparameterbereiche führt zur Bestimmung wichtiger dynamischer Testfälle. Zum Beispiel kann mit der als ungültig anzunehmenden Äquivalenzklasse

$$\{ (a, b, c) \in \Re^3 : (a > 0) \wedge (b > 0) \wedge (c > 0) \wedge (a + b < c) \}$$

die Erkennung unzulässiger Eingabedaten geprüft werden. Bei einer getrennten Betrachtung der Eingabeparameter ist eine entsprechende Definition nicht möglich. Dieses Beispiel zeigt, wie eine verallgemeinerte Herangehensweise an die funktionale Äquivalenzklassenanalyse, die eine Loslösung von der herkömmlichen Idee des eindimensionalen Parameterbereichs und der nahe liegenden Äquivalenzklassengestaltung durch Auswahl von Intervallen zulässt, zu einer sinnvollen Unterstützung im dynamischen Test führen kann. Die hier möglichst allgemein gehaltenen Definitionen unterstützen eine solche verallgemeinerte Sicht.

Kann für eine Spezifikation hingegen keine hinreichende Einteilung der Parameterbereiche in funktionale Äquivalenzklassen erreicht werden, so werden die Parameterbereiche als funktionale Äquiva-

lenzklassen betrachtet. Das Verfahren akzeptiert in diesem Fall beliebige dynamische Testläufe und liefert keinen sinnvollen Beitrag zum dynamischen Test.

Die Ableitung der funktionalen Äquivalenzklassen aus einer informal formulierten Spezifikation wird in Verfahren 4-2 durch eine Zuordnung beschrieben, deren Schritte nicht näher genannt sind. Hier wird deutlich, dass zur Einteilung der Parameterbereiche in funktionale Äquivalenzklassen die Kreativität und Intuition des Testers erforderlich ist. Um eine weitergehende formale Beschreibung des Verfahrens zu erreichen, müssten zusätzliche Vorgaben zur Gestaltung der Spezifikation gemacht werden, die eine eindeutige Identifikation der funktionalen Äquivalenzklassen der Parameterbereiche gestatten. Möchte man die Prinzipien der funktionalen Äquivalenzklassenbildung bereits in der Spezifikation verankern, so ist dabei zusätzlich zu berücksichtigen, dass auch die Verknüpfung funktionaler Anforderungen an Einzelparameter für alle Eingabedaten angegeben werden müssen, um die in Definition 3-2 eingeführte relationale Gestalt der Spezifikation vollständig zu beschreiben. Auf einer entsprechend gestalteten Spezifikation ist dann eine weitgehende Standardisierung und Automatisierung der Testdatenselektion und der Bewertung der Testdatenadäquatheit mit dem dynamischen Test auf Basis einer funktionalen Äquivalenzklassenanalyse möglich. Eine Werkzeugunterstützung für den dynamischen Test auf Basis der funktionalen Äquivalenzklassenanalyse wird beispielsweise in [GroGriWeg93, LehWeg00, MeyGrüSpa03] beschrieben.

4.1.3 Kombinatorischer Test auf Basis einer funktionalen Äquivalenzklassenanalyse

Wenn die spezifizierten Systemreaktionen weniger von einzelnen Parametern, sondern vielmehr von der Kombination verschiedener Parameter des Eingabedatensatzes abhängen, reicht die im dynamischen Test auf Basis einer funktionalen Äquivalenzklassenanalyse geforderte Testfallmenge zur Prüfung der Umsetzung der Funktionalität im Prüfling häufig nicht aus. In diesem Fall kann eine funktionale Äquivalenzklassenanalyse nicht als geeignetes Prüfverfahren angesehen werden [Lig02]. So fordert Verfahren 4-2 nur die Verwendung der Parameter der funktionalen Äquivalenzklassen im dynamischen Test, ihre Kombination bleibt unberücksichtigt. Häufig bleibt, wenn keine weiteren Ansätze für einen systematischen funktionsorientierten Test verfolgt werden können, nur ein kombinatorischer Ansatz, bei dem jede funktionale Äquivalenzklassen eines Parameterbereiches mit den funktionalen Äquivalenzklassenanalysen der anderen Parameterbereiche kombiniert wird [How80, SchKor00]. Dieser Ansatz entspringt dem Wunsch nach einer möglichst vollständigen Prüfung im dynamischen Test und gestaltet sich oft entsprechend der folgenden Darstellung, die der Beschreibung in [SchKor00] folgt.

Verfahren 4-3: Kombinatorischer Test auf Basis einer funktionalen Äquivalenzklassenbildung

(a) Voraussetzungen:

Der Eingabebereich kann als kartesisches Produkt

$$I = I_1 \times I_2 \times \ldots \times I_k \qquad (k \in \mathbb{N})$$

von Parameterbereichen dargestellt werden.

Für die Eingabeparameterbereiche I_v ($1 \leq v \leq k$) können durch

$$\begin{array}{lccl} FÄq_{I_v}: & \Sigma_{I,O} & \rightarrow & \wp^{fin}(\wp(I_v)) \\ & s & \mapsto & FÄq_{I_v}(s) \end{array}$$

Systeme funktionaler Äquivalenzklassen über den Parameterbereichen aus der Spezifikation abgeleitet werden.

(b) Der kombinatorische Test auf Basis einer funktionalen Äquivalenzklassenanalyse wird beschrieben durch

$$X^{komb}: \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_{s,1}(T) \}$$

mit

$$P_{s,1}(T) := \forall(v_1 \in \{1, \ldots, k\}) \forall(A_1 \in F\ddot{A}q_{I_{v_1}}(s)) \forall(v_2 \in \{1, \ldots, k\} \setminus v_1) \forall(A_2 \in F\ddot{A}q_{I_{v_2}}(s)) (\exists(t \in T)((t_{v_1} \in A_1) \wedge (t_{v_2} \in A_2))).$$

Ein solcher kombinatorischer Test auf Basis einer funktionalen Äquivalenzklasse verlangt schon bei wenigen Parameterbereichen und funktionalen Äquivalenzklassen eine immense Menge an dynamischen Testfällen: Ihre Zahl ergibt sich als Produkt der Zahl der funktionalen Äquivalenzklassen der Eingabeparameterbereiche. Dieses exponentielle Wachstum, häufig als kombinatorische Explosion bezeichnet, ist Grund dafür, dass sich der Ansatz als nicht praktikabel erwiesen hat.

In einigen verfügbaren Werkzeugen besteht die Möglichkeit, die kombinierte Prüfung gewisser funktionaler Äquivalenzklassen explizit zu fordern. Der Tester kann zu diesem Zweck logische Verknüpfungen zwischen den Partitionen angeben, die sinnvolle und notwendige Kombinationen von funktionalen Partitionen charakterisieren [LehWeg00]. Dieses Vorgehen überlässt die Entscheidung über den Umfang der dynamischen Prüfung allein dem Tester und unterstützt ihn bei der Organisation.

Um systematisch, aber mit vertretbarem Aufwand die Prüfung der Kombination von funktionalen Äquivalenzklassen verschiedener Eingabeparameterbereiche zu unterstützen, wurden Ansätze zur gezielten Auswahl von Parameterkombinationen entwickelt. In [BroProPha92] und [BerYuh93] wurden beispielsweise Heuristiken wie die Herleitung und Verwendung orthogonaler Parametervektoren zur Auswahl von Parameterkombinationen für den dynamischen Test vorgeschlagen. Die Effizienz dieser Verfahren bei der Fehlerfindung konnte allerdings nicht nachgewiesen werden. Howden schlägt vor, lediglich die Parameter im dynamischen Test miteinander zu kombinieren, die in einer gewissen funktionalen Abhängigkeit zueinander stehen [How80]. Schröder und Korel systematisieren diesen Vorschlag, indem sie zusätzliche Informationen über die Abhängigkeit der Parameterbereiche nutzen [SchKor00]. Um das resultierende Verfahren notieren zu können, wird der folgende, spezifikationsbasierte Begriff der Abhängigkeit zwischen Ein- und Ausgabeparameterbereichen definiert.

Definition 4-7: Abhängigkeit zwischen Ein- und Ausgabeparameterbereichen

Vorausgesetzt wird die Zusammensetzung

$I = I_1 \times I_2 \times \ldots \times I_k \quad (k \in \aleph)$
$O = O_1 \times O_2 \times \ldots \times O_l \quad (l \in \aleph)$

des Ein- beziehungsweise Ausgabebereichs durch einzelne Parameterbereiche entsprechend Definition 3-1.

Ein Ausgabeparameterbereich O_w mit $(1 \leq w \leq l)$ wird als abhängig von einem Eingabeparameterbereich I_v mit $(1 \leq v \leq k)$ bezeichnet, wenn

$Abh_s(I_v, O_w) := (\exists(i \in I)\ \exists(j \in I)\ (\forall(r \in \{1, ..., k\} \setminus \{v\})\ (i_r = j_r)) \wedge (W^{O_w, s, i} \neq W^{O_w, s, j}))$

gilt. Die Abhängigkeit eines Ausgabeparameterbereichs wird im Folgenden mit Hilfe der Menge

$Abh_{s, w} := \{ v \in \aleph : (1 \leq v \leq k) \wedge Abh_s(I_v, O_w) \}$

dargestellt.

Ein- und Ausgabeparameterbereich werden damit als abhängig bezeichnet, wenn die Variation des Eingabedatums über dem Eingabeparameterbereich bei gleichzeitigem Festhalten der übrigen Eingabeparameter zu einer Änderung des spezifizierten Ergebnisses über dem Ausgabeparameterbereich führt. Eine Prüfung der Kombination aller funktionalen Äquivalenzklassen zweier Eingabeparameterbereiche wird in [SchKor00] nur dann gefordert, wenn es einen Ausgabeparameter gibt, der von beiden Eingabeparametern beeinflusst wird. Das Verfahren kann wie folgt notiert werden.

Verfahren 4-4: Reduzierter kombinatorischer Test der funktionalen Äquivalenzklassen nach Schröder und Korel

(a) Voraussetzungen:

Der Eingabebereich I und Ausgabebereich O können als kartesische Produkte

$I = I_1 \times I_2 \times \ldots \times I_k \qquad (k \in \aleph)$

$O = O_1 \times O_2 \times \ldots \times O_l \qquad (l \in \aleph)$

von Parameterbereichen dargestellt werden.

Für die Eingabeparameterbereiche I_v mit $(1 \leq v \leq k)$ kann durch

$$FÄq_{I_v} : \Sigma_{I, O} \rightarrow \wp^{fin}(\wp(I_v)),\quad s \mapsto FÄq_{I_v}(s)$$

ein System funktionaler Äquivalenzklassen über den Parameterbereichen aus der Spezifikation abgeleitet werden.

(b) Der reduzierte kombinatorische Test der funktionalen Äquivalenzklassen nach Schröder und Korel wird beschrieben durch

$$X^{rkomb} : \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I)),\quad s \mapsto \{ T \in \wp^{fin}(I) : P_{s, 1}(T) \wedge P_{s, 2}(T) \}$$

mit

$$P_{s, 1}(T) := \forall(w \in \{1, \ldots, l\})\ \forall(v_1 \in Abh_{s, w})\ \forall(v_2 \in Abh_{s, w} \setminus \{v_1\})\ (\forall(A_1 \in FÄq_{I_{v_1}}(s))\ \forall(A_2 \in FÄq_{I_{v_2}}(s))\ \exists(t \in T)\ ((t_{v_1} \in A_1) \wedge (t_{v_2} \in A_2)))$$

$$P_{s, 2}(T) := \forall(v \in \{1, \ldots, k\})\ \forall(A \in FÄq_{I_v}(s))\ \exists(t \in T)\ (t_v \in A).$$

Schröder und Korel zeigen an Beispielen, dass die Anzahl der geforderten Parameterkombinationen durch die Entkopplung von Parameterbereichen ohne gemeinsamen Einfluss deutlich sinken kann. Dennoch muss berücksichtigt werden, dass sich die Gesamtzahl der geforderten Parameterkombinationen immer noch als Produkt der Zahl der funktionalen Äquivalenzklassen der Parameter ergibt, die in gemeinsamer Abhängigkeit zu einem Ausgabeparameterbereich stehen.

Eine weitere Möglichkeit zur Reduktion der Anzahl der geforderten Parameterkombinationen besteht in der Betrachtung der Gültigkeit der funktionalen Äquivalenzklassen, die Schröder und Korel nicht betrachten. Um diesem Aspekt zusätzlich Rechnung zu tragen und Testfälle mit Maskierung der Ausgabe durch Fehlermeldungen zu vermeiden, kann analog zu den Forderungen in Verfahren 4-2 im Prädikat $P_{s,1}$ die Gültigkeit der kombinierten Eingabeparameter gefordert werden. Die einfache Prüfung ungültiger funktionale Äquivalenzklassen des Eingabebereichs wird durch Prädikat $P_{s,2}$ sichergestellt. Zusätzlich kann durch Kombination mit dem Verfahren 4-2 eine Prüfung von funktionalen Äquivalenzklassen des Ausgabebereichs gefordert werden.

Zur Ermittlung der Abhängigkeiten schlagen Schröder und Korel verschiedene Techniken vor. Ein spezifikationsbasierter Ansatz entsprechend Definition 4-7 führt zum funktionsorientierten Verfahren 4-4. Selten wird jedoch die Spezifikation in einer Form vorliegen, die eine entsprechende Analyse gestattet. Schröder und Korel weisen darauf hin, dass eine Spezifikation in Form eines Ursache-Wirkungs-Graphen, wie in Abschnitt 4.2.2 eingeführt, geeignet sein kann, um die Abhängigkeiten zwischen den Parameterbereichen zu ermitteln. Logische Verknüpfungen, die in einem Ursache-Wirkungs-Graphen angegeben sind, werden durch das Verfahren jedoch nicht systematisch analysiert. Aus diesem Grund sind in diesem Fall die Verfahren des Abschnitts 4.2.2 vorzuziehen. Verfahren 4-4 ist daher insbesondere dann geeignet, wenn die Art der Verknüpfung allgemeiner Natur ist und nicht mit logischen Ausdrücken beschrieben werden kann.

Darüber hinaus schlagen Schröder und Korel die Ermittlung der Abhängigkeiten zwischen den Parameterbereichen anhand der Realisierung im Prüfling vor. Hierzu kann eine statische Analyse des Daten- und Kontrollflusses des Prüflings oder eine dynamische Analyse zur Laufzeit durchgeführt werden. Hierzu ist zu bemerken, dass eine zusätzliche Berücksichtigung der durch die Spezifikation geforderten Abhängigkeiten immer sinnvoll ist, da eine Realisierung der Abhängigkeit im Prüfling möglicherweise vergessen wurde.

4.1.4 Grenzwertanalyse

Zur Wahl konkreter Repräsentanten einer funktionalen Äquivalenzklasse wird oft das Prinzip der Grenzwertanalyse empfohlen [Mye79]. Es beruht auf der Erfahrung, dass bei der Auswahl dynamischer Testfälle, die insbesondere die Grenzen der funktionalen Äquivalenzklassen überprüfen, die Wahrscheinlichkeit der Fehlerfindung höher ist als bei einer zufälligen Auswahl der Parameter. Die Vorstellung, dass gewisse Elemente einer funktionalen Äquivalenzklasse im dynamischen Test anderen vorzuziehen sind, scheint auf den ersten Blick dem Grundprinzip zu widersprechen, das eine gleichartige Behandlung funktional äquivalenter Elemente voraussetzt. Sie beruht auf der Erwartung von Fehlern im Prüfling, die speziell an den „Rändern“ der funktionalen Äquivalenzklassen Abweichungen von der Spezifikation verursachen. Die Grenzwertanalyse ist damit eine fehlerorientierte Testtechnik und hat nicht allein funktionalen Charakter. Da sich die konkrete Fehlererwartung jedoch allein auf den Inhalt der Spezifikation stützt, erfüllt sie die hier zugrunde gelegte Definition 4-1 eines funktionsorientierten Testverfahrens.

Um den Begriff des „Randes“ einer funktionalen Äquivalenzklasse mit möglichst wenigen zusätzlichen Annahmen zu definieren, werden die zugrunde liegenden Ein- und Ausgabeparameterbereiche im Folgenden zunächst als topologische Räume angenommen. Die folgende Definition führt die für die Verwendung topologischer Räume notwendigen Definitionen in Anlehnung an [ReiSoe01] ein.

Definition 4-8: Topologischer Raum, offene Menge, Punkt, Umgebung eines Punktes, Rand einer Menge

Eine Menge M und ein Teilmengensystem $Off_M \subseteq \wp(M)$ ihrer Potenzmenge werden als topologischer Raum (M, Off_M) bezeichnet, wenn die Eigenschaften

(a1) $\emptyset \in Off_M,\ M \in Off_M$

(a2) $(A_1 \in Off_M \wedge A_2 \in Off_M) \Rightarrow A_1 \cap A_2 \in Off_M$

(a3) $Off'_M \subseteq Off_M \Rightarrow \bigcup_{A \in Off'_M} A \in Off_M$

erfüllt sind. Die Elemente von M werden als Punkte bezeichnet, die Elemente von Off_M als offene Mengen.

Eine Menge $U \subseteq M$ wird als Umgebung eines Punktes $m \in M$ bezeichnet, wenn es ein $A \in Off_M$ mit $m \in A \subseteq U$ gibt. Die Menge $Umg_{M, Off_M}(m)$ aller Umgebungen von m in (M, Off_M) wird als Umgebungssystem von m bezeichnet.

Ein Punkt $m \in M$ wird genau dann als Randpunkt einer Menge $A \subseteq M$ bezeichnet, wenn gilt

$\forall\ (U \in Umg_{M, Off_M}(m))\ ((U \cap A \neq \emptyset) \wedge (U \cap (M \setminus A) \neq \emptyset))$.

Die Menge aller Randpunkte zu A wird Rand der Menge A genannt und im Folgenden mit Rand(A) bezeichnet. Zu beachten ist, dass ein Randpunkt m einer Menge A nicht zur Menge selbst gehören muss.

Darauf aufbauend kann das Verfahren der Grenzwertanalyse auf Basis einer funktionalen Äquivalenzklassenbildung wie folgt definiert werden.

Verfahren 4-5: Grenzwertanalyse auf Basis einer funktionalen Äquivalenzklassenbildung

(a) Voraussetzungen:

Der Eingabebereich I und Ausgabebereich O können als kartesische Produkte

$I = I_1 \times I_2 \times \ldots \times I_k \qquad (k \in \aleph)$

$O = O_1 \times O_2 \times \ldots \times O_l \qquad (l \in \aleph)$

von Parameterbereichen dargestellt werden. Die Parameterbereiche können als topologische Räume

$(I_v,\ Off_{I_v}) \qquad (1 \leq v \leq k)$

$(O_w,\ Off_{O_w}) \qquad (1 \leq w \leq l)$

interpretiert werden.

Für die Eingabeparameterbereiche I_v $(1 \leq v \leq k)$ und Ausgabeparameterbereiche O_w $(1 \leq w \leq l)$ können durch

$$FÄq_{I_v} : \Sigma_{I,O} \rightarrow \wp^{fin}(\wp(I_v)),\quad s \mapsto FÄq_{I_v}(s)$$

und

$$FÄq_{O_w} : \Sigma_{I,O} \rightarrow \wp^{fin}(\wp(O_w)),\quad s \mapsto FÄq_{O_w}(s)$$

Systeme funktionaler Äquivalenzklassen über den Parameterbereichen aus der Spezifikation abgeleitet werden.

Die Ränder der funktionalen Äquivalenzklassen besitzen endliche Mächtigkeit.

(b) Verfahrensbeschreibung:

Der dynamische Test der Grenzwertanalyse zur funktionalen Äquivalenzklassenbildung wird beschrieben durch

$$X^{grenz}: \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I) : P_{s,1}(T) \wedge P_{s,2}(T) \wedge P_{s,3}(T) \wedge P_{s,4}(T) \}$$

mit

$$P_{s,1}(T) := \forall(v \in \{1, \ldots, k\}) \forall(A \in F\ddot{A}q_{I_v}(s)) \exists(t \in T)((t_v \in A) \wedge (\forall(x \in \{1, \ldots, k\} \setminus \{v\})\ gültig_{s,I_x}(t_x))))$$

$$P_{s,2}(T) := \forall(w \in \{1, \ldots, l\}) \forall(A \in F\ddot{A}q_{O_w}(s)) (Gültig_{s,O_w}(A) \Rightarrow (\exists(t \in T) \forall(o \in W^{O,s,t})(o_w \in A)))$$

$$P_{s,3}(T) := \forall(v \in \{1, \ldots, k\}) \forall(A \in F\ddot{A}q_{I_v}(s)) \forall(a \in Rand(A)) \exists(t \in T)((t_v = a) \wedge (\forall(x \in \{1, \ldots, k\} \setminus \{v\})\ gültig_{s,I_x}(t_x))))$$

$$P_{s,4}(T) := \forall(w \in \{1, \ldots, l\}) \forall(A \in F\ddot{A}q_{O_w}(s)) \forall(a \in Rand(A)) (Gültig_{s,O_w}(A) \Rightarrow (\exists(t \in T)(W^{O,s,t} = \{a\}))).$$

Das Verfahren der Grenzwertanalyse auf Basis einer funktionalen Äquivalenzklassenbildung zielt darauf ab, das Verhalten des Prüflings an den Rändern der Äquivalenzklassen im Rahmen des dynamischen Tests möglichst vollständig zu prüfen. So wird in den Prädikaten $P_{s,3}$ und $P_{s,4}$ die Prüfung eines jeden im Rand der funktionalen Äquivalenzklasse enthaltenen Parameters gefordert. Die strenge Formulierung des Prädikates $P_{s,4}$ bewirkt eine sichere Prüfung der Ränder der gültigen funktionalen Ausgabeäquivalenzklassen, sie kann in Abhängigkeit von der relationalen Gestaltung der Spezifikation dazu führen, dass das resultierende Verfahren nicht anwendbar ist. Dies kann im Einzelfall eine Abschwächung des Prädikats notwendig machen.

Die zur Anwendung von Verfahren 4-5 notwendigen dynamischen Testläufe werden je nach Parameterbereich und Gestalt der funktionalen Äquivalenzklasse unterschiedlichen Umfang haben. Um die Annahme 3-3 und damit die Durchführbarkeit des dynamischen Testverfahrens in endlicher Zeit sicherzustellen, wird vorausgesetzt, dass die Ränder der funktionalen Äquivalenzklassen endliche Mächtigkeit besitzen. Dies ist beispielsweise der Fall, wenn der Parameterbereich den reellen Zahlen entspricht und die funktionalen Äquivalenzklassen durch Vereinigung endlich vieler Intervalle beschrieben sind. Zusätzlich zur Überprüfung der Ränder ist die Prüfung der funktionalen Äquivalenzklassen selbst durch die Prädikate $P_{s,1}(T)$ und $P_{s,2}(T)$ sicherzustellen. Diese sind bei der Prüfung der Ränder nicht zwingend mit eingeschlossen, da nach der obigen topologischen Definition die Elemente des Randes einer Menge nicht Elemente der Menge selbst sein müssen.

Entsprechend dem Verfahren der Grenzwertanalyse auf Basis einer funktionalen Äquivalenzklassenbildung kann das Verfahren der Grenzwertanalyse auch auf eine beliebige funktionale Partitionierung des Eingabebereichs bezogen werden. Eine Ergänzung der Grenzwertanalyse einer funktionalen Partitionierung des Ausgabebereichs kann analog ergänzt werden, wobei Elemente, die nicht im Bild der Spezifikation liegen, explizit ausgeschlossen werden müssen.

Verfahren 4-6: Grenzwertanalyse für eine funktionale Partitionierung des Eingabebereichs

(a) Voraussetzung:

Der Eingabeparameterbereich kann als topologischer Raum (I, Off_I) interpretiert werden.

Gegeben sei eine Zuordnung

$$\begin{array}{lllll} Fpart_I: & \Sigma_{I,O} & \rightarrow & \wp^{fin}(\wp(I)) \\ & s & \mapsto & Fpart_I(s), \end{array}$$

die einer beliebigen Spezifikation eine funktionale Partitionierung des Eingabebereichs zuordnet. Die Ränder der funktionalen Partitionen besitzen endliche Mächtigkeit.

(b) Verfahrensbeschreibung:

Das Verfahren der Grenzwertanalyse für eine funktionale Partitionierung des Eingabebereichs wird beschrieben durch

$$\begin{array}{lllll} X^{pgr}: & \Sigma_{I,O} & \rightarrow & \wp(\wp^{fin}(I)) \\ & s & \mapsto & \{ T \in \wp^{fin}(I): P_{s,1}(T) \wedge P_{s,2}(T) \} \end{array}$$

mit

$$P_{s,1}(T) := \forall(A \in Fpart_I(s)) \exists(t \in T) (t \in A)$$

$$P_{s,2}(T) := \forall(A \in Fpart_I(s)) \forall(a \in Rand(A)) \exists(t \in T) (t = a).$$

Die obigen Verfahren geben durch die Verwendung der topologischen Definition des Randes eine sehr allgemeine Beschreibung der Grenzwertanalyse. Diese hat den Vorteil, dass durch die freie Wählbarkeit der Topologie Flexibilität bei der Anpassung des Verfahrens an spezielle Parameterbereiche gegeben ist. So bietet sich für numerische Parameterbereiche die Wahl metrischer Räume mit euklidischer Abstandmetrik an. Dennoch ist der Begriff der Topologie abstrakt und die Wahl einer geeigneten Topologie nicht immer einfach. Für eine praxisnähere Definition ist es sinnvoll, sich von der obigen allgemeinen Beschreibung zu lösen und die Verfahrensdefinition im Hinblick auf spezielle Klassen von Parametern zu formulieren. Dies ermöglicht nicht nur eine vereinfachte, besser anwendbare Beschreibung des Verfahrens, sondern auch eine situationsabhängige Erweiterung des Grenzwertbegriffs. Die Notwendigkeit hierzu zeigen die Empfehlungen von Myers: Zur Prüfung der Äquivalenzklasse 1 – 255 eines ganzzahligen Parameters schlägt Myers die Wahl von Testdatensätzen mit den Werten 0, 1, 255 und 256 für den Parameter vor. Für einen Parameter, dessen Datentyp Nachkommastellen zulässt und für den eine Äquivalenzklasse -1.0 – +1.0 vorgegeben ist, rät er zur Belegung mit den Werten -1.0, 1.0, -1.001 und 1.001. Um diese Empfehlungen mit der obigen Verfahrensdefinition in Einklang zu bringen, ist ein erweiterter Grenzwertbegriff im Hinblick auf spezielle Parameterbereiche zu definieren.

Betrachtet man numerische Parameterbereiche in herkömmlichen Rechnersystemen, so können diese im Allgemeinen als endliche Teilmenge der reellen Zahlen $\Re$ dargestellt werden. Die Relation $\leq$ bildet auf diesen Parameterbereichen eine konnexe Ordnungsrelationen, so dass der Vergleich von zwei beliebigen Elementen stets möglich ist. Der Parameterbereich ist damit endlich und vollständig geordnet, so dass sich für jedes Element das nächst größere und das nächst kleinere bestimmen lassen, falls das Element nicht selbst größtes beziehungsweise kleinstes Element des Parameterbereichs ist. Somit können als Grenzwerte einer funktionalen Äquivalenzklasse diejenigen Elemente des Parameterbereichs bezeichnet werden, die selbst Element der funktionalen Äquivalenzklasse sind, deren nächst kleineres oder größeres Element jedoch nicht in der funktionalen Äquivalenzklasse enthalten

ist. Weiterhin werden die Elemente des Komplements der funktionalen Äquivalenzklasse im Parameterbereich als Grenzwert bezeichnet, zu denen das nächst kleinere oder größere Element in der funktionalen Äquivalenzklasse enthalten ist.

Ein weiterer Ansatz zur Definition der Grenzwerte einer funktionalen Äquivalenzklasse über einem numerischen Parameterbereich, der als endliche Teilmenge der reellen Zahlen $\Re$ mit der üblichen Abstandmetrik versehen ist, liefert die Bestimmung von ε-Umgebungen um die Randpunkte. Als ε-Umgebung zu einem Element p eines metrischen Raumes wird eine Teilmenge des Raumes bezeichnet, die alle Elemente mit Abstand von p kleiner als ε enthält. Für ε ist ein Wert zu wählen, der klein ist, jedoch größer als der maximale Abstand zwischen zwei benachbarten Parametern. Damit ist sichergestellt, dass die ε-Umgebung eines beliebigen Elementes nicht leer ist. In einem Verfahren zur Prüfung der Grenzwerte einer funktionalen Äquivalenzklasse kann nun gefordert werden, alle Elemente im dynamischen Test zu prüfen, deren ε-Umgebung sowohl Elemente der funktionalen Äquivalenzklasse als auch Elemente ihres Komplements enthält.

Neben dem gängigen Fall eines numerischen Parameterbereichs, der aus einer Teilmenge der reellen Zahlen besteht, können auch mehrdimensionale numerische Parameterbereiche im Rahmen der Grenzwertanalyse betrachtet werden. Wie bereits für das Verfahren des dynamischen Tests auf Basis einer funktionalen Äquivalenzklassenanalyse angemerkt, kann beispielsweise eine endliche Teilmenge des $\Re^n$ angenommen werden. Die Anzahl der nach den oben skizzierten Verfahren zu testenden Punkte kann über diesem Parameterbereich groß und das Verfahren 4-5 nicht praktisch durchführbar sein. Falls für die Gestalt der Ränder einer funktionalen Partition zusätzliche Annahmen gemacht werden können, kann die Prüfung der Grenzwerte daran orientiert und spezielle Elemente zur Prüfung ausgewählt werden. Ein Beispiel sind lineare Bereichsgrenzen, die in der Form $c_1b_1 + c_2b_2 +... + c_kb_k$ ROP c_{k+1} mit den reellen Parametern $c_1, c_2, ..., c_{k+1}$, den Basiselementen $b_1, b_2, ..., b_k$ des Parameterbereichs und einem relationalen Operator ROP der Gestalt =, ≠, ≥, >, ≤, < beschrieben werden können. Hier kann die Auswahl der Testdaten in Anlehnung an ein in [WhiCoh80] vorgeschlagenes Verfahren stattfinden: Um die Realisierung solcher Bereichsgrenzen durch den Prüfling zu testen, sind – soweit vorhanden – k linear unabhängige Datenpunkte aus dem Schnitt der durch $c_1b_1 + c_2b_2 +... + c_kb_k = c_{k+1}$ aufgespannten Hyperebene mit dem Parameterbereich zu wählen. Zusätzlich ist bei Bereichsgrenzen mit den relationalen Operatoren ≤, ≥, <, > ein weiteres Eingabedatum auf der offenen Seite der Bereichsgrenze zu wählen. Für die relationalen Operatoren =, ≠ werden zusätzlich Testdaten auf beiden Seiten der Hyperebene gefordert, deren Projektion in die konvexe Hülle des durch die Testdaten in der Hyperebene beschriebenen Polyeders fällt. Ihr Abstand zur Bereichsgrenze ist möglichst gering zu wählen. Die so gewählten Testdaten eignen sich zur Prüfung der Lage einer Hyperebene und zur Prüfung ihrer Ausrichtung als abgeschlossene, einseitig offene oder beidseitig offene Bereichsgrenze.

Diese Wahl der Testdaten wird in [WhiCoh80] zur Prüfung von linear begrenzten Teilmengen des Datenbereichs verwendet, die sich aus einer strukturorientierten Pfadbetrachtung ergeben. Es wurde entwickelt, um systematisch die Realisierung der Grenzen der Pfadbereiche über den Eingabedaten zu untersuchen und kann als Pfadbereichstest bezeichnet werden. Da im strukturorientierten Test die Linearität der Bereichgrenzen durch Analyse des Prüflings sichergestellt werden kann, hat das Verfahren eine sehr hohe Aussagekraft [WhiCoh80]. Bei der hier vorgeschlagenen Anwendung des Verfahrens im Rahmen einer spezifikationsbasierten Grenzwertanalyse ist die Aussagekraft jedoch wesentlich geringer: So kann im spezifikationsbasierten Vorgehen nicht vorausgesetzt werden, dass die im Prüfling realisierten Grenzen der „funktional äquivalenten“ Behandlung tatsächlich linear sind. Daher können aus dem Bestehen der dynamischen Testfälle keine Rückschlüsse auf die Behandlung weite-

rer Eingabedaten durch den Prüfling gezogen werden können. Dies illustriert Abbildung 4-1: Hier wird über einem zweidimensionalen Parameterbereich eine linear begrenzte funktionale Partition gezeigt, die schraffiert dargestellt ist. Die Zugehörigkeit der Grenze zur funktionalen Partition beziehungsweise zu ihrem Komplement ist durch Pfeile angedeutet. Die für die Partition spezifizierte Funktionalität sei durch den Prüfling im Bereich außerhalb der gestrichelt markierten Parabel realisiert. Werden die Testdaten t_1, t_2 und t_3 gemäß dem oben beschriebenen Verfahren wie im der Graphik eingetragen gewählt, wird die fehlerhafte Realisierung nicht bemerkt.

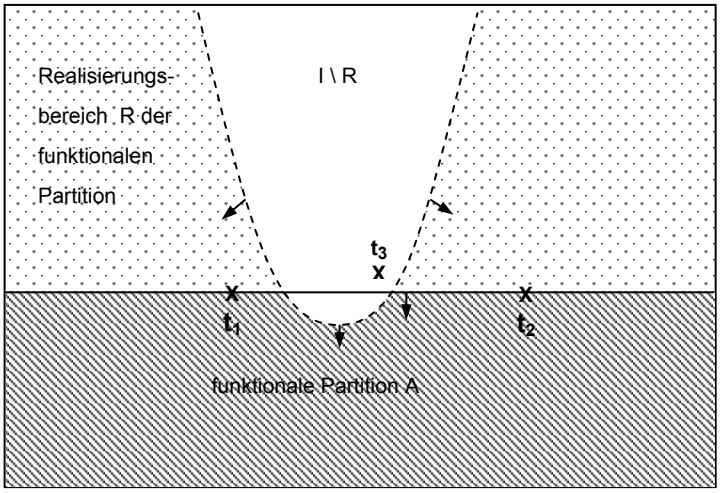


Abbildung 4-1: Nichtentdeckung der fehlerhaften Realisierung einer funktionalen Partition im spezifikationsorientierten Bereichstest

Kann eine lineare Realisierung der Grenzen der Bereiche mit funktional äquivalenter Behandlung der Eingabedaten vorausgesetzt beziehungsweise durch Analyse des Prüflings sichergestellt werden, ergibt sich im spezifikationsorientierten Bereichstest eine entsprechende Genauigkeit bei der Entdeckung fehlerhafter Realisierungen wie beim Verfahren von White und Cohen.

Auch für Textparameter und Aufzählungsbereiche, Listen und Dateien sind Grenzwertbetrachtungen möglich. So kann die lexikalische Reihenfolge oder die Längenvorgabe für einen Textparameter zur Bestimmung von Grenzwerten genutzt werden. In [WeyJen91] wird ein Abstandsmaß für Textparameter vorgeschlagen, das auf der Anzahl der abweichenden Zeichen beruht und ebenfalls eine Grenzwertermittlung ermöglicht. Entsprechend können Dateien, Tabellen oder Listen betrachtet werden. Hier fordert Myers die besondere Betrachtung ihrer Länge sowie des ersten und letzten Elements. Auch für Aufzählungsbereiche sind Grenzwertbetrachtungen denkbar: So kann etwa in einem Musikalienhandel zwischen Holz- und Blechbläsern unterschieden werden. Als Grenzfall zwischen diesen Instrumentenklassen ist die Querflöte anzusehen, die zwar vollständig aus Blech ist, aber dennoch zu den Holzbläsern zählt. Wird in einer Spezifikation die Unterscheidung der beiden Instrumentenklassen gefordert, ist die Wahl eines dynamischen Testfalls sinnvoll, der die Behandlung der Querflöte durch den Prüfling untersucht, da hier die Wahrscheinlichkeit eines Irrtums bei der Realisierung des Prüflings hoch ist. Problematisch sind Grenzwertbetrachtungen für strukturierte Datentypen, die mehrere Basistypen kapseln. Dies gilt entsprechend für die Grenzwertbetrachtung von Parameterbereichen vom Typ einer Klasse. Hier muss der Grenzwertbegriff auf die Belegung einzelner Basistypen oder Attribute übertragen werden und vor dem Hintergrund der intendierten Semantik des Parameters betrachtet werden.

4.1.5 Mögliche Erweiterungen der Verfahren

In den vorangegangenen Abschnitten wurden Verfahren des dynamischen Tests diskutiert, die auf Basis einer funktionalen Partitionierung oder Äquivalenzklasseneinteilung der Datenbereiche operieren. Die Verfahren wurden in ihrer ursprünglichen und verbreiteten Form eingeführt und mit Hilfe der Terminologie aus Kapitel 3 formalisiert. Auf dieser Basis ergeben sich die folgenden Möglichkeiten zur Erweiterung der Verfahren.

4.1.5.1 Erweiterung im Hinblick auf die Performanz

Die in den vorangegangenen Abschnitten beschriebenen dynamischen Verfahren beruhen auf der Einteilung des Ein- oder Ausgabebereichs nach funktionalen Aspekten. Anforderungen an die Performanz des Prüflings spielen bei der Festlegung der Einteilung und bei der Auswahl der Testfälle keine Rolle. Lediglich bei der Durchführung und Auswertung der Testfälle entsprechend Definition 3-6 wird die Einhaltung der spezifizierten Performanz geprüft. Diese Prüfung der Performanz ist nicht systematisch, eine systematische Analyse der Performanz des Prüflings bei verschiedenen Eingabekonstellationen unterbleibt. Es kann jedoch durchaus sinnvoll sein, bei der spezifikationsbasierten Auswahl dynamischer Testfälle zur Prüfung der Performanzanforderungen die in diesem Abschnitt beschriebenen Verfahren anzuwenden. So kann es zielführend sein, den Zeitbereich in ein System funktionaler Äquivalenzklassen aufzuteilen, das beispielsweise einen Hochperformanzbereich, einen Bereich mit mäßigen Performanzanforderungen und einen Bereich mit untergeordneten Performanzanforderungen unterscheidet. Der dynamische Test auf Basis der funktionalen Äquivalenzanalyse kann bei entsprechender Modifikation sicherstellen, dass für jeden dieser Bereiche wenigstens ein dynamischer Testfall vorzusehen ist. Im Rahmen eines dynamischen Tests auf Basis der Grenzwertanalyse ist die Erreichung der Randbereiche dieser Performanzkriterien anzustreben. Auch im Hinblick auf die Auswahl der Kombinationen von funktionalen Äquivalenzklassen über dem Eingabebereich kann die Performanz wichtige Hinweise geben. So kann es sinnvoll sein, die in Definition 4-7 eingeführte Abhängigkeitsanalyse auf die Performanz auszudehnen. Diese kann entsprechend den Ausgabeparametern Hinweise auf Abhängigkeiten der Eingabeparameterbereiche geben.

4.1.5.2 Kombination mit dem Bereichstest

Bei der Vorstellung der Grenzwertanalyse für mehrdimensionale numerische Parameterbereiche wurde in Abschnitt 4.1.4 eine Prüfung stückweise linearer Bereichsgrenzen mit einer Teststrategie angeregt, die sich an dem in [WhiCoh80] veröffentlichten und in [ClaHasRic82, WeyJen91] erweiterten Bereichstest orientiert. Anhand der Diskussion einer Situation, in der im dynamischen Test eine fehlerhafte Realisierung durch das Verfahren möglicherweise nicht entdeckt wird, wurde deutlich, dass im rein spezifikationsbasierten Verfahren Annahmen über die Realisierung gemacht werden müssen: Hier muss sichergestellt werden, dass die Grenzen im Prüfling tatsächlich in linearer Form implementiert sind. Ebenso wird in der ursprünglichen, am Prüfling orientierten Formulierung des Verfahrens eine lineare Formulierung der Bereichgrenzen in der Spezifikation vorausgesetzt. Nur unter diesen Voraussetzungen kann wie in [WhiCoh80] gezeigt werden, dass eine Verschiebung oder fehlerhafte Umsetzung von Bereichsgrenzen im dynamischen Test durch das Verfahren entdeckt wird.

Diese Überlegung legt nahe, das Verfahren des dynamischen Tests auf Basis einer funktionalen Partitionierung des Eingabebereichs und die darauf aufbauende spezifikationsorientierte Grenzwertuntersuchung mit dem strukturorientierten Pfadbereichstest entsprechend Definition 3-12 zu kombinieren.

Durch das resultierende Verfahren werden die Grenzen der statisch ermittelten Pfadbereiche und die der funktionalen Partitionen gleichermaßen geprüft. So besteht die Möglichkeit, eine Übereinstimmung der funktionalen Partitionen und ihrer Ränder mit den im Prüfling implementierten Pfadbereichen zu prüfen und gleichzeitig die Übereinstimmung der geforderten und der implementierten Funktionalität über den Bereichen abzusichern. Abweichungen zwischen spezifizierten und realisierten Grenzen können unter den in [WhiCoh80] zugrunde gelegten, allerdings sehr restriktiven Annahmen zuverlässig entdeckt werden. Sie entsprechen den in [How76] eingeführten und [WhiCoh80] untersuchten Bereichsfehlern. Gleichzeitig werden durch die Spezifikation geforderte funktionale Partitionen identifiziert, die im Prüfling nicht in einen Pfad umgesetzt wurden. Ebenso werden Pfadbereichsgrenzen, die keine Entsprechung in der Aufteilung der funktionalen Partitionen haben, auf Konformität mit der Spezifikation geprüft.

In [WhiCoh80] wird die Annahme zugrunde gelegt, dass eine zufällige Übereinstimmung des Ergebnisses eines dynamischen Testfalls mit dem spezifizierten Ergebnis („coincidental correctness“) auszuschließen ist, falls der dynamische Testfall auf einem anderen Pfad als dem intendierten behandelt wurde. Weitet man diese Annahme so aus, dass man von der Übereinstimmung des Testergebnisses mit der Spezifikation nicht nur auf das Durchlaufen des korrekten Pfades, sondern auch auf die Anwendung einer korrekt implementierten Funktionalität schließen kann, so folgt aus dem Bestehen des kombinierten dynamischen Testverfahrens die Korrektheit des Prüflings. Auch unter schwächeren und realitätsnäheren Voraussetzungen hat das kombinierte Verfahren eine starke Aussagekraft: So kann aus der alleinigen Verwendung von Operatoren im Prüfling, die vorgegebenen, dem diskreten Parameterraum angepassten Stetigkeitsbedingungen genügen, auf eine entsprechende Stetigkeit der durch den Prüfling über einem Pfadbereich realisierten Funktionalität geschlossen werden. Eine solche Stetigkeitsbedingung unterstützt die Ableitung von Aussagen über das Verhalten des Prüflings auch für nicht getestete Eingabedaten des Pfadbereichs. Falls anhand der Quelltextes der Prüflings auch auf den Typ der über dem Bereich realisierten Funktionalität geschlossen werden kann, kann durch die Wahl spezieller Testdaten das Verhalten des Prüflings über dem gesamten Bereich analysiert werden. Einfaches Beispiel ist die Berechnung einer Funktion der Gestalt $f(i_1, i_2) = c_0+c_1i_1+c_2i_2$ über einem Pfadbereich des Eingabebereichs $I = \Re^2$, deren Implementierung durch die Auswahl dreier Datenpunkte des Bereichs geprüft werden kann.

Diese Betrachtungen zeigen die Aussagekraft eines kombinierten Bereichstest auf Basis einer Pfadbereichsanalyse und einer funktionalen Partitionierung mit Grenzwertbetrachtung. Es muss jedoch beachtet werden, dass der Bereichstest auf Basis einer Pfadanalyse nur unter engen Restriktionen durchführbar ist. In [WhiCoh80] wird die Verwendung von indizierten Feldern und von Unterprogrammbeziehungsweise Funktionsaufrufen verzichtet. Auch die Verwendung von Zeigern ist problematisch. Diese Einschränkungen sind nötig, um die Bestimmung der Pfadbereiche mit Hilfe statischer Analysetechniken zu ermitteln. Sie werden in [WeyJen91] gelockert, doch bleibt die Bestimmung der Pfadbereiche eins der Hauptprobleme beim Einsatz des Verfahrens. Weiterhin ist der hohe Aufwand zu berücksichtigen, der zur Durchführung des dynamischen Testverfahrens an sich benötigt wird: So wird für jede funktionale Partition, aber auch jeden möglichen Pfad durch den Prüfling ein Bereich festgelegt, dessen Grenzen mit mehreren dynamischen Testfällen zu prüfen sind. Das hier diskutierte kombinierte dynamische Testverfahren subsumiert damit den Test aller Pfade durch den Kontrollfluss des Prüflings.

Die Idee der Kombination einer spezifikationsbasierten und einer strukturorientierten Partitionierung des Eingabebereichs ist in der Literatur bereits vorgeschlagen und umfassend diskutiert worden. Schon in [GooGer75] wird die Nutzung von Informationen aus Spezifikation und Struktur des Prüflings

im dynamischen Test empfohlen. Diese Ansätze werden in [WeyOst80] analysiert und erweitert. Hier wird eine Partitionierung des Eingabebereichs angestrebt, die Eingabedaten zusammenfasst, die alle gleichermaßen fehlerhaft oder korrekt durch den Prüfling behandelt werden. Solche Partitionen werden als fehleroffenbarende Partitionen („revealing subdomains“) bezeichnet. Sie besitzen den Vorteil, dass der Test eines einzelnen Repräsentanten einer Partition eine Aussage über die Konformität aller Eingabedaten der Partition mit der Spezifikation zulässt. Um im realen Testprozess eine annähernd fehleroffenbarende Partitionierung zu erhalten, empfehlen Weyuker und Ostrand den Schnitt der funktionalen Partitionen der Spezifikation mit strukturorientierten Pfadbereichen. Weiterhin sollen Fehler, die im Prüfling vermutet werden, bei der Partitionierung berücksichtigt werden. In [RicCla81, RicCla85] wird über geeigneten Partitionen der Eingabebereichs ein Nachweis der Konformität des Prüflings mit der Spezifikation mit Verfahren der formalen Verifikation erbracht. Das oben vorgeschlagene Verfahren der Bestimmung der im Prüfling realisierten Zuordnung über den Pfadbereichen mit Mitteln der statischen Analyse und der anschließende Vergleich mit der spezifizierten Zuordnung kann als Spezialfall dieses Vorgehens betrachtet werden.

4.2 Verfahren zur Prüfung von Ursachen und ihren Wirkungen

In den Verfahren des vorangegangenen Abschnitts werden Ein- und Ausgabebereich nach funktionalen Gesichtspunkten in Teilmengensysteme unterteilt. Diese liefern den dort beschriebenen dynamischen Testverfahren eine Grundlage für die Auswahl und Akzeptanz der dynamischen Testläufe. Ein- und Ausgabebereich werden dabei meist separat betrachtet, logische Verknüpfungen zwischen beiden ignoriert. Lediglich der in Verfahren 4-4 beschriebene reduzierte kombinatorische Test der funktionalen Äquivalenzklassen prüft, ob im Prüfling ein Bezug zwischen zwei Eingabeparametern mit Einfluss auf mindestens einen Ausgabeparameter realisiert ist. Die Art des Bezuges spielt bei der Auswahl der Testfälle keine Rolle.

Das systematische Testen von spezifizierten Ursachen und Wirkungen schließt diese Lücke: Ziel ist hier eine umfassende Prüfung der Realisierung der kausalen Zusammenhänge im Prüfling mit einer möglichst geringen Anzahl von Testfällen. Der dynamische Test zur Prüfung von Ursachen und ihren Wirkungen existiert in zwei Ausprägungen, die sich hauptsächlich in der Detaillierung der Beschreibung des Vorgehens zur Ableitung der zu prüfenden kausalen Zusammenhänge unterscheiden. Das in Abschnitt 4.2.1 beschriebene Verfahren des dynamischen Tests auf Basis von Entscheidungstabellen geht von einer weitgehend vollständigen Prüfung der kausalen Zusammenhänge aus, die schnell einen nicht mehr akzeptablen Umfang erreicht. Die Ursache-Wirkungs-Analyse, in Abschnitt 4.2.2 beschrieben, führt hingegen die Ableitung der zu prüfenden kausalen Zusammenhänge mit Hilfe eines Algorithmus durch, der eine fundierte Prüfung mit einer möglichst geringen Anzahl von dynamischen Testfällen anstrebt.

4.2.1 Entscheidungstabellen und Entscheidungsbäume

Der dynamische Test auf Basis von Entscheidungstabellen nutzt die Notation der Entscheidungstabellen zur Auswahl und Darstellung der relevanten kausalen Zusammenhänge, deren Prüfung im dynamischen Test gefordert wird. Es wurde bereits 1975 vorgeschlagen [GooGer75] und wird heute in Standardwerken zum dynamischen Test [Bei90, Bal98, Lig02] behandelt.

Die folgende Beschreibung geht von einer Spezifikation der geforderten Funktionalität als Fließtext in natürlicher Sprache aus. Hier sei das geforderte Systemverhalten durch Angabe von kausalen Zu-

sammenhängen zwischen Ursachen und ihren Wirkungen textuell beschrieben. Da das Verfahren schnell zu einer hohen Zahl von Testfällen führt und wenig Möglichkeiten für eine systematische Beschränkung auf eine Teilmenge davon bietet, ist die Spezifikation in kurze, mit dem Verfahren beherrschbare Teilabschnitte zu zerlegen. Dies entspricht dem zu Beginn des Kapitels 4 vorgestellten Ansatz, der die Durchführung des funktionsorientierten Tests einer Spezifikation als Kombination von Prüfungen einzelner funktionaler Teilaspekte beschreibt. Anschließend werden die Spezifikationsabschnitte im Hinblick auf die dort spezifizierten kausalen Zusammenhänge untersucht. Die Ursachen der kausalen Zusammenhänge geben relevante Eigenschaften der Elemente des Eingabebereichs an, die für das intendierte Systemverhalten von besonderem Interesse sind. Sie werden in semantische Einheiten des Spezifikationstextes beschrieben und können als Prädikat über dem Eingabebereich aufgefasst werden.

Definition 4-9: Ursache

Eine Ursache wird beschrieben durch eine semantische Einheit im Spezifikationstext, die als Eigenschaft der Eingabedaten interpretiert werden kann und die von Bedeutung für das spezifizierte Systemverhalten ist. Sie kann als einstelliges Prädikat

$\forall(i \in I)\ (u(i) :\Leftrightarrow$ (Element i hat die Eigenschaften der beschriebenen Ursache))

über dem Eingabebereich I aufgefasst werden.

Durch jede Ursache u wird eine Teilmenge

$U := \{ i \in I: u(i) \} \subseteq I$

des Eingabebereichs beschrieben, deren Elemente die Ursache erfüllen und die daher im Hinblick auf die Ursache eine gewisse funktionale Äquivalenz aufweisen. Es wird vorausgesetzt, dass im Rahmen des betrachteten Teilabschnitts des Spezifikationstextes endlich viele Ursachen beschrieben sind, die im Folgenden mit $u_1, u_2, \ldots, u_m$ ($m \in \aleph$) bezeichnet werden.

Ebenso wie die Ursachen werden auch die Wirkungen der kausalen Zusammenhänge durch semantische Einheiten des Spezifikationstextes beschrieben.

Definition 4-10: Wirkung

Eine Wirkung wird beschrieben durch eine semantische Einheit im Spezifikationstext, die als Eigenschaft von Ausgabedaten interpretiert werden kann und die ein Resultat des spezifizierten Systemverhaltens ist. Sie kann formal als einstelliges Prädikat

$\forall(o \in O)\ (w(o) :\Leftrightarrow$ (Element o hat die Eigenschaften der beschriebenen Wirkung))

über dem Ausgabebereich O aufgefasst werden.

Wirkungen werden in der Literatur auch als Aktionen bezeichnet [Bei90, Lig02]. Eine Wirkung w gibt durch

$W := \{ o \in O: w(o) \} \subseteq O$

eine Teilmenge von Elementen des Ausgabebereichs mit funktionaler Äquivalenz im Hinblick auf die Wirkung an. Auch für Wirkungen wird die Endlichkeit ihrer Anzahl vorausgesetzt, so dass sie im Folgenden mit $w_1, w_2, \ldots, w_n$ ($n \in \aleph$) bezeichnet werden. Oft werden beim dynamischen Test auf Basis einer Entscheidungstabelle nur eine einzige Wirkung sowie die komplexen Bedingungen ihres Zustandekommens untersucht.

Das Vorgehen im dynamischen Test auf Basis einer Entscheidungstabelle sieht nun vor, dass alle kombinatorisch möglichen, mit der Spezifikation vereinbaren Ursachenkombinationen zusammengestellt werden. Diese werden in einer Entscheidungstabelle als Bedingungen eingetragen. Aus dem Spezifikationstext werden die durch sie bedingten Wirkungen abgeleitet und in der Entscheidungstabelle als Aktionen eingetragen. Das Ergebnis wird hier als vollständige Ursache-Wirkungs-Tabelle bezeichnet und kann wie folgt notiert werden.

Definition 4-11: Ursache-Wirkungs-Tabelle, vollständige Ursache-Wirkungs-Tabelle

Zu einer Spezifikation $s \in \Sigma_{I, O}$ wird eine Matrix über der Menge { 0, 1 } mit m + n Spalten und l Zeilen ($l \in \aleph$) als Ursache-Wirkungs-Tabelle UWT_s bezeichnet, wenn sie die spezifizierten Abhängigkeiten zwischen dem Eintreten der Ursachen $u_1, u_2, \ldots, u_m$ ($m \in \aleph$) und dem der Wirkungen $w_1, w_2, \ldots, w_n$ ($n \in \aleph$) wie folgt beschreibt:

(a) Alle Ursachenkombinationen und ihre resultierenden Wirkungen werden in jeweils einer Zeile der Matrix beschrieben.

(b) Das Vorliegen einer Ursache u_i ($1 \leq i \leq m$) in einer Ursachenkombinationen wird durch den Eintrag 1 in Spalte i der zugehörigen Zeile gekennzeichnet.

(c) Eine durch die in den Spalten 1 bis m beschriebene Ursachenkombination bedingte Wirkung w_j ($1 \leq j \leq n$) wird durch den Eintrag 1 in der Spalte m + j der zugehörigen Zeile gekennzeichnet.

(d) Mit 0 gekennzeichnete Ursachen und Wirkungen liegen in der Ursachenkombination nicht vor oder können nicht evaluiert werden.

Eine Ursache-Wirkungs-Tabelle zu einer Spezifikation s wird als vollständige Ursache-Wirkungs-Tabelle $vUWT_s$ bezeichnet, wenn alle kombinatorisch möglichen und gleichzeitig mit der Spezifikation vereinbaren Ursachenkombinationen und ihre Wirkungen in der Matrix beschrieben sind.

Alternativ können Entscheidungstabellen auch spaltenweise angegeben werden, was einer Notation in Form der Transponierten der oben beschriebenen Matrix führt. Da im Rahmen einer Spezifikation nach Definition 3-2 eine Zuordnung mehrerer unterschiedlicher Ausgabedaten zu einem Eingabedatum zugelassen ist, können mehrere alternative Wirkungskombinationen zu einer Ursachenkombination möglich sein. Dies führt zu mehreren Zeilen in der vollständigen Ursache-Wirkungs-Tabelle, die bei gleicher Ursachenkombination unterschiedliche Wirkungen beschreiben.

Auch wenn die Spezifikation jeder möglichen Ursachenkombination nur eine Wirkungskombination zuordnet, wird die zugehörige vollständige Ursache-Wirkungs-Tabelle mit bis zu $l = 2^m$ Zeilen sehr umfangreich. Sie ist damit ungeeignet als Vorgabe für die im dynamischen Test zu prüfenden Ursachenkombinationen. Aus diesem Grund wird eine Reduktion der vollständigen Ursache-Wirkungs-Tabelle angestrebt, die weniger relevante Testfälle ausblendet. Dazu werden Ursachenkombinationen gesucht, die zu denselben Wirkungen führen, und die sich lediglich im Hinblick auf die Ausprägung einer einzigen Ursache unterscheiden. Solche Ursachenkombinationen werden in einer Zeile zusammengefasst. Die nicht einheitliche Ursachenbelegung wird in diesem Fall durch ein Symbol gekennzeichnet, dass in [Lig02] als „Don't Care" bezeichnet wird und mit „-" notiert werden kann. Zeilen, die das Symbol „Don't Care" bereits enthalten, können weiter in die Reduktion mit einbezogen werden. Die Reduktion der Entscheidungstabelle kann solange fortgesetzt werden, bis die relevanten kausalen

Zusammenhänge identifiziert sind und durch eine akzeptable Anzahl von dynamischen Testfällen geprüft werden können.

Definition 4-12: Reduzierte Ursache-Wirkungs-Tabelle

Zu einer vollständigen Ursache-Wirkungs-Tabelle $vUWT_s$ wird eine Matrix über der Menge { 0, 1, - } mit m + n Spalten und k Zeilen ($k \in \aleph$) als reduzierte Ursache-Wirkungs-Tabelle $rUWT_s$ bezeichnet, die spezifizierten Abhängigkeiten zwischen dem Eintreten der Ursachen u_1, u_2, ... , u_m ($m \in \aleph$) und dem der Wirkungen w_1, w_2, ... , w_n ($n \in \aleph$) wie folgt beschreibt:

(a) Alle Ursachenkombinationen und ihre resultierenden Wirkungen werden in jeweils einer Zeile der Matrix beschrieben.

(b) Das Vorliegen einer Ursache u_i ($1 \leq i \leq m$) in einer Ursachenkombinationen wird durch den Eintrag 1 in Spalte i der zugehörigen Zeile gekennzeichnet.

(c) Eine durch die in den Spalten 1 bis m beschriebene Ursachenkombination bedingte Wirkung w_j ($1 \leq j \leq n$) wird durch den Eintrag 1 in der Spalte m + j der zugehörigen Zeile gekennzeichnet.

(d) Mit 0 gekennzeichnete Ursachen und Wirkungen liegen in der Ursachenkombination nicht vor beziehungsweise treten nicht ein oder sind nicht evaluierbar.

(e) Mit „-“ gekennzeichnete Ursachen und Wirkungen sind für den kausalen Zusammenhang nicht relevant, sie können vorliegen oder nicht.

(f) Jede Zeile der vollständigen Ursache-Wirkungs-Tabelle wird durch eine Zeile der reduzierten Ursache-Wirkungs-Tabelle mit beschrieben.

In der Literatur wird die reduzierte Darstellung der Ursache-Wirkungs-Tabelle auch als optimierte Entscheidungstabelle bezeichnet [Bal98, Lig02]. Es ist möglich, die Entscheidungstabelle bei der Ableitung aus der Spezifikation direkt in reduzierter Form anzugeben. Hierbei ist das in Definition 4-11 formulierte Vollständigkeitskriterium geeignet umzusetzen. Beizer jedoch warnt vor der Verwendung von „Don’t Care“ bei der Spezifikation logischer Verknüpfungen und weist auf mögliche entstehende Lücken hin [Bei90]. So sollte die Reduktion immer im Sinne eines geeigneten Kompromisses zwischen Testvollständigkeit und Testumfang entsprechen.

Der dynamische Test auf Basis von Entscheidungstabellen fordert nun, dass alle in der reduzierten Ursache-Wirkungs-Tabelle beschriebenen kausalen Zusammenhänge durch mindestens einen dynamischen Testfall geprüft werden. Um dieses Verfahren im Kontext der Definitionen aus Abschnitt 3.2 formulieren zu können, sind die geforderten Ursachenkombinationen mit den Mitteln der Aussagenlogik zu beschreiben. Die folgende Definition führt einen entsprechenden Formalismus ein, der für die Verfahrensbeschreibung benötigt wird.

Definition 4-13: Ursachenkombinationsforderungen zu einer reduzierten Ursache-Wirkungs-Tabelle, Erfüllung einer Ursachenkombinationsforderung durch ein Eingabedatum

Ein logischer Ausdruck $\lambda_f^{u_1, u_2, ..., u_m}$ über dem Alphabet der Ursachenbezeichner, der mit den Operatoren $\wedge$ und $\neg$ und ohne Verwendung von Klammerungen aufgebaut ist und der alle Ursachenbezeichner maximal einmal enthält, wird als Ursachenkombinationsforderung f_v ($1 \le v \le k$) zu einer gegebenen reduzierten Ursache-Wirkungs-Tabelle $rUWT_s$ mit k Zeilen ($k \in \aleph$) bezeichnet, wenn

(a) jeder der in der Zeile v mit 1 gekennzeichneten Ursachenbezeichner im logischen Ausdruck enthalten und nicht negiert ist,

(b) jeder der in der Zeile v mit 0 gekennzeichneten Ursachenbezeichner im logischen Ausdruck enthalten und negiert ist und

(c) keiner der in der Zeile v mit „-“ gekennzeichneten Ursachenbezeichner im logischen Ausdruck enthalten ist.

Ein Eingabedatum $i \in I$ erfüllt eine Ursachenkombinationsforderung f_v ($1 \le v \le k$), wenn der zugehörige logische Ausdruck

$f_v(i) := \lambda_{f_v}^{u_1(i), u_2(i), ..., u_m(i)}$

bei Belegung mit den Ursachenprädikaten $u_1(i)$, …, $u_m(i)$ des Eingabedatums wahr ist.

Die aus der reduzierten Ursache-Wirkungs-Tabelle abgeleiteten Ursachenkombinationsforderungen erlauben die folgende Verfahrensbeschreibung für den dynamischen Test auf Basis von Entscheidungstabellen.

Verfahren 4-7: Dynamisches Testen auf Basis einer Entscheidungstabelle

(a) Voraussetzung:

Zur Spezifikation $s \in \Sigma_{I, O}$ können die Ursachen

$U_s = \{ u_1, u_2, ... , u_m \}$ ($m \in \aleph$),

die Wirkungen

$W_s = \{ w_1, w_2, ... , w_n \}$ ($n \in \aleph$),

die reduzierte Ursache-Wirkungs-Tabelle

$rUWT_s$

mit k Zeilen und m+n Spalten sowie eine Menge von Ursachenkombinationsforderungen

$F^{rUWT_s} = \{ f_1, f_2, ... , f_k \}$ ($k \in \aleph$)

abgeleitet werden.

(b) Verfahrensbeschreibung:

Der dynamische Test auf Basis der Entscheidungstabelle wird beschrieben durch

$$X^{UWT}: \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) \quad := \quad \forall(v \in \{ 1, \ldots, k \}) \, \exists(t \in T) \, f_v(t).$$

Die Entscheidungstabelle in Form einer reduzierten Ursache-Wirkungs-Tabelle gibt somit eine abstrakte Beschreibung der Testdaten: Sie beschreibt die Ursachenkombinationen, die im dynamischen Test zu prüfen sind und stellt somit eine Anleitung zur Testdatenselektion und ein Testdatenadäquatheitskriterium bereit. Gleichzeitig nennt sie die Wirkungen, die laut Spezifikation aus den Ursachenkombinationen resultieren. Damit ist ein Prüfkriterium für die Testdurchführung gegeben, das eine erste Plausibilitätsprüfung des Testergebnisses ermöglicht. Es ist allerdings weniger genau als eine Prüfung auf Konformität mit der Spezifikation und entspricht damit einem suboptimalen Testorakel entsprechend Definition 3-13. Die Evaluierung der Wirkungen zu einem dynamischen Testfall bei gleichzeitiger Prüfung vor dem Hintergrund der Spezifikation ermöglicht es, mit dem Prüfling auch die Entscheidungstabelle zu prüfen, die einem aus der Spezifikation abgeleiteten Modell entspricht.

In der Literatur [Bal98, Lig02] wird auf die Möglichkeit verwiesen, Entscheidungsbäume für die Beschreibung kausaler Zusammenhänge zu verwenden. Es handelt sich dabei um eine spezielle Darstellungsform reduzierter Entscheidungstabellen. Ein Entscheidungsbaum beschreibt die relevanten Ursachenkombinationen im Hinblick auf eine Wirkung. Er wird als binärer Baum beschrieben, dessen innere Knoten die Ursachen darstellen und dessen Blätter die Wirkung beschreiben. Auf einem Weg von der Wurzel zu einem Blatt wird an jedem inneren Knoten entschieden, welche Ausprägung der zugehörigen Ursache angenommen wird. Das erreichte Blatt zeigt dann die geforderte Ausprägung der Wirkung. Ein Entscheidungsbaum legt die Reihenfolge der Evaluierung der Ursachen fest, unvollständige Evaluierungen der Ursachen sind zulässig. Ursachen, die auf einem Weg nicht evaluiert werden, entsprechen den Einträgen „Don't Care" in die reduzierte Ursache-Wirkungs-Tabelle. Entscheidungsbäume können durch das systematische Durchlaufen aller Wege von der Wurzel zu den Blättern zeilenweise in einer reduzierten Entscheidungstabelle abgelegt werden. Der dynamische Test auf Basis von Entscheidungsbäumen ist daher vollständig durch Verfahren 4-7 beschrieben.

Die Ableitung der reduzierten Ursache-Wirkungs-Tabelle und der zugehörigen Ursachenkombinationsforderungen aus einer informal beschriebenen Spezifikation wird in Verfahren 4-7 durch ein Vorgehen beschrieben, das von Intuition und Ermessen des Testers abhängt: So ist bei der Ableitung der Ursachen und Wirkungen und den logischen Verknüpfungen aus der Spezifikation das Spezifikationsverständnis des Testers entscheidender Bedeutung. Diese Modellbildung auf Basis des Prosatextes hat eine Reihe von Vorzügen: Sie schärft das Verständnis für die in der Spezifikation beschriebenen kausalen Abhängigkeiten und ermöglicht die Aufdeckung von Widersprüchen sowie Unvollständigkeiten in der Spezifikation. Eine Automatisierung ist daher nicht möglich und auch nicht sinnvoll. Im Anschluss an die Analyse des Spezifikationstextes werden alle semantisch sinnvollen Ursachenkombinationen aufgezählt und ihre geforderten Wirkungen notiert. Diese Aufstellung kann durch ein Werkzeug unterstützt werden, die Sinnhaftigkeit einzelner Kombinationen muss jedoch vom Tester vor dem semantischen Hintergrund überprüft werden. Die Ursache-Wirkungs-Tabelle wird anschließend reduziert und auf die relevanten kausalen Zusammenhänge beschränkt. Diese Reduktion kann für große Tabellen sinnvoll durch eine automatisierte Suchfunktion unterstützt werden. Sie wird jedoch durch das Ermessen des Testes bestimmt, der über die Relevanz der Ursachenkombinationen für den dynamischen Test zu entscheiden hat.

Mit der endgültigen Beschreibung der reduzierten Ursache-Wirkungs-Tabelle ist das weitere Vorgehen im dynamischen Test weitgehend festgelegt. Die Möglichkeit zur weiteren automatisierten Unterstützung hängt nun von der automatisierten Auswertbarkeit der prädikativen Beschreibung von Ursachen

und Wirkungen ab: Falls es möglich ist, die Testdaten automatisch bezüglich der zugehörigen Ursachen und die Testergebnisse bezüglich der zugehörigen Wirkungen auszuwerten, kann eine Adäquatheitsprüfung für beliebige dynamische Testläufe durch ein Werkzeug geleistet werden. Weiterhin ist es in diesem Fall möglich, die Auswertung des dynamischen Tests durch ein automatisiertes suboptimales Testorakel entsprechend Definition 3-13 zu unterstützen, das die Konformität eines dynamischen Testfalls t und seines Ergebnisses p(t) mit der Entscheidungstabelle prüft und Abweichungen identifiziert. Ist es darüber hinaus möglich ist, dynamische Testfälle zu vorgegebenen Ursachenkombinationen zu erzeugen, kann die Testdatenselektion automatisiert werden.

Eine darüber hinaus gehende Automatisierung oder Standardisierung des dynamischen Tests auf Basis einer Entscheidungstabelle ist nur dann denkbar, wenn ihre Darstellung bereits in der Spezifikation verankert wird. Dies wird beispielsweise in den Empfehlungen in [Bei90] angeregt. Eine solche Beschreibung kann die Spezifikation sinnvoll ergänzen oder auch hauptsächlicher Träger der Informationen sein. Dabei ist jedoch zu beachten, dass in der Spezifikation eine möglichst vollständige Beschreibung der zu spezifizierenden Relation entsprechend Definition 3-2 erreicht werden muss.

4.2.2 Ursache-Wirkungs-Analyse

Das Verfahren der Ursache-Wirkungs-Analyse beschreibt eine weitere Ausprägung des dynamischen Tests auf Basis der spezifizierten kausalen Zusammenhänge zwischen Ursachen und Wirkungen. Hier wird eine konkrete Vorgabe zur Ableitung der für den Test relevanten Ursachenkombinationen gemacht, die auf eine intensive Prüfung der spezifizierten Zusammenhänge mit einer möglichst geringen Anzahl von dynamischen Testfällen abzielt [Mye79]. Abschnitt 4.2.2.1 stellt die klassische Ursache-Wirkungs-Analyse entsprechend ihrer üblichen Darstellung vor. Dabei werden Definitionen festgehalten, die eine weitgehend formalisierte Verfahrensbeschreibung ermöglichen. Abschnitt 4.2.2.2 beschreibt mögliche Vervollständigungen und Erweiterungen des klassischen Verfahrens.

Auf eine Einführung der graphischen Beschreibung der Ursache-Wirkungs-Analyse mit Hilfe von logischen Netzwerken wird hier verzichtet, da diese für eine formale Beschreibung weniger geeignet ist und umfassend in den Standardwerken [Mye79, Rie97, Lig02] behandelt wird.

4.2.2.1 Klassische Ursache-Wirkungs-Analyse

Das Verfahren der klassischen Ursache-Wirkungs-Analyse wurde bereits 1976 von Myers veröffentlicht [Mye76] und später in seinem Lehrbuch zum Softwaretest [Mye79] einer breiten Öffentlichkeit vermittelt. Seine Verwendung wird heute in Normen und Standardwerken zum dynamischen Test empfohlen [IEC 61508 99, EN 50128 01, Rie97, Lig02].

Myers geht bei seinen Ausführungen von einer Spezifikation der geforderten Funktionalität als Fließtext in natürlicher Sprache aus, in der das geforderte Systemverhalten durch Angabe von Ursachen und ihren Wirkungen beschrieben ist. Diese Form der Spezifikation entspricht den Gepflogenheiten der Praxis und ist häufig anzutreffen. Obwohl informal, beinhaltet sie die Chance, einen semantischen Bezug zwischen dem formal festzulegenden Ein- und Ausgabebereich und dem intendierten Umfeld des spezifizierten Systems herzustellen. Dieser Hintergrund liefert wichtige Informationen zur Analyse des intendierten Systemverhaltens, die auch für die Ursache-Wirkungs-Analyse genutzt werden können.

Ebenso wie bei der Analyse der Spezifikation zur Aufstellung einer Entscheidungstabelle wird zur Durchführung der Ursache-Wirkungs-Analyse die Spezifikation in beherrschbare Teilabschnitte zer-

legt. Die Teilabschnitte der Spezifikation werden auf ihre kausalen Zusammenhänge untersucht, die durch Ursachen und ihre Wirkungen beschrieben sind. Die in der Ursache-Wirkungs-Analyse betrachteten Ursachen u_1, u_2, ..., u_m ($m \in \aleph$) und Wirkungen w_1, w_2, ..., w_n ($n \in \aleph$) können wie in Definition 4-9 und Definition 4-10 definiert werden.

Anders als bei der Notation in Form einer Entscheidungstabelle werden bei der Ursache-Wirkungs-Analyse die im Spezifikationstext beschriebenen kausalen Zusammenhänge zwischen Ursachen und Wirkungen nicht durch die möglichen Wahrheitswertbelegungen charakterisiert, sondern mit Hilfe aussagenlogischer Formeln notiert. Für diese Formulierung wird in der Praxis im Allgemeinen eine graphische Darstellung als so genannte logische Netzwerke verwendet [Mye79, Rie97, Lig02]. Hier wird eine äquivalente Darstellung durch logische Ausdrücke genutzt, die sich für eine prädikative Verfahrensbeschreibung besser eignet. Sie werden im Folgenden als Ursache-Wirkungs-Zusammenhänge bezeichnet und wie folgt formuliert.

Definition 4-14: Ursache-Wirkungs-Zusammenhang

Ein Ursache-Wirkungs-Zusammenhang der Ursachen u_1, u_2, ..., u_m ($m \in \aleph$) und Wirkungen w_1, w_2, ..., w_n ($n \in \aleph$) wird beschrieben durch einen in der Spezifikation formulierten kausalen Zusammenhang zwischen einer Wirkung und mehreren Ursachen. Er wird als Paar

$$z := (w,\ \lambda_w^{u_{x_1}, u_{x_2}, \ldots, u_{x_g}})$$

mit $x_i \in \{ 1, \ldots, m \}$ für $i \in \{ 1, \ldots, g \}$ und $g \in \aleph$ notiert. Hierbei bezeichnet w die Wirkung und $\lambda_w^{u_{x_1}, u_{x_2}, \ldots, u_{x_g}}$ einen logischen Ausdruck über dem Alphabet der Ursachenbezeichner, der durch Verwendung der logischen Operatoren $\wedge$, $\vee$ und $\neg$ sowie durch beliebige Klammerungen aufgebaut ist.

Werden für eine Wirkung w im Spezifikationstext mehrere Ursache-Wirkungs-Zusammenhänge beschrieben, so können diese mit dem Operator $\vee$ verknüpft als ein Zusammenhang $\lambda_w^{u_{x_1}, u_{x_2}, \ldots, u_{x_g}}$ dargestellt werden. Die Anzahl der Ursache-Wirkungs-Zusammenhänge z_1, z_2, ... , z_n ($n \in \aleph$) kann daher der Anzahl der Wirkungen gleichgesetzt werden.

Zusätzlich zu den spezifizierten Abhängigkeiten zwischen Ursachen und Wirkungen können im Spezifikationstext Randbedingungen für Ursachen oder Wirkungen angegeben werden, die immer erfüllt sein müssen. Auch können sich Randbedingungen aus der semantischen Interpretation der Ursachen und Wirkungen vor dem Hintergrund des intendierten Systemumfelds ergeben. Die Randbedingungen können wie folgt beschrieben werden.

Definition 4-15: Randbedingung bezüglich der Ursachen und Wirkungen

(a) Eine Randbedingung r bezüglich der Ursachen beschreibt eine Forderung an das gleichzeitige Eintreten oder Nichteintreten von Ursachen, die immer zu gewährleisten ist. Sie wird entweder direkt im Spezifikationstext beschrieben oder ergibt sich vor dem semantischen Hintergrund der intendierten Systemumgebung. Formal kann sie durch einen logischen Ausdruck $\lambda_r^{u_{x_1}, u_{x_2}, ..., u_{x_g}}$ mit $x_i \in \{ 1, \ldots, m \}$ für $i \in \{ 1, \ldots, g \}$ und $g \in \aleph$ über dem Alphabet der Ursachenbezeichner beschrieben werden.

(b) Eine Randbedingung r zwischen Wirkungen beschreibt eine Forderung an das gleichzeitige Eintreten oder Nichteintreten von Wirkungen, die immer zu gewährleisten ist. Sie wird entweder direkt im Spezifikationstext beschrieben oder ergibt sich vor dem semantischen Hintergrund der intendierten Systemumgebung. Formal kann sie durch einen logischen Ausdruck $\lambda_r^{w_{x_1}, w_{x_2}, ..., w_{x_g}}$ mit $x_i \in \{ 1, \ldots, n \}$ für $i \in \{ 1, \ldots, g \}$ und $g \in \aleph$ über dem Alphabet der Wirkungsbezeichner beschrieben werden.

Die aus der Spezifikation resultierende Menge der Randbedingungen für Ursachen und Wirkungen wird als endlich vorausgesetzt und mit $r_1, r_2, ..., r_q$ ($q \in \aleph$) bezeichnet. In der Literatur wird eine Vielzahl von Randbedingungen diskutiert. Beispielsweise fordert die so genannte exklusive Abhängigkeit

$$\lambda_r^{exkl, u_{x_1}, u_{x_2}, ..., u_{x_g}} := \neg(u_{x_1} \wedge u_{x_2}) \wedge \neg(u_{x_1} \wedge u_{x_3}) \wedge ... \wedge \neg(u_{x_{g-1}} \wedge u_{x_g}),$$

dass von mehreren Ursachen $u_{x_1}, u_{x_2}, ..., u_{x_g}$ nur maximal eine eintreten kann. Entsprechend wird die Randbedingung, dass von mehreren Ursachen $u_{x_1}, u_{x_2}, ..., u_{x_g}$ stets mindestens eine eintreten muss, als inklusive Beziehung durch

$$\lambda_r^{inkl, u_{x_1}, u_{x_2}, ..., u_{x_g}} := (u_{x_1} \vee u_{x_2} \vee ... \vee u_{x_g})$$

beschrieben. Die Forderung, dass von den Ursachen $u_{x_1}, u_{x_2}, ..., u_{x_g}$ immer genau eine erfüllt sein muss, wird als „one, and only one"-Bedingung bezeichnet und kann durch

$$\lambda_r^{eins, u_{x_1}, u_{x_2}, ..., u_{x_g}} := \lambda_r^{exkl, u_{x_1}, u_{x_2}, ..., u_{x_g}} \wedge \lambda_r^{inkl, u_{x_1}, u_{x_2}, ..., u_{x_g}}$$

notiert werden. Weiterhin ist es möglich, dass bei Eintreten einer Ursache u_{x_1} immer auch das Eintreten einer anderen Ursache u_{x_2} gefordert ist. Diese Abhängigkeit wird als „requires"-Bedingung bezeichnet und durch

$$\lambda_r^{erfordert, u_{x_1}, u_{x_2}} := (u_{x_1} \Rightarrow u_{x_2})$$

beschrieben. Der Sachverhalt, dass eine Ursache u_1 das Eintreten einer anderen Ursache u_2 verhindert, wird mit der „verhindert"-Bedingung dargestellt. Formal kann die „verhindert"-Bedingung durch

$$\lambda_r^{verhindert, u_{x_1}, u_{x_2}} := (u_{x_1} \Rightarrow \neg u_{x_2})$$

angegeben werden. Liggesmeyer bezeichnet diese Bedingung in [Lig02] als „Maskierung", während Myers in [Mye79] die Maskierung als Verknüpfung

$$\lambda_r^{maskiert, w_{x_1}, w_{x_2}} := (w_{x_1} \Rightarrow \neg w_{x_2})$$

zwischen Wirkungen einführt. Zahlreiche weitere Verknüpfungen zwischen Ursachen sind denkbar. So schlägt Riedemann die Einführung einer weiteren Bedingung bezüglich der Ursachen vor, die die Irrelevanz einer Ursache u_2 bei Eintreten einer anderen Ursache u_1 beschreibt [Rie97]. Ziel ist die eindeu-

tige Handhabung von Prädikaten, die unter bestimmten Bedingungen nicht evaluiert werden können. So kann beispielsweise der Typ des nächsten Datensatzes nicht ausgewertet werden, wenn das Dateiende erreicht wurde. Betrachtet man Prädikate nur dann als erfüllt, wenn die vollständig evaluiert werden können, ist eine zusätzliche Verknüpfung nicht notwendig. Sie kann dann, wie in [Lig02] beschrieben, bei negativer Belegung des nicht evaluierbaren Prädikats durch eine „verhindert"-Bedingung ersetzt werden. Eine negative Belegung nicht evaluierbarer Prädikate entspricht der Definition 4-9. Zusätzlich zu diesen in der Literatur beschriebenen Bedingungen kann die Einführung weiterer Randbedingungen sinnvoll sein: Da Randbedingungen im dynamischen Test stets zu erfüllen sind, helfen sie bei der Auswahl semantisch sinnvoller Testfälle und beschränken so die Zahl der geforderten Testfälle. Es ist daher zweckmäßig, möglichst viele Randbedingungen festzuhalten und bei der Testfalldefinition zu berücksichtigen.

Im Anschluss an die Analyse des Spezifikationstextes liegt eine formale Beschreibung der Zusammenhänge zwischen Ursachen und Wirkungen der ursprünglich informal beschriebenen Spezifikation vor. Dieses Modell beschreibt die Ursachen und Wirkungen, die spezifizierten Ursache-Wirkungs-Zusammenhänge sowie die Randbedingungen bezüglich des gleichzeitigen Auftretens von Ursachen oder Wirkungen.

Definition 4-16: Ursache-Wirkungs-Modell

Als Ursache-Wirkungs-Modell zu einer Spezifikation $s \in \Sigma_{I, O}$ wird im Folgenden ein Quadrupel

$Mod_s := (U_s, W_s, Z_s, R_s)$

einer endlichen Menge von Ursachen U_s aus s, einer endlichen Menge von Wirkungen W_s aus s, der zugehörigen, in s beschriebenen Ursache-Wirkungs-Zusammenhänge Z_s und der in s beschriebenen Menge von Randbedingungen R_s zwischen den Ursachen U_s und zwischen den Wirkungen W_s bezeichnet.

Auf den im Ursache-Wirkungs-Modell zusammengefassten Informationen operiert das von Myers vorgeschlagene dynamische Testverfahren. Er leitet daraus eine abstrakte Beschreibung der dynamischen Testfälle ab, die sowohl zur Testdatenselektion als auch zur Prüfung der Testdatenadäquatheit verwendet wird. Der wichtigste Schritt zur Ableitung dieser Beschreibung ist die Auswahl der Ursachenkombinationen, die im dynamischen Test zu prüfen sind. Myers beschreibt die geforderten Ursachenkombinationen und die durch sie bedingten Wirkungen in Form einer Ursache-Wirkungs-Tabelle entsprechend Definition 4-11. Hier wird zur besseren Referenzierbarkeit in der Verfahrensbeschreibung die äquivalente Form der im Folgenden definierten Ursachenkombinationsforderung verwendet.

Definition 4-17: Ursachenkombinationsforderung, resultierende Wirkung und Konformität mit Randbedingungen

(a) Zu einem gegebenen Ursache-Wirkungs-Modell

$Mod_s := (U_s, W_s, Z_s, R_s)$

wird ein logischer Ausdruck $\lambda_f^{u_1, u_2, \ldots, u_m}$ über dem Alphabet der Ursachenbezeichner als Ursachenkombinationsforderung f bezeichnet, wenn er ohne Verwendung von Klammerungen mit den Operatoren $\wedge$ und $\neg$ aufgebaut ist, alle Ursachenbezeichner genau einmal enthalten sind und seine Erfüllung unter Einhaltung der Randbedingungen bezüglich Ursachen und Wirkungen möglich ist.

(b) Eine Wirkung w resultiert genau dann aus einer gegebenen Ursachenkombinationsforderung f, wenn die zugehörigen logischen Ausdrücke die Beziehung

$$\lambda_f^{u_1, u_2, ..., u_m} \Rightarrow \lambda_w^{u_{x1}, u_{x2}, ..., u_{xg}}$$

erfüllen.

(c) Die Ursachenkombinationsforderung f wird genau dann als konform mit einer Randbedingung r bezeichnet, wenn

$$\lambda_f^{u_1, u_2, ..., u_m} \Rightarrow \lambda_r^{u_{x1}, u_{x2}, ..., u_{xg}}$$

beziehungsweise

$$\lambda_f^{u_1, u_2, ..., u_m} \Rightarrow \lambda_r^{w_{x1}, w_{x2}, ..., w_{xg}}$$

gilt.

Zur Ableitung der Ursachenkombinationsforderungen aus einem Ursache-Wirkungs-Modell schlägt Myers die schrittweise Analyse einzelner Ursache-Wirkungs-Zusammenhänge ausgehend von der Wirkung vor. Für die Ursache-Wirkungs-Zusammenhänge und ihre Teilbedingung werden Forderungen an die Wahrheitswertbelegung formuliert, die sukzessive zu Ursachenkombinationsforderungen verfeinert werden. Die folgende Formulierung ist aus der Vorgehensbeschreibung in [Mye79] und den Beispielen abgeleitet und wurde lediglich an die hier verwendete Terminologie und Darstellungsweise angepasst.

Heuristik 4-1: Herleitung der Ursachenkombinationsforderungen nach [Mye79]

Für jede Wirkung werden alle diejenigen Ursachenkombinationsforderungen aufgestellt, die die Wirkung verursachen, die Randbedingungen nicht verletzen und die den folgenden Einschränkungen genügen:

(a) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\vee$ beschrieben, und wird ihre Erfüllung gefordert, so wird für jede Teilbedingung eine Ursachenkombination gefordert, die die Teilbedingung erfüllt, alle anderen aber nicht.

(b) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\wedge$ beschrieben, und wird ihre Nichterfüllung gefordert, so wird für jede mögliche Belegung ihrer Teilbedingungen, die zur Nichterfüllung führt, eine Ursachenkombination gefordert. Für Teilbedingungen, die in einer solchen Konstellation erfüllt sind, reicht die Angabe einer einzigen Ursachenkombination in der Ursache-Wirkungs-Tabelle.

(c) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\wedge$ beschrieben, und wird ihre Nichterfüllung gefordert, so wird nur eine Kombination von Ursachen gefordert, die alle Teilbedingungen nicht erfüllt.

Für die so ermittelten Ursachenkombinationsforderungen ist das Eintreten beziehungsweise Nichteintreten aller Wirkungen aus den spezifizierten Ursache-Wirkungs-Zusammenhängen zu ermitteln. Ursachenkombinationen, die nicht mit den Randbedingungen konform sind, sind aus der Ursache-Wirkungs-Tabelle zu streichen.

Diese Heuristik zielt auf eine systematische Prüfung der Realisierung der spezifizierten Ursache-Wirkungs-Zusammenhänge mit einer möglichst geringen Anzahl von Testfällen. Weniger aussagekräf-

tige Ursachenkombinationen sollen vermieden werden, ebenso Fehlermaskierungen, die eine falsche Realisierung im Prüfling im dynamischen Test verschleiern können. Als Herleitungsalgorithmus für die Ursachenkombinationsforderungen ist diese Darstellung nicht geeignet, da sie hauptsächlich Restriktionen angibt, aber nicht konstruktiv formuliert ist. In [Lig02] wird die Herleitung der Ursachenkombinationsforderungen in einer algorithmischen Form angegeben. Sie wird im Folgenden mit der hier verwendeten Terminologie wiedergegeben.

Algorithmus 4-1: Herleitung der Ursachenkombinationsforderungen nach [Lig02]

1. Auswahl einer Wirkung.
2. Ermittlung von Ursachenkombinationen, die die Wirkung eintreten beziehungsweise nicht eintreten lassen. Dabei wird schrittweise der zugehörige Ursache-Wirkungs-Zusammenhang von der Wirkung hin zu den Ursachen analysiert:

(a) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\vee$ beschrieben, und wird ihre Erfüllung gefordert, so sind auf der unterlagerten Ebene die Wahrheitswertbelegungen zu fordern, die jeweils eine der Teilbedingungen erfüllen, alle anderen jedoch nicht erfüllen.

(b) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\vee$ beschrieben, und wird ihre Nichterfüllung gefordert, so ist auf der unterlagerten Ebene die Nichterfüllung aller Teilbedingungen zu fordern.

(c) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\wedge$ beschrieben, und wird ihre Nichterfüllung gefordert, so sind auf der unterlagerten Ebene alle Wahrheitswertbelegungen zu fordern, die jeweils eine der Teilbedingungen nicht erfüllen, alle anderen jedoch erfüllen.

(d) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\wedge$ beschrieben, und wird ihre Erfüllung gefordert, so ist auf der unterlagerten Ebene die Erfüllung aller Teilbedingungen zu fordern.

3. Ermittlung der Wirkungen zu den Ursachenkombinationen, Prüfung auf Konformität mit den Randbedingungen, gegebenenfalls Aufstellung der Ursachenkombinationsforderung.

Diese konstruktive, algorithmische Beschreibung der Herleitung der Ursache-Wirkungs-Tabelle systematisiert und vervollständigt die Ideen aus [Mye79]. Für die Wirkungen des Ursache-Wirkungs-Modells werden endlich viele Ursachenkombinationsforderungen aufgestellt; ihre Anzahl wird durch das Vorgehen im Algorithmus deutlich begrenzt: Für jede komplexe Teilbedingung eines Ursache-Wirkungs-Zusammenhangs, die eine Verknüpfung aus k weiteren Teilbedingungen darstellt, werden nur höchstens k + 1 der maximal möglichen 2^k Forderungen an die Teilbedingungen abgeleitet. Die abgeleiteten Ursachenkombinationsforderungen ergänzen das Ursache-Wirkungs-Modell und gestatten die folgende Beschreibung des dynamischen Tests auf Basis einer Ursache-Wirkungs-Analyse.

Verfahren 4-8: Dynamisches Testen auf Basis einer Ursache-Wirkungs-Analyse

(a) Voraussetzung:
Aus der Spezifikation $s \in \Sigma_{I,O}$ kann das Ursache-Wirkungs-Modell $Mod_s = (U_s, W_s, Z_s, R_s)$ mit den Ursachen

$$U_s = \{ u_1, u_2, \ldots, u_m \} \qquad (m \in \mathbb{N}),$$

den Wirkungen

$W_s = \{ w_1, w_2, \ldots, w_n \}$ $(n \in \aleph)$,

den Ursache-Wirkungs-Zusammenhängen

$Z_s = \{ z_1, z_2, \ldots, z_n \}$

und den Randbedingungen

$R_s = \{ r_1, r_2, \ldots, r_q \}$ $(q \in \aleph)$

abgeleitet werden. Aus diesem seien nach Algorithmus 4-1 die Ursachenkombinationsforderungen

$F_{Mod_s} = \{ f_1, f_2, \ldots, f_k \}$ $(k \in \aleph)$

abgeleitet.

(b) Verfahrensbeschreibung:

Der dynamische Test auf Basis einer Ursache-Wirkungs-Analyse wird beschrieben durch

$$X^{UWA}: \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(v \in \{ 1, \ldots, k \}) \exists(t \in T)\ f_v(t).$$

Der dynamische Test auf Basis einer Ursache-Wirkungs-Analyse akzeptiert alle Testläufe, die zu jeder Ursachenkombinationsforderung mindestens einen Testfall enthalten, der diese erfüllt. Weitere Kriterien werden in der Verfahrensbeschreibung nicht aufgenommen werden, da sie bereits bei der Erstellung der Kombinationsforderungen durch Algorithmus 4-1 berücksichtigt werden. Die Ursachenkombinationsforderungen geben somit, ebenso wie die reduzierte Ursache-Wirkungs-Tabelle in Verfahren 4-7, eine abstrakte Beschreibung der Testdaten: Sie beschreiben die Ursachenkombinationen, die im dynamischen Test zu prüfen sind, und erlauben die Ermittlung der Wirkungen, die dem Ursache-Wirkungs-Modell zufolge aus diesen Ursachenkombinationen resultieren. Sie stellen somit eine Anleitung zur Testdatenselektion und ein Testdatenadäquatheitskriterium bereit. Durch die Ermittlung der geforderten Wirkungen ist ein Prüfkriterium für die Testdurchführung gegeben. Es ermöglicht eine erste Plausibilitätsprüfung des Testergebnisses, ist allerdings weniger genau als eine vollständige Prüfung auf Basis der Spezifikation. Die Prüfung entspricht damit einem suboptimalen Testorakel entsprechend Definition 3-13.

Zur Ableitung der Ursachenkombinationsforderungen aus einer informal beschriebenen Spezifikation nach Verfahren 4-8 sind Ursachen, Wirkungen, Ursache-Wirkungs-Zusammenhänge und Randbedingungen aus der Spezifikation herauszulesen. Hierbei sind Kreativität des Testers von entscheidender Bedeutung. So sind Randbedingungen und logische Abhängigkeiten nicht immer explizit im Text beschrieben, sondern ergeben sich möglicherweise erst vor dem Hintergrund der semantischen Belegung im realen Kontext des Systems. Dieses intuitive Vorgehen der Modellbildung auf Basis des Prosatextes hat eine Reihe von Vorzügen: So schärft es das Verständnis für die in der Spezifikation beschriebenen kausalen Abhängigkeiten und ermöglicht die Aufdeckung von Widersprüchen und Unvollständigkeiten in der Spezifikation. Es stellt somit einen wichtigen Schritt in der Qualitätssicherung zum Formalisierungsprozess der Softwareentwicklung dar. Eine Automatisierung dieser Analyse ist daher weder möglich noch wünschenswert, Werkzeugunterstützung ist lediglich bei der syntaktischen Analy-

se des Prosatextes und bei der Beschreibung und Verwaltung der logischen Zusammenhänge denkbar. Im Anschluss an die Analyse des Spezifikationstextes werde die Ursachenkombinationsforderungen mit Algorithmus 4-1 aus dem Modell abgeleitet. Diese Ableitung kann bei umfangreichen, eventuell mehrstufig aufgebauten Ursache-Wirkungs-Zusammenhängen mühevoll und zeitintensiv sein, da neben den Vollständigkeits- und Restriktionsvorgaben auch die Randbedingungen betrachtet werden müssen. Diese Analyse kann jedoch automatisiert werden: Myers verweist auf bestehende Werkzeuge, die jedoch nicht öffentlich zugänglich sind [Mye79]. Weitere Möglichkeiten zur Werkzeugunterstützung der Durchführung des dynamischen Tests auf Basis des erweiterten Ursache-Wirkungs-Modells entsprechend Verfahren 4-8 hängen von der automatisierten Auswertbarkeit der prädikativen Beschreibung von Ursachen und Wirkungen ab: Falls es möglich ist, die Testdaten automatisch bezüglich der zugehörigen Ursachen und die Testergebnisse bezüglich der zugehörigen Wirkungen auszuwerten, kann eine Adäquatheitsprüfung für dynamische Testläufe durch ein Werkzeug geleistet werden. Weiterhin ist es in diesem Fall möglich, die Auswertung des dynamischen Tests durch ein automatisiertes suboptimales Testorakel entsprechend Definition 3-13 zu unterstützen, dass die Konformität eines dynamischen Testfalls t und seines Ergebnisses p(t) mit dem Ursache-Wirkungs-Modell prüft und Abweichungen identifiziert. Ist es darüber hinaus möglich ist, dynamische Testfälle zu vorgegebenen Ursachenkombinationen zu erzeugen, kann auch die Testdatenselektion automatisiert werden. Um eine weitgehende Unterstützung der Ursache-Wirkungs-Analyse zu erreichen, ist es daher sinnvoll, auf die automatisierte Auswertbarkeit der prädikativen Beschreibungen der Ursachen und Wirkungen zu achten.

Eine darüber hinaus gehende Automatisierung des dynamischen Tests auf Basis einer Ursache-Wirkungs-Analyse ist nur dann denkbar, wenn eine Darstellung der Ursache-Wirkungs-Modells entsprechend Definition 4-16 bereits in der Spezifikation verankert wird. Eine solche Beschreibung kann die Spezifikation sinnvoll ergänzen oder auch hauptsächlicher Träger der Information sein. Dabei ist jedoch zu beachten, dass in der Spezifikation eine möglichst vollständige, über die Angaben des Ursache-Wirkungs-Modells hinausgehende Beschreibung der in Definition 3-2 beschriebene Relation erreicht werden muss.

4.2.2.2 Mögliche Erweiterungen der Ursache-Wirkungs-Analyse

Das Verfahren der klassischen Ursache-Wirkungs-Analyse liefert einen guten Ansatz zur Ableitung und Bewertung von dynamischen Testläufen. Es kann jedoch aufgrund des heuristischen Charakters der verwendeten Algorithmen eine Vollständigkeit oder Minimalität der abgeleiteten dynamischen Testläufe nicht garantieren. Die folgenden Abschnitte verdeutlichen diese Probleme und untersuchen Möglichkeiten zur Erweiterung des klassischen Verfahrens der Ursache-Wirkungs-Analyse.

4.2.2.2.1 Auswahl von Ursachenkombinationen im Hinblick auf Effizienz

Das folgende Beispiel zeigt, dass trotz der in Algorithmus 4-1 beschriebenen Einschränkungen der geforderten Wahrheitswertbelegungen umfassendes Optimierungspotential bei der Bestimmung der zu prüfenden Ursachenkombinationen in Verfahren 4-8 nicht genutzt wird. Selbst bei strikter Anwendung der Heuristiken werden Ursachenkombinationen gefordert, deren Prüfung bei näherem Hinsehen nicht sinnvoll erscheint. So werden im folgenden Beispiel Anforderungen an die Wahrheitswertbelegungen einer komplexen Teilbedingung gestellt, die die Wirkung gar nicht beeinflussen können. Das Beispiel geht von einem Ursache-Wirkungs-Zusammenhang

$$z := (w,\ \lambda_w^{u_1,u_2,u_3,u_4,u_5})$$

mit

$$\lambda_w^{u_1,u_2,u_3,u_4,u_5} := (u_1 \wedge u_2 \wedge u_3) \vee u_4 \vee u_5$$

aus. So ergeben sich aus der Forderung der Erfüllung und Nichterfüllung der Wirkung im dynamischen Test

(a) $\exists(t \in T)\ (\lambda_w^{u_1,u_2,u_3,u_4,u_5}(t))$

(b) $\exists(t \in T)\ (\neg\lambda_w^{u_1,u_2,u_3,u_4,u_5}(t))$

nach Algorithmus 4-1 die Forderungen

(a1) $\exists(t \in T)\ ((u_1(t) \wedge u_2(t) \wedge u_3(t)) \wedge (\neg u_4(t)) \wedge \neg(u_5(t)))$,

(a2) $\exists(t \in T)\ (\neg(u_1(t) \wedge u_2(t) \wedge u_3(t)) \wedge u_4(t) \wedge \neg(u_5(t)))$,

(a3) $\exists(t \in T)\ (\neg(u_1(t) \wedge u_2(t) \wedge u_3(t)) \wedge (\neg u_4(t)) \wedge u_5(t))$,

(b1) $\exists(t \in T)\ (\neg(u_1(t) \wedge u_2(t) \wedge u_3(t)) \wedge \neg u_4(t) \wedge \neg u_5(t))$.

Forderung (a3) führt ihrerseits zu den Ursachenkombinationsforderungen

(a31) $\exists(t \in T)\ (\neg u_1(t) \wedge u_2(t) \wedge u_3(t) \wedge \neg u_4(t) \wedge u_5(t))$,

(a32) $\exists(t \in T)\ (u_1(t) \wedge \neg u_2(t) \wedge u_3(t) \wedge \neg u_4(t) \wedge u_5(t))$,

(a33) $\exists(t \in T)\ (u_1(t) \wedge u_2(t) \wedge \neg u_3(t) \wedge \neg u_4(t) \wedge u_5(t))$.

Diese umfassende Prüfung der Teilbedingung $(u_1 \wedge u_2 \wedge u_3)$ unter Vorgabe der Forderung (a3) macht aber nur wenig Sinn, weil eine falsche Realisierung im Prüfling in den dynamischen Testfällen durch $u_5(t)$ maskiert würde. Die Heuristik sollte daher nur dann für eine komplexe Teilbedingung mehr als eine Wahrheitswertbelegung fordern, wenn der Wahrheitswert der Teilbedingung unter den vorgegebenen übergeordneten Bedingungen die Wirkung beeinflussen kann. Ist dies nicht der Fall, reicht die Forderung einer einzigen Wahrheitswertbelegung als Grundlage für die Auswahl einer Ursachenkombination.

Weiterhin ist es sinnvoll, die Realisierung einer komplexen Teilbedingung im Prüfling nur ein einziges Mal zu überprüfen. So genügt es, sich zur weiteren Verfeinerung der Anforderungen (a1) und (b1) auf die Anforderungen

(a11) $\exists(t \in T)\ (u_1(t) \wedge u_2(t) \wedge u_3(t) \wedge \neg u_4(t) \wedge \neg u_5(t))$

(b11) $\exists(t \in T)\ (\neg u_1(t) \wedge u_2(t) \wedge u_3(t) \wedge \neg u_4(t) \wedge \neg u_5(t))$

(b12) $\exists(t \in T)\ (u_1(t) \wedge \neg u_2(t) \wedge u_3(t) \wedge \neg u_4(t) \wedge \neg u_5(t))$

(b13) $\exists(t \in T)\ (u_1(t) \wedge u_2(t) \wedge \neg u_3(t) \wedge \neg u_4(t) \wedge \neg u_5(t))$

zu beschränken. Falsche Realisierungen im Prüfling haben unter diesen Wahrheitswertbelegungen direkten Einfluss auf die Wirkung. Zur Prüfung der übrigen geforderten Wahrheitswertbelegungen

(a2) $\exists(t \in T)\ (\neg(u_1(t) \wedge u_2(t) \wedge u_3(t)) \wedge u_4(t) \wedge \neg u_5(t))$

(a3) $\exists(t \in T)\ (\neg(u_1(t) \wedge u_2(t) \wedge u_3(t)) \wedge \neg u_4(t) \wedge u_5(t))$

kann dann eine beliebige Ursachenkombination für die erste Teilbedingung gewählt werden, weitere Forderungen an Ursachenkombinationen zur Prüfung der ersten Teilbedingung sind nicht sinnvoll.

Aus diesen beiden Überlegungen resultiert die Idee, die in Algorithmus 4-1 formulierten Forderungen nur auf Teilbedingungen anzuwenden, die die Wirkung direkt beeinflussen können und deren Prüfung noch aussteht. Eine Erweiterung des Algorithmus kann damit wie folgt formuliert werden.

Algorithmus 4-2:	Herleitung der Kombinationsforderungen unter Berücksichtigung des Belegungseinflusses

1. Auswahl einer Wirkung.
2. Ermittlung von Ursachenkombinationen, die die Wirkung eintreten beziehungsweise nicht eintreten lassen. Dabei wird der zum Ursache-Wirkungs-Zusammenhang gehörende Syntaxbaum wie folgt analysiert:

(a) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\vee$ beschrieben und wird ihre Nichterfüllung gefordert, so ist auf der unterlagerten Ebene die Nichterfüllung aller Teilbedingungen zu fordern.

(b) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\vee$ beschrieben, wird ihre Erfüllung gefordert, hat eine Änderung ihrer Wahrheitswertbelegung unter Beibehaltung der übrigen Wahrheitswertbelegungen einen Einfluss auf die Wirkung und wurde das Zustandekommen der Erfüllung der komplexen Bedingung noch nicht systematisch geprüft, so sind auf der unterlagerten Ebene die Wahrheitswertbelegungen zu fordern, die jeweils eine der Teilbedingungen erfüllen, alle anderen jedoch nicht.

(c) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\vee$ beschrieben, wird ihre Erfüllung gefordert, und hat eine Änderung ihrer Wahrheitswertbelegung unter Beibehaltung der übrigen Wahrheitswertbelegungen keinen Einfluss auf die Wirkung oder wurde das Zustandekommen der Erfüllung der komplexen Bedingung bereits systematisch geprüft, so ist auf der unterlagerten Ebene eine beliebige Belegung zu fordern, die zur Erfüllung der Teilbedingung führt.

(d) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\wedge$ beschrieben, und wird ihre Erfüllung gefordert, so ist auf der unterlagerten Ebene die Erfüllung aller Teilbedingungen zu fordern.

(e) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\wedge$ beschrieben, wird ihre Nichterfüllung gefordert, hat eine Änderung ihrer Wahrheitswertbelegung unter Beibehaltung der übrigen Wahrheitswertbelegungen einen Einfluss auf die Wirkung und wurde das Zustandekommen der Nichterfüllung der komplexen Bedingung noch nicht systematisch geprüft, so sind auf der unterlagerten Ebene alle Belegungen zu fordern, die jeweils eine der Teilbedingungen nicht erfüllen, alle anderen jedoch erfüllen.

(f) Ist eine komplexe Bedingung als Verknüpfung mehrerer Teilbedingungen mit dem Operator $\wedge$ beschrieben, wird ihre Nichterfüllung gefordert, und hat eine Änderung ihrer Wahrheitswertbelegung unter Beibehaltung der übrigen Wahrheitswertbelegungen keinen Einfluss auf die Wirkung oder wurde das Zustandekommen der Nichterfüllung der komplexen Bedingung bereits systematisch geprüft, so sind auf der unterlagerten Ebene eine beliebige Belegung zu fordern, die zur Nichterfüllung der Teilbedingung führt.

3. Ermittlung der Wirkungen zu den Ursachenkombinationen, Prüfung auf Konformität mit den Randbedingungen, gegebenenfalls Aufstellung der Ursachenkombinationsforderung.

Auch wenn die textliche Darstellung des Algorithmus umfangreich und schwer lesbar ist, lässt sich die Ableitung der Kombinationsforderungen aus dem Ursache-Wirkungs-Zusammenhang einfach auf dem Syntaxbaum des zum Ursache-Wirkungs-Zusammenhang gehörenden logischen Ausdrucks umsetzen. Die Strategie einer Tiefensuche kann dabei die systematische Prüfung unterlagerter Teilbedingungen auf einem Pfad mit Auswirkung auf die Wirkung erleichtern. Weiterhin sollte eine Markierung

der bereits überprüften komplexen Teilbedingungen vorgenommen werden, sofern eine fehlerhafte Gestaltung der Teilbedingung einen Einfluss auf die Wirkung hätte haben können. Nach Abschluss der Analyse einer Teilbedingung ist eine erneute Anwendung der Heuristiken bei weiteren Belegungsforderungen nicht notwendig: Hier genügt es, jeweils eine feste Ursachenkombination zur Erfüllung beziehungsweise Nichterfüllung der Teilbedingung zu verwenden.

Durch diese zusätzlichen Regeln bei der Ableitung der relevanten Ursachenkombinationen wird die Anzahl der Ursachenkombinationen deutlich reduziert, ohne dass die Aussagkraft des Verfahrens eingeschränkt wird. So werden im obigen Beispiel zur Prüfung des Ursache-Wirkungs-Zusammenhangs

$$z := (w,\ \lambda_w^{u_1,u_2,u_3,u_4,u_5})$$

mit

$$\lambda_w^{u_1,u_2,u_3,u_4,u_5} := (u_1 \wedge u_2 \wedge u_3) \vee u_4 \vee u_5$$

nach dem ursprünglichen Verfahren zehn Ursache-Wirkungs-Kombinationen gefordert, die in der Ursache-Wirkungs-Tabelle 4-1 zusammengestellt sind.

Bez.	Ursachen					Wirkung
	u_1	u_2	u_3	u_4	u_5	$\lambda_w^{u_1,u_2,u_3,u_4,u_5}$
(a11)	wahr	wahr	wahr	falsch	falsch	wahr
(a21)	falsch	wahr	wahr	wahr	falsch	wahr
(a22)	wahr	falsch	wahr	wahr	falsch	wahr
(a23)	wahr	wahr	falsch	wahr	falsch	wahr
(a31)	falsch	wahr	wahr	falsch	wahr	wahr
(a32)	wahr	falsch	wahr	falsch	wahr	wahr
(a33)	wahr	wahr	falsch	falsch	wahr	wahr
(b11)	falsch	wahr	wahr	falsch	falsch	falsch
(b12)	wahr	falsch	wahr	falsch	falsch	falsch
(b13)	wahr	wahr	falsch	falsch	falsch	falsch

Tabelle 4-1: Ursache-Wirkungs-Kombinationen nach klassischer Ursache-Wirkungs-Analyse am Beispiel

Bei einer Berücksichtigung der Modifikationen reicht hingegen die Prüfung der sechs in der Ursache-Wirkungs-Tabelle 4-2 beschriebenen Ursache-Wirkungs-Kombinationen.

Bez.	Ursachen					Wirkung
	u_1	U_2	u_3	u_4	u_5	$\lambda_w^{u_1,u_2,u_3,u_4,u_5}$
(a11)	wahr	wahr	wahr	falsch	falsch	wahr
(a21)	falsch	wahr	wahr	wahr	falsch	wahr
(a31)	falsch	wahr	wahr	falsch	wahr	wahr
(b11)	falsch	wahr	wahr	falsch	falsch	falsch
(b12)	wahr	falsch	wahr	falsch	falsch	falsch
(b13)	wahr	wahr	falsch	falsch	falsch	falsch

Tabelle 4-2: Ursache-Wirkungs-Kombinationen nach Ursache-Wirkungs-Analyse mit modifiziertem Algorithmus 4-2 am Beispiel

4.2.2.2.2 Auswahl von Ursachenkombinationen im Hinblick auf Vollständigkeit

Im folgenden Abschnitt wird gezeigt, dass auch bei Vorgehensweise nach dem in Abschnitt 4.2.2.1 beschriebenen Verfahren die Vollständigkeit der Prüfung nicht sichergestellt ist. So kann beispielsweise eine zur Prüfung notwendige Ursachenkombination explizit ausgeschlossen sein. Dies zeigt das einfache Beispiel eines Ursache-Wirkungs-Zusammenhangs

$$z := (w,\ \lambda_w^{u_1,u_2,u_3})$$

mit

$$\lambda_w^{u_1,u_2,u_3} := u_1 \vee u_2 \vee u_3.$$

Zusätzlich sei die Randbedingung

$$\lambda_r^{u_1,u_2} := (u_1 \Rightarrow u_2)$$

vorgegeben. Aus den initialen Forderungen der Erfüllung beziehungsweise Nichterfüllung der Wirkung

(a) $\exists(t \in T)\ (\lambda_w^{u_1,u_2,u_3}(t))$

(b) $\exists(t \in T)\ (\neg\lambda_w^{u_1,u_2,u_3}(t))$

werden nach den oben aufgeführten Regeln die Forderungen

(a1) $\exists(t \in T)\ (u_1(t) \wedge \neg u_2(t) \wedge \neg u_3(t))$

(a2) $\exists(t \in T)\ (\neg u_1(t) \wedge u_2(t) \wedge \neg u_3(t))$

(a3) $\exists(t \in T)\ (\neg u_1(t) \wedge \neg u_2(t) \wedge u_3(t))$

sowie

(b1) $\exists(t \in T)\ (\neg u_1(t) \wedge \neg u_2(t) \wedge \neg u_3(t))$

abgeleitet. Bei der Prüfung der Konformität mit der Randbedingung wird die Forderung (a1) eliminiert. Die verbleibenden Forderungen sind jedoch nicht geeignet, eine vollständige Prüfung der Realisierung der kausalen Zusammenhänge sicherzustellen: Keine der Forderungen stellt die Belegung der Teilbedingung u_1 mit dem Wert „wahr“ im dynamischen Test sicher. Eine falsche Umsetzung der Abhängigkeit im Prüfling, beispielsweise in der Form

$$\tilde{\lambda}_w^{u_1,u_2,u_3} := \neg u_1 \wedge (u_2 \vee u_3),$$

würde nicht entdeckt. Tabelle 4-3 macht dies deutlich und zeigt, wie die falsche Realisierung im Prüfling durch Hinzunahme der nicht den Heuristiken entsprechenden Ursachenkombination (x1) aufgedeckt werden kann.

Bez.	Ursachen			Wirkungen	
	u_1	u_2	u_3	$\lambda_w^{u_1,u_2,u_3}$	$\tilde{\lambda}_w^{u_1,u_2,u_3}$
(a2)	falsch	wahr	falsch	wahr	wahr
(a3)	falsch	falsch	wahr	wahr	wahr
(b1)	falsch	falsch	falsch	falsch	falsch
(x1)	wahr	wahr	falsch	wahr	falsch

Tabelle 4-3: Ergänzung der klassischen Ursache-Wirkungs-Kombinationen am Beispiel

Es ist daher notwendig, die Vollständigkeit der Ursache-Wirkungs-Kombinationen im Anschluss an ihre Ableitung und an die Prüfung der Konformität mit den Randbedingungen zu prüfen. Das Prüfkriterium sollte eine umfassende Prüfung von allen spezifizierten Teilbedingungen der komplexen Ursache-Wirkungs-Zusammenhänge sicherstellen. Als Prüfkriterien bieten sich zunächst die in Heuristik 4-1, Algorithmus 4-1 oder Algorithmus 4-2 formulierten Regeln an, die eine Identifikation der fehlenden Ursachenkombinationsforderungen ermöglichen, jedoch keine Hilfestellung bei der Ergänzung geben.

Zur Ableitung weiterer Prüf- und Ergänzungskriterien können Verfahren der strukturorientierten Bedingungsüberdeckung betrachtet werden, die beispielsweise in [Mye79], [ChiMil94] und [Lig02] beschrieben werden: Dort wird eine umfassende und effiziente Prüfung der im Quelltext des Prüflings verankerten Bedingungen angestrebt, während hier eine umfassende und effiziente Prüfung der spezifizierten Ursachenkombinationen gefordert ist. So ist es möglich, auch hier im Sinne einer einfachen Bedingungsüberdeckung die Erfüllung und Nichterfüllung aller atomaren Teilbedingungen, das heißt aller Ursachen, zu fordern. Diese Forderung stellt zwar ein sehr schwaches Kriterium dar, hätte aber bereits die zusätzliche Ursachenkombination (x1) in Tabelle 4-3 gefordert. Auch eine minimal mehrfache Bedingungsüberdeckung kann als Prüfkriterium in Betracht gezogen werden. Sie verlangt das Eintreten und Nichteintreten aller Bedingungen – der Ursachen, ihrer Wirkungen sowie aller Teilbedingungen der Verknüpfung. Auch sie stellt ein schwächeres Kriterium als die ursprünglichen Heuristiken dar, kann aber auf eine unvollständige Prüfung von Teilbedingungen hinweisen, die durch Restriktionen durch die Randbedingungen verursacht wird. Die Forderung einer vollständigen Prüfung aller möglichen Kombinationen von Ursachen im Sinne einer mehrfachen Bedingungsüberdeckung impliziert für m Ursachen die Aufstellung von bis zu 2^m Ursachenkombinationsforderungen, was dem Grundgedanken der sinnvollen Beschränkung der Anzahl der Ursachenkombinationsforderungen durch die Algorithmen entgegensteht. Das modifizierte Bedingungs- und Entscheidungskriterium [ChiMil94] lässt sich hingegen sehr gut zur Vollständigkeitsprüfung der Ursachenkombinationsforderungen anwenden: Übertragen auf die Ursache-Wirkungs-Analyse fordert es für jede Ursache, dass sie mindestens einmal direkt die Wirkung beeinflusst, während die Kombination der übrigen Ursachen nicht verändert wird. Diese Forderung ist in erstaunlichem Maß mit den Zielen der Ursache-Wirkungs-Analyse vereinbar: Wählt man die strenge Form der modifizierten Bedingungs- und Entscheidungsüberdeckung, bei der für jede Instanz einer Ursache im logischen Ausdruck eines Ursache-Wirkungs-Zusammenhangs ihr Einfluss auf die Wirkung bei gleichzeitiger Maskierung der weiteren Instanzen zu

prüfen ist, so lässt sich aus der Erfüllung dieses Kriteriums eine vollständige Überdeckung im Sinne des Algorithmus 4-2 ableiten. Dies ist darin begründet, dass die Variation der Wahrheitswertbelegung einer Ursacheninstanz bei Festhalten aller übrigen Wahrheitswerte auf jeder syntaktischen Ebene genau eine Teilbedingung verändert, die einen direkten Einfluss auf die Wirkung hat. Darüber hinaus beschränkt die Anwendung des modifizierte Bedingungs- und Entscheidungskriterium die Anzahl der zu prüfenden Ursachenkombinationen auf maximal $2 \cdot n \cdot m$ für n Wirkungen beschrieben durch m Ursacheninstanzen [ChiMil94].

Die Existenz von Ursachenkombinationen, die die hier genannten Vollständigkeitskriterien erfüllen, ist nicht immer gewährleistet. So ist beispielsweise, wie in [ChiMil94] ausgeführt, eine Variation einzelner Ursacheinstanzen bei gleichzeitiger Maskierung der übrigen Instanzen derselben Ursache nicht immer möglich. Darüber hinaus ist die Konformität mit den Randbedingungen zu wahren, was eine weitere Einschränkung der möglichen Ursachenkombinationen mit sich bringt. Dennoch ist die Bewertung der Vollständigkeit der Ursachenkombinationen nach den oben genannten Kriterien sinnvoll: So werden Unvollständigkeiten in den Kombinationsforderungen sichtbar gemacht, die auf mögliche Lücken bei der Prüfung der Realisierung des Ursache-Wirkungs-Zusammenhangs im Prüfling hinweisen. Um diese Lücken zu schließen, können weitere Ursachenkombinationen gewählt werden. Falls dies nicht möglich ist, kann eventuell durch die Erfüllung eines weniger strengen Kriteriums eine akzeptable Prüfung des Ursache-Wirkungs-Zusammenhangs erreicht werden. Das verwendete Vollständigkeitskriterium ist entsprechend der Kritikalität des geplanten Einsatzes des Prüflings zu wählen.

4.2.2.2.3 Kombination mit dem strukturorientierten Bedingungsüberdeckungstest

Die Auswahl der Ursachenkombinationen nach der hier beschriebenen klassischen und erweiterten Ursache-Wirkungs-Analyse orientiert sich vollständig an den aus der Spezifikation abgeleiteten kausalen Zusammenhängen. So werden die Ursachenkombinationen mit dem Ziel ausgewählt, die Realisierung der kausalen Zusammenhänge systematisch zu prüfen und Fehlermaskierungen zu vermeiden. Die Systematik der Prüfung wird dabei vollständig an Gestalt und syntaktischem Aufbau der spezifizierten Ursache-Wirkungs-Zusammenhänge orientiert, ihre Realisierung im Prüfling wird nicht betrachtet. Sie kann nur dann eine effiziente Prüfung garantieren, wenn die Realisierung in weiten Teilen der spezifizierten Struktur entspricht.

Die Realisierung von kausalen Zusammenhängen der Spezifikation führt im Prüfling im Allgemeinen zur Verwendung von Prädikaten, die in komplexen Bedingungen analysiert werden und im dynamischen Test den Kontrollfluss des Prüflings und damit die realisierte Wirkung beeinflussen. Spezifizierte Ursachen fließen so in Bedingungen der Realisierung ein, während eine Entsprechung der Wirkungen in den Pfadbereichen [How76] gesehen werden kann. Strebt man eine vollständige Analyse der kausalen Zusammenhänge für den Prüfling an, so ist es daher sinnvoll, neben der Herleitung der Kriterien für die dynamischen Testfälle aus der Spezifikation auch Kriterien im Hinblick auf die im Prüfling realisierten komplexen Bedingungen aufzustellen. Hierfür bieten sich insbesondere die Kriterien der einfachen Bedingungsüberdeckung, der minimal mehrfachen Bedingungsüberdeckung, der mehrfachen Bedingungsüberdeckung [Mye79] sowie die modifizierte Bedingungs- und Entscheidungsüberdeckung [ChiMil94] an. Wie in Abschnitt 4.2.2.2.2 bereits festgestellt, ist letztere besonders gut mit den Zielen der Ursache-Wirkungs-Analyse vereinbar: Auch hier ist es das Ziel, eine systematische Prüfung der Auswirkung einzelner atomarer Bedingungen auf den Gesamtausgang der komplexen Bedingung mit einem vertretbaren, linearen Aufwand zu erreichen. Eine Vermeidung von Fehlermaskierungen wird dabei nur im Hinblick auf die komplexe Bedingung selbst erreicht, sie wirkt sich aller-

dings auf die Zuordnung des Pfadbereichs aus. Ein dynamischer Test im Hinblick auf Struktur und Spezifikation des Prüflings ist somit sinnvoll, um sowohl die spezifizierten als auch die realisierten kausalen Zusammenhänge bei der Prüfung zu berücksichtigen.

4.2.2.2.4 Kombination mit der Grenzwertanalyse

Nach Definition 4-9 und Definition 4-10 werden Ursachen als Prädikate über dem Ein- und Ausgabebereich verstanden, die Eigenschaften der Ein- beziehungsweise Ausgabedaten beschreiben. Wie in der Diskussion zu den Definitionen bereits beschrieben, bestimmen die Ursachen- und Wirkungsprädikate Partitionierungen des Ein- beziehungsweise Ausgabedatenbereichs, deren Elemente als funktional äquivalent anzusehen sind. Die dynamischen Testverfahren auf Basis einer Entscheidungstabelle oder auf Basis einer Ursache-Wirkungs-Analyse fordern nun eine spezielle Auswahl von dynamischen Testfällen aus den Partitionen beziehungsweise ihren Schnittmengen. Sie können somit als Erweiterungen des dynamischen Tests auf Basis einer funktionalen Partitionierung aufgefasst werden, die zusätzlich zur Prüfung einzelner Partitionen des Ein- und Ausgabebereichs auch noch die Prüfung gewisser Kombination vorgeben. Vor diesem Hintergrund bietet sich bei der Testdatenselektion eine zusätzliche Berücksichtigung des Prinzips der Grenzwertbetrachtung entsprechend Verfahren 4-5 oder Verfahren 4-6 an. Kritisch ist dabei die Anzahl der Testfälle zu beobachten, die bei der Grenzwertanalyse sehr umfangreich sein kann.

4.2.2.2.5 Erweiterung der Begriffsdefinitionen zu Ursache und Wirkung

Das in den vorigen Abschnitten beschriebene Verfahren der Ursache-Wirkungs-Analyse fasst Ursachen und Wirkungen prädikativ als Eigenschaften über dem Ein- und Ausgabebereich auf. Diese Auffassung kann erweitert und ergänzt werden.

So wird beispielsweise in [Lig02] darauf hingewiesen, dass eine Wirkung auch eine Systemtransformation bedeuten kann. Eine solche Wirkung ist sinnvoll, wenn Spezifikation und Prüfling zustandsbehaftet sind, wie beispielsweise in Abschnitt 4.4 diskutiert werden wird. Eine Wirkung kann in diesem Fall einen Zustandsübergang des Systems bedeuten. Auch in der Ursachenbeschreibung können Informationen über den Zustand des Systems aufgenommen werden. Wird eine Beschreibung der Ursachen und Wirkungen im Rahmen der Spezifikation eines zustandsbehafteten Systems gegeben, kann die Ursache-Wirkungs-Analyse auch in diesem Kontext sinnvoll angewendet werden. Die Kombinationsmöglichkeiten der Verfahren der Ursache-Wirkungs-Analyse mit den zustandsbasierten Verfahren werden in Abschnitt 4.4.4.2 diskutiert.

Weiterhin kann im Spezifikationstext im Zusammenhang mit einer Wirkung eine Anforderung an die Performanz formuliert sein. In diesem Fall kann die Wirkung als Eigenschaft über dem kombinierten Ausgabe- und Performanzbereich $O \times \Re^+$ aufgefasst und prädikativ formuliert werden. Aus Definition 4-10 ergibt sich damit die folgende Beschreibung einer Wirkung über dem Ausgabe- und Performanzbereich.

Definition 4-18: Wirkung über dem Ausgabe- und Performanzbereich

Eine Wirkung wird beschrieben durch eine semantische Einheit im Spezifikationstext, die als Eigenschaft von Elementen des Ausgabe- und Performanzbereichs $O \times \Re^+$ interpretiert werden kann und die Resultat eines spezifizierten Systemverhaltens ist. Sie kann formal als einstelliges Prädikat

$\forall((o, r) \in O \times \mathfrak{R}^+)$
(w(o, r) :⇔ (Element (o, r) hat die Eigenschaften der beschriebenen Wirkung))
über dem Bereich $O \times \mathfrak{R}^+$ aufgefasst werden.

Es ist ebenfalls möglich, Wirkungen allein im Performanzbereich zu beschreiben. Dies ist der Fall, wenn Performanzanforderungen an eigenen Ursachen festgemacht werden. In diesem Fall kann das zugehörige Prädikat über $O \times \mathfrak{R}^+$ mit Indifferenz gegenüber dem Ausgabedatum formuliert werden.

Das Verfahren der Ursache-Wirkungs-Analyse kann auf dieser Basis in herkömmlicher Weise Ursachenkombinationen für die Wirkung über dem Ausgabe- und Performanzbereich ableiten. Es sind keine Modifikationen in der Verfahrensbeschreibung notwendig. Bei der Durchführung des dynamischen Tests ist der Eintritt der Wirkungen im Hinblick auf Ausgabedatum und Performanz zu untersuchen. Entsprechend können auch beim dynamischen Test auf Basis einer Entscheidungstabelle Wirkungen berücksichtigt werden, die Performanzanforderungen beinhalten.

4.2.2.2.6 Betrachtung temporallogischer Zusammenhänge

Die klassische Ursache-Wirkungs-Analyse beschreibt die Ableitung kausaler Zusammenhänge aus einer informal beschriebenen Spezifikation. Häufig sind jedoch in einer Spezifikation nicht nur kausale, sondern auch kombiniert kausal-temporale Ursache-Wirkungs-Zusammenhänge beschrieben. Dies ist insbesondere bei technischen Systemen der Fall, wo neben den logischen Zusammenhängen auch der zeitliche Verlauf eine wichtige Rolle spielt und Lebendigkeits- und Sicherheitseigenschaften garantiert werden müssen. Bei der Formulierung solcher Eigenschaften können temporale Logiken wie beispielsweise CTL*, CTL oder LTL verwendet werden [Pnu77, Eme90, ManPnu95]. Diese erlauben es, aussagenlogische Formulierungen mit einer zeitlichen Darstellung in einer linearen oder baumartigen Zeitstruktur zu verbinden. Sehr viel häufiger als diese formale Gestaltung der Spezifikation wird in der Praxis eine natürlichsprachliche Darstellung der temporallogischen Zusammenhänge in Form eines Fließtexts gegeben. Um die Vorzüge einer formalen Darstellung in einer geeigneten Logik mit der Verständlichkeit natürlichsprachlicher Formulierungen zu verbinden, wurden semiformale Beschreibungsformen wie beispielsweise „Safety Pattern“ [Bit01] entwickelt.

Die Spezifikation mit temporallogischen Methoden ist insbesondere für reaktive Systeme geeignet. Um den dynamischen Test eines solchen Systems zu beschreiben, muss das in Kapitel 3 beschriebene Modell erweitert werden, da hier mit Annahme 3-1 eine Ein-Schritt-Verarbeitungssemantik zugrunde gelegt wurde. Möglich ist jedoch, wie Abschnitt 4.4 zeigen wird, eine Formulierung des dynamischen Tests zustandsbasierter Systeme im Rahmen des Modells. Da für reaktive Systeme in der Praxis häufig eine zustandsbasierte Realisierung gewählt wird und das Verfahren der Ursache-Wirkungs-Analyse auch für zustandsbasierte Systeme anwendbar ist, kann eine Erweiterung des Verfahrens im Hinblick auf temporallogische Anforderungen der Spezifikation insbesondere für diese Systeme sinnvoll sein.

Im dynamischen Test eines zustandsbasierten Systems mit temporallogischen Anforderungen muss eine systematische Prüfung der Zusammenhänge zwischen Ursachen und ihren Wirkungen erreicht werden. Diese Zusammenhänge können allein mit logischen Operatoren, aber auch unter Zuhilfenahme von temporalen Operatoren und Pfadquantoren beschrieben sein. Zur Berücksichtigung dieser Zusammenhänge sind einige Anpassungen vorzunehmen. So eignet sich die in der klassischen Ursache-Wirkungs-Analyse verwendete graphische Beschreibungssprache nicht zur Beschreibung tempo-

rallogischer Zusammenhänge. Die hier gewählte Darstellung mit Hilfe logischer Ausdrücke über einem Alphabet von Ursachen- und Wirkungsbezeichnern kann jedoch auf eine temporale Logik ausgeweitet werden. Dies führt zu einer Erweiterung des Ursache-Wirkungs-Modells in Definition 4-16. Kern der Erweiterung der Ursache-Wirkungsanalyse für temporallogische Aspekte muss jedoch eine Anpassung der Ableitung der Ursachenkombinationsforderungen sein. Hier müssen neben den aussagenlogischen Operatoren alle weiteren Operatoren der verwendeten Logik berücksichtigt werden. Dies führt zu Ursachenkombinationsforderungen für den dynamischen Test, die ebenfalls mit Operatoren der erweiterten Logik aufgebaut sein können. Da ihre Erfüllung nicht anhand der Einzelbetrachtung dynamischer Testfälle, die in diesem Kontext einzelne Ereignisse darstellen, geprüft werden kann, ist hier die Testdatenselektion und –adäquatheit im Hinblick auf dynamische Testsequenzen zu formulieren, die in Definition 4-46 eingeführt werden.

Bei der Formulierung eines dynamischen Testverfahrens zur Prüfung temporallogischer Anforderungen in einem zustandsbehafteten System ist immer zu beachten, dass seine Aussagekraft nicht über die einer stichprobenartigen Prüfung hinausgehen wird. Aussagen allgemeiner Gültigkeit können nur über verifizierende Verfahren erreicht werden. Im oben beschriebenen Kontext bietet sich insbesondere das Model Checking an, das in zustandsbasierten Systemen einen Nachweis von temporallogisch spezifizierten Anforderungen ermöglicht [ClaGruPel00]. Die Notwendigkeit eines dynamischen Tests kann bei Nachweis der Anforderungen als Eigenschaften des Prüflings entfallen. Ist ein vollständiger Nachweis der Eigenschaften nicht möglich, was bei Systemen mit großen internen Zustandsräumen und Systemen mit regelungstechnischen Anteilen beispielsweise der Fall sein kann, muss eine Qualitätssicherung mit Verfahren des dynamischen Tests durchgeführt werden. Hierbei kann ein Verfahren, wie es in den vorangegangenen Absätzen skizziert wurde, sinnvoll unterstützen.

4.3 Verfahren zum Test von graphisch spezifizierter Funktionalität

Häufig werden in Spezifikationen graphische Beschreibungsformen verwendet, die die Funktionalität des Prüflings darstellen. Grund hierfür ist die intuitive Verständlichkeit der graphisch dargestellten Sachverhalte, die durch Informationen in Textform ergänzt werden können. So ermöglichen Graphen eine einfache Repräsentation von komplexen, beispielsweise vernetzten Strukturen, die in einem Text nur schwer, nämlich als Sequenz und unter Verwendung von Bezugnahmen, dargestellt werden können. Diese intuitive Eingängigkeit gepaart mit der Darstellungsmöglichkeit komplexer Sachverhalte erleichtert den Formalisierungsprozess der Softwareentwicklung, indem sie die kognitiven Strukturen des menschlichen Gehirns nutzt und unterstützt. So kommt den graphischen Spezifikationstechniken ein zunehmender Stellenwert in der Softwareentwicklung zu, beispielsweise in der objektorientierten Modellierung. Bei der Verwendung graphischer Spezifikations- und Modellierungstechniken ist immer darauf zu achten, dass die Semantik der verwendeten Darstellungsform möglichst eindeutig beschrieben ist. Dies ist notwendig, um Missverständnisse bei der oft schnellen, sehr intuitiven Interpretation der Darstellung auszuschließen und Irrtümer in der Systementwicklung zu vermeiden. Freiheitsgrade bei der Spezifikation und Modellierung dürfen daher nicht durch fehlende Semantik, sondern nur in der Wahl des Abstraktionsgrads entstehen.

Graphische Beschreibungsformen und Modellierungsansätze werden seit langem verwendet, ihre Ausprägung hängt stark von ihrem Einsatzgebiet ab. So werden Graphen zur Beschreibung von Grammatiken bei der Definition formaler Sprachen verwendet [Bei90]. Graphische Beschreibungsformen werden zur Darstellung von Aktionen oder Zuständen eines Systems oder zur Visualisierung von

Kommunikationsflüssen verwendet. Auch komponentenbasierte Sichten auf ein System können graphisch spezifiziert werden, wobei die Verteilung von Aufgaben und Arbeitschritten und die Kommunikation zwischen den Komponenten und zur Außenwelt dargestellt werden können. Einige dieser Modellierungsformen, die sich insbesondere in Spezifikation und Design von objektorientierten Systemen bewährt haben, wurden in der Unified Modeling Language (UML) vereinheitlicht [BooRumJac98, Oes98, OMG03]. Auch Petri-Netze, die eine verhaltensorientierte Beschreibung eines Systems geben, stellen eine graphische Spezifikationsform dar [Pet62, Pet81, FraSie91, AjmEA95].

Viele graphische Spezifikationstechniken beruhen auf graphentheoretischen Beschreibungen, die mit einer speziellen Semantik hinterlegt sind. Um diese Beschreibungsformen adäquat in einem funktionsorientierten Test berücksichtigen zu können, wird im Folgenden eine allgemeine Form des Testens von graphisch spezifizierter Funktionalität eingeführt. Diese bezieht sich auf gerichtete Graphen, die als Grundform in Abschnitt 4.3.1 eingeführt werden. In Abschnitt 4.3.2 werden dynamische Testverfahren eingeführt, die sich an den Strukturelementen dieser Grundform orientieren. Die Anwendung und Erweiterung dieser Verfahren auf spezielle Formen der graphischen Spezifikation wie Syntaxgraphen, Petri-Netze, Aktivitätsdiagramme und Sequenzdiagramme werden Abschnitt 4.3.3 beschrieben.

In der Diskussion des funktionsorientierten Tests graphischer Spezifikationen wird zunächst vorausgesetzt, dass die Graphen lediglich die in Definition 3-2 eingeführte relationale Beschreibung der geforderten Zuordnung zwischen Ein und Ausgabedaten formulieren oder veranschaulichen. Dies ist von den zustandsbasierten Beschreibungsformen zu unterscheiden, die einzelne Zustandsübergänge eines Systems als Resultat separater Ausführungen spezifizieren. Als einfaches Unterscheidungskriterium kann die Ausführungssemantik des spezifizierten Graphen analysiert werden: Ist es möglich, in einem dynamischen Testfall einen Weg über mehrere Kanten zwischen einem Start- und Zielknoten zurückzulegen, so unterstützt der Graph die Beschreibung der Ein-/Ausgaberelation. Im dynamischen Test sind die im Folgenden beschriebenen Verfahren anzuwenden. Kann in einem dynamischen Testfall lediglich eine einzelne Kante durchlaufen werden, deren Anfangs- und Endknoten den Zustand des Systems vor und nach Ausführung des dynamischen Testfalls repräsentiert, so liegt eine zustandsbasierte Beschreibung vor. Zustandsbasierte Beschreibungsformen und die für sie adäquaten dynamischen Testverfahren werden in Abschnitt 4.4 vorgestellt.

4.3.1 Spezifikation mit Hilfe von Graphen

Vielen graphischen Spezifikationsformen liegen graphentheoretische Strukturen zugrunde. Im Folgenden wird die häufig verwendete Grundstruktur eines endlichen gerichteten Graphen mit einfachen Kanten vorausgesetzt. Bei Bedarf können die Ergebnisse und Verfahren auch für erweiterte graphentheoretische Strukturen wie endliche Multi- oder Hypergraphen angepasst werden. Die folgenden Definitionen führen Begriffe und Bezeichnungen aus der Graphentheorie ein, die in den folgenden Notationen verwendet werden.

Definition 4-19: Endlicher gerichteter Graph, Kanten- und Knotenmenge, Weg, Zyklus, Kreis, Länge eines Weges, Knoten eines Weges, Anzahl der Vorkommen von Knotenfolgen in einem Weg

(a) Als endlicher gerichteter Graph g wird ein Tupel (V, E) bezeichnet, das aus einer endlichen Knotenmenge V und einer Menge von gerichteten Kanten $E \subseteq V \times V$ zwischen diesen Knoten besteht. Knoten(g) bezeichnet die Knotenmenge V des Graphen, Kanten(g) die Kantenmenge E.

(b) Für eine Kante $e \in E$ mit $e = (v_1, v_2)$ bezeichnet $anf(e) = v_1$ den Anfangsknoten, $end(e) = v_2$ den Endknoten.

(c) Eine Folge $(v_1, v_2, \ldots, v_k)$ $(k \in \aleph \setminus \{ 0 \})$ von Knoten eines Graphen $g = (V, E)$, die der Bedingung

$$\forall(l \in \{ 1, \ldots., k\text{-}1 \}) ((v_l, v_{l+1}) \in E)$$

genügt, wird als Weg der Länge k bezeichnet.

(d) Ein Weg $(v_1, v_2, \ldots, v_k)$ der Länge k $(k \in \aleph \setminus \{ 0, 1 \})$ im Graphen $g = (V, E)$, der der Bedingung

$$(v_1 = v_k)$$

genügt, wird als Zyklus bezeichnet.

(e) Gilt für einen Zyklus $(v_1, v_2, \ldots, v_k)$ im Graphen $g = (V, E)$ die Bedingung

$$\forall(l \in \{ 1, \ldots., k\text{-}1 \}) \forall(m \in \{ 1, \ldots., k\text{-}1 \}) ((l \neq m) \Rightarrow (v_l \neq v_m)),$$

so wird er als Kreis bezeichnet. In einem endlichen gerichteten Graphen existieren nur endlich viele verschiedene Kreise.

(f) Zu einem Weg $w = (v_1, v_2, \ldots, v_k)$ im Graphen $g = (V, E)$ bezeichnet

$$Länge(w) := k$$

die Länge.

(g) Zu einem Weg $w = (v_1, v_2, \ldots, v_k)$ im Graphen $g = (V, E)$ bezeichnet

$$w_l := v_l \qquad (l \in \{ 1, \ldots, Länge(w) \})$$

den l-ten Knoten.

(h) Im Graphen $g = (V, E)$ gibt die Funktion $Anz(w, v)$ zu einem Weg $w = (v_1, v_2, \ldots, v_k)$ der Länge k $(k \in \aleph \setminus \{ 0 \})$ und einer Knotenfolge $v = (v_1, v_2, \ldots, v_l)$ der Länge l $(l \in \aleph \setminus \{ 0 \})$ die Anzahl der Vorkommen der Teilfolge v im Weg w an.

Meist sind in den in Spezifikationen verwendeten Typen von Graphen ein Startknoten und eventuell mehrere Zielknoten gekennzeichnet. In diesem Fall sind speziell die Wege von Interesse, die vom Startknoten zu einem Zielknoten führen.

Definition 4-20: Endlicher gerichteter Graph mit Start- und Zielknoten, Pfad, Pfade mit höchstens n-fachem Durchlauf der enthaltenen Kreise $(n \in \aleph)$, pfadzusammenhängender Graph

(a) Ein endlicher gerichteter Graph (V, E) kann durch die Angabe eines Startknoten $v_{start} \in V$ und einer Zielknotenmenge $V_{ziel} \subseteq V$ ergänzt werden. Das Quadrupel $(V, E, v_{start}, V_{ziel})$ wird in diesem Fall als endlicher gerichteter Graph mit Start- und Zielknoten bezeichnet.

(b) In einem endlichen gerichteten Graphen mit Start- und Zielknoten $g = (V, E, v_{start}, V_{ziel})$ wird ein Weg $(v_1, v_2, \ldots, v_k)$ der Länge k $(k \in \aleph \setminus \{ 0 \})$ als Pfad der Länge k bezeichnet, wenn die Bedingung

$$(v_1 = v_{start}) \wedge (v_k \in V_{end})$$

erfüllt ist.

(c) Die Menge aller Pfade durch einen endlichen gerichteten Graphen mit Start- und Zielknoten g wird im Folgenden mit

Pfade(g)

bezeichnet.

(d) Die Menge aller Pfade zu einem endlichen gerichteten Graphen mit Start- und Zielknoten $g = (V, E, v_{start}, V_{ziel})$, die alle enthaltenen Kreise höchstens n Mal ($n \in \aleph$) enthalten, wird im Folgenden mit

$Pfade_n(g)$

bezeichnet. $Pfade_0(g)$ bezeichnet somit die Menge aller zyklenfreien Pfade.

(e) Ein endlicher gerichteter Graph mit Start- und Zielknoten $g = (V, E, v_{start}, V_{ziel})$ wird als pfadzusammenhängend bezeichnet, wenn alle Knoten Teil eines Pfades sind, wenn also die Bedingung

$$\forall(v \in V)\ \exists(w \in Pfade(g))\ \exists(k \in \aleph \setminus \{ 0 \})\ (w_k = v)$$

gilt. Ein pfadzusammenhängender endlicher gerichteter Graph mit Start- und Zielknoten wird im Folgenden abkürzend als pfadzusammenhängender Graph bezeichnet.

Mit diesen Definitionen stehen die notwendigen Begriffe zur Beschreibung und Analyse graphentheoretischer Aspekte von Spezifikationen und zu ihrer Behandlung im dynamischen Test bereit.

4.3.2 Verfahren zum dynamischen Test auf Basis von Graphen

Zur Untersuchung des dynamischen Tests wird im Folgenden die Beschreibung der Spezifikation in Form eines oder mehrerer pfadzusammenhängender Graphen vorausgesetzt. Um die in einem dynamischen Testfall oder Testlauf geprüfte Funktionalität vor dem Hintergrund der Strukturelemente des Graphen beschreiben zu können, wird eine Assoziation zwischen den dynamischen Testfällen und den Pfaden angenommen. Die Gestalt dieser Assoziation wird an dieser Stelle nicht näher definiert, da sie von der Semantik der verwendeten Spezifikationsform abhängt. Dies verdeutlicht eine kurze Betrachtung verschiedener Spezifikationstechniken: So wird im Falle eines Syntaxgraphen eine syntaktische Einheit als Eingabedatum übergeben, deren Spezifikationskonformität und Eigenschaften sukzessive beim Durchlaufen des Graphen geprüft werden. Der Pfad durch den Syntaxgraphen ergibt sich durch sukzessive Analyse und Zerlegung der syntaktischen Einheit. Ein Aktivitätsdiagramm hingegen spezifiziert die Verarbeitungsschritte eines Eingabedatums. Pfade sind von den Eigenschaften des Eingabedatums bestimmt, die in den Transitionsbedingungen des Diagramms referenziert werden. Pfade durch ein Petri-Netz im Anschluss an die Stimulation des Systems mit einer Startmarkierung werden durch den Markenfluss beschrieben, der von netzinternen logischen, möglicherweise auch stochastischen Eigenschaften abhängt. Um die vielfältigen Assoziationsmöglichkeiten zwischen den dynamischen Testfällen und den Pfaden nicht einzuschränken, wird die Assoziation zwischen einem Eingabedatum und einer Menge von Pfaden durch den Graphen daher zunächst abstrakt mit Hilfe der folgenden Zuordnung beschrieben.

Definition 4-21: Pfadassoziation zu einem Eingabedatum

Vorausgesetzt wird die Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch einen pfadzusammenhängenden Graphen $g = (V, E, v_{start}, V_{ziel})$.

Ein Pfad $(v_1, v_2, \ldots, v_k)$ durch den Graphen wird genau dann als mit einem Eingabedatum $i \in I$ assoziiert bezeichnet, wenn laut Spezifikation für das Eingabedatum die durch den Pfad beschriebene Funktionalität anzuwenden ist. Dies wird durch die Zuordnung

$$\begin{array}{lccl} PfadAssoz_s: & I & \rightarrow & \wp(Pfade(g)) \\ & i & \mapsto & PfadAssoz_s(i) \end{array}$$

ausgedrückt, die jedem Eingabedatum die Menge der mit ihm assoziierten Pfade zuweist. Die Zuordnung wird im Folgenden als Pfadassoziation bezeichnet.

Mit Hilfe der Pfadassoziation kann die Überdeckung der Strukturelemente des spezifizierten Graphen im dynamischen Test beschrieben werden.

Definition 4-22: Überdeckung der Strukturelemente eines pfadzusammenhängenden Graphen durch einen dynamischen Testfall

Vorausgesetzt wird die Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch einen pfadzusammenhängenden Graphen $g = (V, E, v_{start}, V_{ziel})$.

(a) Die Überdeckung $pcov_s \subseteq I \times Pfade(g)$ von Pfaden des Graphen g durch Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(w \in Pfade(g))$$
$$((i, w) \in pcov_s) :\Leftrightarrow (w \in PfadAssoz_s(i))$$

über der Menge der Eingabedaten und der Pfade durch den Graphen.

(b) Die Überdeckung $ecov_s \subseteq I \times E$ von Kanten des Graphen g durch Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(e \in E)$$
$$(((i, e) \in ecov_s) :\Leftrightarrow$$
$$(\exists(w \in PfadAssoz_s(i))\ \exists(k \in \{ 1, \ldots, Länge(w)-1 \})\ (e = (w_k, w_{k+1}))))$$

über der Menge der Eingabedaten und der Kanten des Graphen.

(c) Die Überdeckung $vcov_s \subseteq I \times V$ von Knoten des Graphen g durch Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(v \in V)$$
$$(((i, v) \in vcov_s) :\Leftrightarrow$$
$$(\exists(w \in PfadAssoz_s(i))\ \exists(k \in \{ 1, \ldots, Länge(w) \})\ (v = w_k)))$$

über der Menge der Eingabedaten und der Knoten des Graphen.

Ein Pfad durch einen Graphen wird nach dieser Definition genau dann durch einen Testfall überdeckt, wenn der Testfall mit dem Pfad entsprechend Definition 4-21 assoziiert ist. Alle Knoten und Kanten des Pfades werden in diesem Fall durch den Testfall überdeckt.

Durch die Gestaltung der Assoziation zwischen dynamischen Testfällen und Pfaden wird die Definition der Überdeckung rein spezifikationsbasiert an der Semantik des Graphen ausgerichtet: Die Überde-

ckung ist durch die für ein Eingabedatum im Graphen beschriebene Funktionalität festgelegt, nicht durch die tatsächliche Behandlung des Eingabedatums durch den Prüfling.

Der Begriff der Überdeckung der Strukturelemente des spezifizierten Graphen ist für die Beschreibung der dynamischen Testverfahren von zentraler Bedeutung. Mit seiner Hilfe können die dynamischen Testverfahren einfach auf Basis der Überdeckungsrelation beschrieben werden. Das folgende Verfahren beschreibt eine Überdeckung aller Knoten des Graphen im dynamischen Test.

Verfahren 4-9: Knotenüberdeckung eines pfadzusammenhängenden Graphen

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch einen pfadzusammenhängenden Graphen $g = (V, E, v_{start}, V_{ziel})$ sowie die Pfadassoziation $PfadAssoz_s$ zu den Eingabedaten.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Knotenüberdeckung eines pfadzusammenhängenden Graphen wird beschrieben durch

$$X^{Vcov}: \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(v \in V)\, \exists(t \in T)\, ((t, v) \in vcov_s).$$

Das Verfahren betrachtet somit alle dynamischen Testläufe, die jeden Knoten des Graphen mit mindestens einem dynamischen Testfall überdecken, als adäquat für den Test der im Graphen spezifizierten Funktionalität. Ebenso kann ein dynamisches Testverfahren definiert werden, das die Überdeckung aller Kanten im dynamischen Test fordert.

Verfahren 4-10: Kantenüberdeckung eines pfadzusammenhängenden Graphen

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch einen pfadzusammenhängenden Graphen $g = (V, E, v_{start}, V_{ziel})$ sowie die Pfadassoziation $PfadAssoz_s$ zu den Eingabedaten.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Kantenüberdeckung eines pfadzusammenhängenden Graphen wird beschrieben durch

$$X^{Ecov}: \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(e \in E)\, \exists(t \in T)\, ((t, e) \in ecov_s).$$

Durch dieses Verfahren werden dynamische Testläufe als adäquat betrachtet, die zu jeder Kante einen dynamischen Testfall enthalten, der diese überdeckt. Auch das dynamische Testverfahren zur Erreichung einer Pfadüberdeckung lässt sich entsprechend beschreiben.

Verfahren 4-11: Pfadüberdeckung eines pfadzusammenhängenden Graphen

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I,\,O}$ durch einen pfadzusammenhängenden Graphen $g = (V, E, v_{start}, V_{ziel})$ sowie die Pfadassoziation $PfadAssoz_s$ zu den Eingabedaten.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Pfadüberdeckung eines pfadzusammenhängenden Graphen wird beschrieben durch

$$\begin{array}{lllll} X^{Pcov}: & \Sigma_{I,\,O} & \rightarrow & \wp(\wp^{fin}(I)) \\ & s & \mapsto & \{\, T \in \wp^{fin}(I) : P_s(T) \,\} \end{array}$$

mit

$$P_s(T) \quad := \quad \forall (w \in Pfade(g))\ \exists (t \in T)\ ((t, w) \in pcov_s).$$

Enthält der Graph einen Kreis, so besitzt die Pfadmenge unendliche Mächtigkeit. Kein dynamischer Testlauf kann in diesem Fall das Verfahren der Pfadüberdeckung erfüllen, das Verfahren ist nicht anwendbar im Sinne der Definition 3-9. Das folgende Verfahren beschränkt die Forderung der Überdeckung auf alle diejenigen Pfade, die einen beliebigen Kreis nicht öfter als n Mal durchlaufen.

Verfahren 4-12: Überdeckung aller Pfade eines pfadzusammenhängenden Graphen, die alle enthaltenen Kreise höchstens n Mal durchlaufen ($n \in \aleph$)

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I,\,O}$ durch einen pfadzusammenhängenden Graphen $g = (V, E, v_{start}, V_{ziel})$ sowie die Pfadassoziation $PfadAssoz_s$ zu den Eingabedaten.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller Pfade eines pfadzusammenhängenden Graphen, die alle enthaltenen Kreise höchstens n Mal durchlaufen, wird beschrieben durch

$$\begin{array}{lllll} X^{P(n)Cov}: & \Sigma_{I,\,O} & \rightarrow & \wp(\wp^{fin}(I)) \\ & s & \mapsto & \{\, T \in \wp^{fin}(I) : P_s(T) \,\} \end{array}$$

mit

$$P_s(T) \quad := \quad \forall (w \in Pfade_n(g))\ \exists (t \in T)\ ((t, w) \in pcov_s).$$

Für n = 0 ist bei diesem Verfahren zu beachten, dass das Durchlaufen von Kreisen nicht gefordert wird.

Obwohl die Anzahl der Pfade in $Pfade_n(g)$ endlich ist, kann die Mächtigkeit dieser Menge immer noch sehr groß sein. Um eine adäquate Prüfung der durch die Kreise festgelegten Funktionalität mit einer schwächeren Forderung zu erreichen, kann das folgende Verfahren genutzt werden.

Verfahren 4-13: Überdeckung der bis zu n-fachen Ausführung aller Kreise eines pfadzusammenhängenden Graphen ($n \in \aleph \setminus \{ 0 \}$)

(a) Voraussetzungen:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I,\ O}$ durch einen pfadzusammenhängenden Graphen $(V, E, v_{start}, V_{ziel})$ mit $V_{ziel} \neq \{ v_{start} \}$ sowie die Pfadassoziation $PfadAssoz_s$ zu den Eingabedaten.

$Kreise(g) := \{ k_1, \ldots, k_l \}$ $(l \in \aleph)$ bezeichne die Menge aller Kreise im Graphen.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung der bis zu n-fachen Ausführung aller Kreise eines pfadzusammenhängenden Graphen wird beschrieben durch

$$X^{K(n)Cov}: \quad \Sigma_{I,\ O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(k \in K)\ \forall(m \in \{ 0, \ldots, n \})$$
$$\exists(t \in T)\ \exists(w \in PfadAssoz_s(t))\ (Anz(w, k) = m).$$

Zur Erfüllung dieses Verfahrens werden weitaus weniger Pfade benötigt als zur Erfüllung des Kriteriums in Verfahren 4-12: Hier genügt die Prüfung des m-fachen Durchlaufs eines beliebigen Kreises auf einem einzelnen Pfad, dort müssen alle Pfade, die den Kreis m-fach durchlaufen, geprüft werden. Graphen, bei denen der Startknoten und der einzige Zielknoten identisch sind, müssen von der Betrachtung mit diesem Verfahren ausgeschlossen werden, da für sie der 0-fache Durchlauf des Zyklus zwischen Start- und Zielknoten nicht abgedeckt werden kann. Dieser kann in einem angepassten Verfahren explizit von der Prüfung ausgeschlossen werden.

In pfadzusammenhängenden Graphen genügen dynamische Testläufe, die das Kriterium der Kantenüberdeckung erfüllen, auch immer dem Verfahren der Knotenüberdeckung. Das Verfahren X^{Ecov} der Kantenüberdeckung subsumiert somit das der Knotenüberdeckung X^{Vcov} nach Definition 3-10. Ebenso ist unter diesen Voraussetzungen klar, dass das Verfahren der Pfadüberdeckung X^{Pcov} das Verfahren der Kantenüberdeckung X^{Ecov} subsumiert. Dies leistet bereits ein Verfahren zur Überdeckung aller Pfade mit bis zu n-facher Ausführung aller Kreise $X^{P(n)Cov}$, falls $(n > 0)$ gilt. Für natürliche Zahlen m und n mit $(m < n)$ gilt die Subsumption des Verfahrens $X^{P(m)Cov}$ durch das Verfahren $X^{P(n)Cov}$, da die Menge der Pfade, die alle Zyklen bis zu m Mal durchlaufen, Teilmenge der Menge aller Pfade ist, die alle Zyklen bis zu n Mal durchlaufen. Mit der gleichen Begründung gilt die Subsumption eines Verfahrens $X^{K(m)Cov}$ durch das Verfahren $X^{K(n)Cov}$ für natürliche Zahlen m und n mit $(m < n)$. Ein Verfahren $X^{P(n)Cov}$ mit $(n > 0)$ subsumiert das entsprechende Verfahren $X^{K(n)Cov}$, da dieses nur einzelne Pfade zur Überdeckung der n-fachen Kreisausführung fordert, während $X^{P(n)Cov}$ die Prüfung aller Pfade mit bis zu n-facher Kreisausführung verlangt. Diese Subsumptionsbeziehungen sind in Abbildung 4-2 dargestellt. Ein einfacher formaler Nachweis kann, wie angedeutet, über die Teilmengenbeziehungen zwischen den zur Überdeckung geforderten Pfadmengen geführt werden.

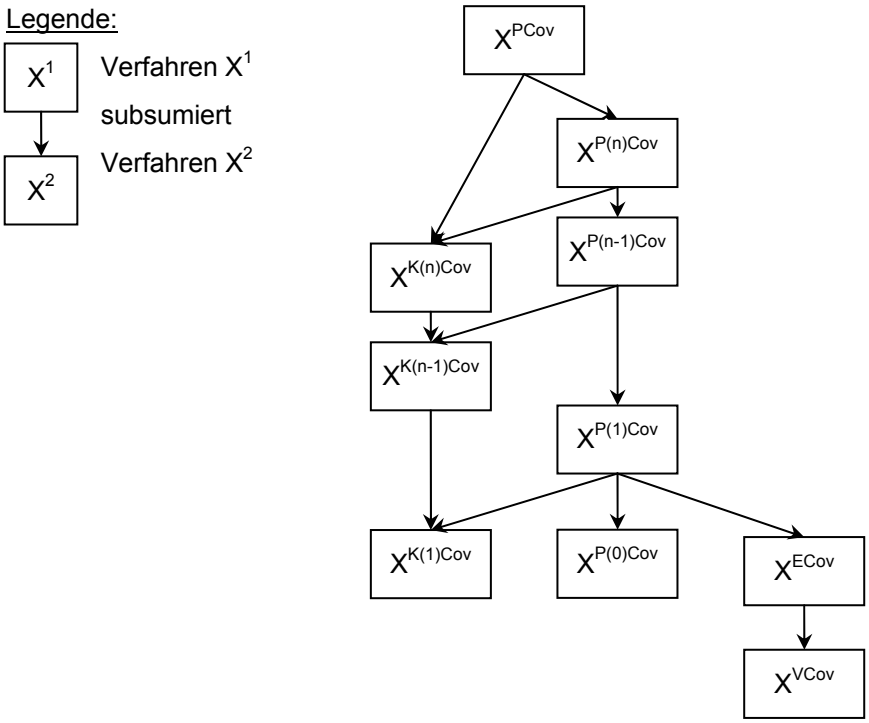


Abbildung 4-2: Subsumptionshierarchie für die Verfahren zur strukturellen Überdeckung von pfadzusammenhängenden endlichen gerichteten Graphen mit Start- und Endknoten ($n \in \mathbb{N}, n > 1$)

Die Anwendbarkeit der oben definierten Verfahren nach Definition 3-9 hängt neben der Endlichkeit der Menge der zu überdeckenden Pfade auch von der Pfadassoziation zu den Eingabedaten ab: Nur wenn Eingabedaten zu den zu überdeckenden Pfaden existieren, können die Kriterien in den dynamischen Testläufen erfüllt werden. Dies kann für spezielle Gestaltungen der Pfadassoziation garantiert werden, für manche jedoch nicht. So kann zu einem beliebigen Pfad durch einen Syntaxgraphen ein Eingabedatum konstruiert werden, dass zu diesem assoziiert ist. Dies wird in Abschnitt 4.3.3.1 deutlich. In Aktivitätsdiagrammen ist es hingegen möglich, dass – wie in 4.3.3.2 beschrieben – aufgrund widersprüchlicher Transitionsbedingungen für einen Pfad kein assoziiertes Eingabedatum existiert.

Die in diesem Abschnitt eingeführten Verfahren sind nicht als eigenständige Testverfahren zu betrachten; sie stellen lediglich Prototypen von bekannten Testverfahren dar, die auf speziellen Graphen operieren. Diese werden in den folgenden Abschnitten für verschiedene Formen der graphischen Spezifikation vorgestellt.

Bei geeigneter Anpassung der in Definition 4-21 beschriebenen Assoziation zwischen Eingabedaten und Pfaden können auch Graphen betrachtet werden, die nicht aus der Spezifikation abgeleitet sind: So kann beispielsweise der aus der Struktur des Prüflings abgeleitete Kontrollflussgraph [Lig02] als Grundlage der Überdeckungsrelationen in Definition 4-22 und damit als Grundlage der dynamischen Testverfahren verwendet werden. Die Pfadassoziation wird dabei durch die im Prüfling implementierten Pfadbedingungen vorgegeben. Die Überdeckung ist in diesem Fall nicht spezifikations-, sondern strukturabhängig; die resultierenden dynamischen Testverfahren sind strukturorientiert. Bezogen auf den Kontrollflussgraphen eines Prüflings entsprechen die hier vorgestellten dynamischen Testverfahren beispielsweise der Anweisungsüberdeckung, der Zweigüberdeckung, der Pfadüberdeckung beziehungsweise Varianten des strukturierten Pfadtests oder „boundary-interior“ Tests [How75, Tai80,

Lig02]. Sie ermöglichen somit eine sehr allgemeine Betrachtung des dynamischen Tests zur Überdeckung graphentheoretischer Strukturen.

4.3.3 Spezielle Ausprägungen graphischer Spezifikationen und ihr dynamischer Test

Im vorangegangenen Abschnitt wurden dynamische Testverfahren definiert, die auf die Überdeckung von Strukturelementen eines im Rahmen der Spezifikation verwendeten Graphen abzielen. Die Gestalt des Graphen wurde mit Hilfe allgemeiner graphentheoretischer Notationen beschrieben. Grundlage für die Definition der dynamischen Testverfahren war die Assoziation der Pfade des Graphen mit den Eingabedaten, die mit dem Ziel einer möglichst allgemeinen Definition zunächst nicht näher spezifiziert wurde. Diese Assoziation wird nun im Folgenden für spezielle graphische Spezifikationsformen beschrieben, ihre Bedeutung wird vor dem Hintergrund der zugehörigen Semantik diskutiert. Auf dieser Basis können bekannte dynamische Testverfahren, die auf graphischen Spezifikationen operieren, einfach beschrieben werden.

4.3.3.1 Syntaxgraphen

Syntaxgraphen werden für die Spezifikation von Parsern verwendet, also für die Spezifikation von Software zur Analyse syntaktischer Strukturen. Genutzt werden sie beispielsweise im Compilerbau und bei der Spezifikation von Benutzerschnittstellen zur Eingabe von Zeichenketten. Mit Hilfe von Syntaxgraphen können beliebige kontextfreie Grammatiken – auch als Typ 2 Grammatiken der Chomsky-Hierarchie bezeichnet – beschrieben werden. Syntaxgraphen sind somit äquivalent zu der für die Definition von kontextfreien Grammatiken häufig verwendeten, textuellen Backus-Naur-Form [Bac59]. Im dynamischen Test werden Syntaxgraphen als Grundlage für die Auswahl dynamischer Testfälle und die Bewertung ihrer Vollständigkeit verwendet [Bei90, Lig02]. Dieses Vorgehen wird im Folgenden mit Hilfe der im Absatz 4.3.2 eingeführten dynamischen Testverfahren beschrieben.

Zur Angabe einer kontextfreien Grammatik ist eine Festlegung des Eingabealphabets der beschriebenen Sprache notwendig, das im Folgenden in der Menge $Symb_{term}$ der terminalen Symbole zusammengefasst wird. Weitere zur Beschreibung der Grammatik verwendete Symbole werden nicht-terminale Symbole genannt und in eine Menge $Symb_{nterm}$ zusammengefasst, die zur Menge $Symb_{term}$ disjunkt sein muss. Für alle nicht-terminalen Symbole werden Produktionsregeln angegeben, die in kontextfreien Grammatiken unabhängig vom Kontext des Auftretens des nicht-terminalen Symbols sind. Formal können sie als Teilmenge des kartesischen Produktes

$$Prod \subseteq Symb_{nterm} \times (Symb_{nterm} \cup Symb_{term})^*$$

angegeben werden, wobei der Operator * die Konkatenation beliebiger Symbole der Menge $(Symb_{n\text{-}term} \cup Symb_{term})$ bezeichnet. Um eine auswertbare Grammatik zu erhalten, muss eins der nicht-terminalen Symbole als Startsymbol für die Auswertung gekennzeichnet sein. Die Wörter der Grammatik über $(Symb_{n\text{-}term} \cup Symb_{term})^*$ ergeben sich nun ausgehend vom Startsymbol durch Anwendung der Produktionsregeln. Die Anwendung der Produktionsregeln wird als Ableitung bezeichnet. Alle aus dem Startsymbol in endlich vielen Schritten ableitbaren Wörter werden als Wörter der Grammatik bezeichnet. Die Menge der Wörter über dem Alphabet der terminalen Symbole $(Symb_{term})^*$ wird als Sprache bezeichnet, die durch die kontextfreie Grammatik definiert ist. Die folgende Definition fasst die für das Verständnis der Syntaxgraphen wichtigen Aspekte kontextfreier Grammatiken und der zugehörigen Sprachen zusammen.

Definition 4-23: Kontextfreie Grammatik, Ableitung aus dem Startsymbol, Wörter einer kontextfreien Grammatik, Sprache einer kontextfreien Grammatik

(a) Eine kontextfreie Grammatik wird beschrieben durch ein Quadrupel

$$gr = (Symb_{term}, Symb_{nterm}, Prod, symb_{start}),$$

wobei $Symb_{term}$ die Menge der terminalen Symbole, $Symb_{nterm}$ die Menge der nicht-terminalen Symbole, Prod die Menge der Produktionsregeln und $symb_{start}$ das Startsymbol bezeichnet. Die Mengen $Symb_{nterm}$ und $Symb_{term}$ müssen disjunkt sein. Für die Menge

$$Prod \subseteq Symb_{nterm} \times (Symb_{nterm} \cup Symb_{term})^*$$

der Produktionsregeln muss die Vollständigkeitsbedingung

$$\forall(w \in Symb_{nterm})\ \exists(w' \in (Symb_{nterm} \cup Symb_{term})^*)\ ((w, w') \in Prod))$$

gelten, so dass für jedes nicht-terminale Symbol mindestens eine Produktionsregel enthalten ist. Für das Startsymbol gilt

$$symb_{start} \in Symb_{nterm}.$$

(b) Die Relation $\Rightarrow_{gr}$ über $(Symb_{nterm} \cup Symb_{term})^* \times (Symb_{nterm} \cup Symb_{term})^*$ wird durch

$$\forall(u \in (Symb_{nterm} \cup Symb_{term})^*)\ \forall(v \in (Symb_{nterm} \cup Symb_{term})^*)$$
$$((u \Rightarrow_{gr} v) :\Leftrightarrow$$
$$\exists(x \in (Symb_{nterm} \cup Symb_{term})^*)\ \exists(z \in (Symb_{nterm} \cup Symb_{term})^*)$$
$$\exists(y \in Symb_{nterm})\ \exists(y' \in (Symb_{nterm} \cup Symb_{term})^*)$$
$$((u = xyz) \wedge (v = xy'z) \wedge ((y, y') \in Prod)))$$

definiert und als Ableitung bezeichnet. Eine endliche Folge $(w_1, w_2, ..., w_k)$ $(k \in \mathbb{N} \setminus \{ 0 \})$ von Wörtern über $(Symb_{nterm} \cup Symb_{term})^*$, die der Bedingung

$$(w_1 = symb_{start}) \wedge (\forall(i \in \{ 1, 2, ..., k-1 \})\ (w_i \Rightarrow_{gr} w_{i+1}))$$

genügt, wird als Ableitung aus dem Startsymbol bezeichnet.

(c) Die Menge W_{gr} der Wörter der kontextfreien Grammatik über $(Symb_{nterm} \cup Symb_{term})^*$ besteht aus allen Wörtern, für die eine Ableitung aus dem Startsymbol angegeben werden kann.

(d) Die Menge

$$L_{gr} := (Symb_{term})^* \cap W_{gr}$$

wird als Sprache der kontextfreien Grammatik gr bezeichnet.

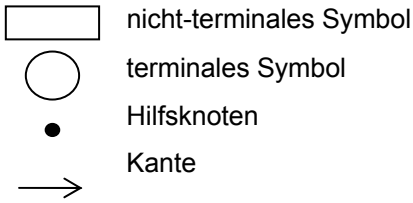
Eine knappe und übersichtliche Einführung in das Thema der formalen Sprachen wird beispielsweise in [Sch01] gegeben.

Für die Angabe der Produktionsregeln wurden verschiedene Formalismen entwickelt. Sie können beispielsweise in Backus-Naur-Form [Bac59], erweiterter Backus-Naur-Form oder in verwandten textuellen Darstellungen angegeben werden [Sch01]. Graphisch können sie mit Hilfe von Syntaxgraphen [Bei90] beschrieben werden, in denen die Symbole in den Knoten und ihre mögliche Konkatenation durch gerichtete Kanten zwischen den Knoten angegeben werden. Die Syntaxgraphen beschreiben durch ihre rechte Seite Produktionsregeln und müssen daher immer einem nicht-terminalen Symbol, beispielsweise dem Startsymbol, zugeordnet sein. Die Anwendung einer mit Hilfe eines Syntaxgraphen beschriebenen Produktionsregel geschieht durch Konkatenation der Symbole der auf einem Pfad durch den Graphen besuchten Knoten. Mögliche Verzweigungen im Syntaxgraphen führen zur

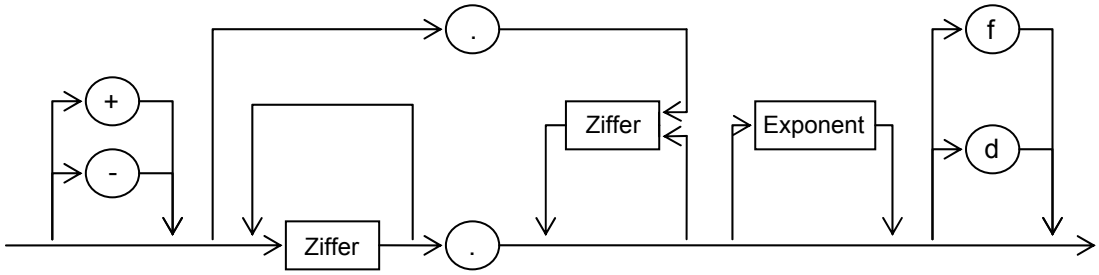
Ableitung mehrerer Produktionsregeln. Typische Darstellungsformen für Syntaxgraphen können beispielsweise in [Bei90] und [Lig02] eingesehen werden.

Damit für den dynamischen Test der durch einen Syntaxgraph beschriebenen Funktionalität die Verfahren des vorangegangenen Abschnitts verwendet werden können, muss der Syntaxgraph als gerichteter Graph aufgefasst werden. Hierfür sind Anpassungen erforderlich, die die Semantik des Syntaxgraphen nicht verändern dürfen. Größter Unterschied zwischen Syntaxgraphen und gerichteten Graphen ist die Verwendung der Kanten. So werden Kanten in gerichteten Graphen nur zwischen zwei Knoten erlaubt, wohingegen in Syntaxgraphen die Kanten auch, wie die Darstellungen in [Bei90] und [Lig02] zeigen, an beliebigen Stellen anderer Kanten beginnen oder enden können. Die Auswertung eines Syntaxgraphen beginnt an einer Kante, die keinen Anfangsknoten hat und auch nicht an einer anderen Kante beginnt. Das Ende der Auswertung ist an einer Kante ohne Endknoten oder weiterführende Kante erreicht. Um einen Syntaxgraphen als gerichteten Graphen zu notieren, sind alle enthaltenen Kanten dort zu unterbrechen, wo andere Kanten beginnen oder enden. An diesen Stellen sind Knoten zu ergänzen, die keinem Symbol entsprechen und daher als Hilfsknoten bezeichnet werden. Ferner werden Hilfsknoten als Start- und Zielknoten am Anfang der Startkante beziehungsweise am Ende der Zielkante ergänzt. Auch die terminalen und nicht-terminalen Symbole des Syntaxgraphen werden als Knoten betrachtet. Ein so ergänzter Syntaxgraph kann als gerichteter Graph gemäß Definition 4-19 aufgefasst werden. Die Auswertung der im Syntaxgraphen beschriebenen Produktionsregeln kann nun durch Analyse der Pfade vom Start- zum Zielknoten und durch Konkatenation der Symbole der besuchten Knoten werden. Hilfsknoten werden nicht beachtet, so dass die Semantik des Graphen unverändert bleibt. Die graphentheoretische Beschreibung eines Syntaxgraphen wird in Abbildung 4-3 beispielhaft in Anlehnung an eine Darstellung aus [Lig02] gezeigt.

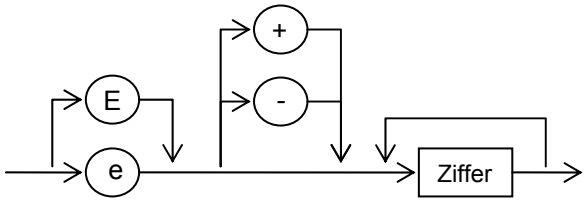
Legende:



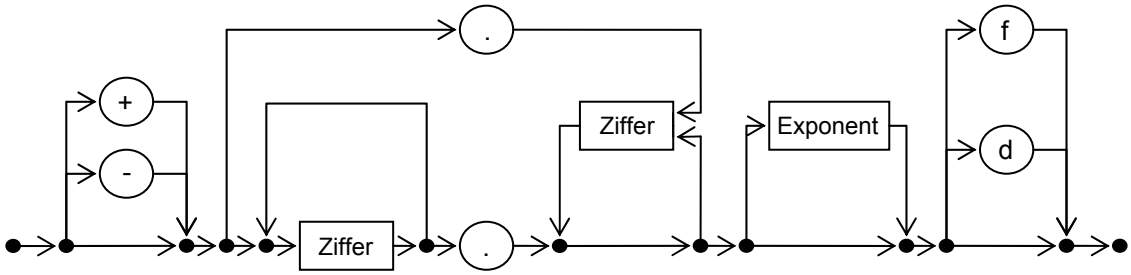
Syntax von Java-Gleitkommazahlen - Darstellung als Syntaxdiagramm:



Syntax von Java-Exponenten - Darstellung als Syntaxdiagramm:



Syntax von Java-Gleitkommazahlen - Darstellung als gerichteter Graph:



Syntax von Java-Exponenten - Darstellung als gerichteter Graph:

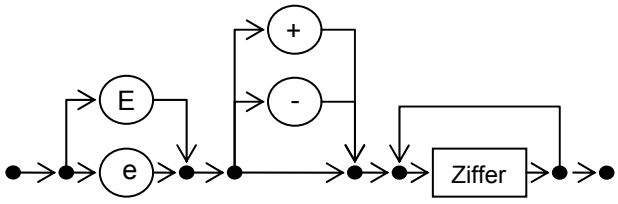


Abbildung 4-3: Überführung von Syntaxgraphen in eine graphentheoretische Darstellungsform

Diese Ergänzung ermöglicht die folgende Beschreibung eines Syntaxgraphen mit Hilfe einer graphentheoretischen Struktur.

Definition 4-24: Syntaxgraph

Ein Syntaxgraph wird beschrieben durch das Tupel (V, E, v_{start}, v_{ziel}, V_{term}, V_{nterm}, $Symb_{term}$, $Symb_{nterm}$, Abb_{term}, Abb_{nterm}). Hierbei ist (V, E, v_{start}, { v_{ziel} }) ein pfadzusammenhängender Graph. Die Mengen V_{term} und V_{nterm} sind disjunkte Teilmengen der Knotenmenge V, für die Abbildungen auf die Menge der terminalen Symbole $Symb_{term}$ beziehungsweise der nicht-terminalen Symbole $Symb_{nterm}$ durch

Abb_{term}: $V_{term} \rightarrow Symb_{term}$

Abb_{nterm}: $V_{nterm} \rightarrow Symb_{nterm}$

gegeben sind.

Auf Basis dieser Definition können dynamische Testverfahren bezüglich der mit Syntaxgraphen beschriebenen Funktionalität angegeben werden. Einzelne Syntaxgraphen einer Grammatik können dabei wie in Definition 4-2 als Teilspezifikationen betrachtet werden. Dynamische Testverfahren können sich daher auf einzelne Syntaxgraphen beziehen und für einen Test der in der Gesamtspezifikation beschriebenen Funktionalität kombiniert werden.

Als Eingabebereich sind im dynamischen Test für die Spezifikation von Grammatiken alle Ausdrücke $(Symb_{term})^*$ zu betrachten, die aus dem Eingabealphabet $Symb_{term}$ zu bilden sind. Darüber hinaus sind eventuell auch unzulässige Zeichen einer Menge $A_{ein} \supseteq Symb_{term}$ zu berücksichtigen, so dass als Eingabebereich die Menge $(A_{Ein})^*$ zu betrachten ist. Dies ist insbesondere dann wichtig, wenn der Syntaxgraph nur eine Teilspezifikation darstellt. Zur Beschreibung der Ausgabe wird angenommen, dass die Syntaxgraphen die Identifikation und eventuell auch Klassifikation von gültigen Eingabedaten leisten. Vereinfachend wird hier die Ausgabe durch die endliche Menge { 0, 1, 2, …, k } natürlicher Zahlen codiert, so dass sich insgesamt für Ein- und Ausgabebereich der Spezifikation die Mengen

$I = (A_{Ein})^*$

$O = \{ 0, 1, 2, \ldots, k \}$ $(k \in \mathbb{N} \setminus \{ 0 \})$

ergeben.

Den Eingabedaten sind nun im Sinne der Assoziation aus Definition 4-21 Pfade durch die Syntaxgraphen zuzuordnen. Wie oben bereits ausgeführt, beschreiben die Pfade durch den Syntaxgraphen Produktionsregeln der Grammatik: Die linke Seite der Produktionsregel ergibt sich aus dem dem Syntaxgraphen zugeordnete nicht-terminale Symbol, die rechte Seite durch Konkatenation der Symbole auf dem Pfad.

Definition 4-25: Assoziation der Pfade eines Syntaxgraphen zu einem Eingabedatum

Vorausgesetzt wird die Beschreibung einer Spezifikation $s \in \Sigma_{I, O}$ mit

$I = (A_{Ein})^*$

$O = \{ 0, 1, 2, \ldots, k \}$ $(k \in \mathbb{N})$

durch Angabe einer kontextfreien Grammatik gr = ($Symb_{term}$, $Symb_{nterm}$, Prod, $symb_{start}$), deren Produktionsregeln mit Syntaxgraphen beschrieben sind. Betrachtet wird ein Syntaxgraph (V, E, v_{start}, v_{ziel}, V_{term}, V_{nterm}, $Symb_{term}$, $Symb_{nterm}$, Abb_{term}, Abb_{nterm}) als Teilspezifikation.

Ein Pfad $(v_1, v_2, \ldots, v_k)$ durch den Syntaxgraphen wird genau dann als mit einem Eingabedatum $i \in I$ assoziiert bezeichnet, wenn es eine Ableitung des Eingabedatums i aus dem Startsymbol $symb_{start}$ der Grammatik gibt, die die durch den Pfad beschriebene Produktionsregel verwendet.

Bei der Pfadassoziation zu den Eingabedaten ist zu beachten, dass es möglicherweise mehrere Ableitungen eines Eingabedatums aus dem Startsymbol gibt, und dass nicht alle Ableitungen dieselben Produktionsregeln verwenden müssen. So kann auf Basis der Spezifikation nicht immer sichergestellt werden, dass ein gegebener Pfad eines Syntaxgraphen in dynamischen Test tatsächlich durchlaufen wurde, selbst wenn im Prüfling die Produktionsregeln entsprechend der Spezifikation umgesetzt sind.

Mit Definition 4-25 der Pfadassoziation eines Syntaxgraphen können die Überdeckungsrelationen zwischen den Eingabedaten einerseits und den Knoten, Kanten beziehungsweise Pfaden des Syntaxgraphen andererseits durch Anwendung der Definition 4-22 auf den dem Syntaxgraphen unterliegenden gerichteten Graphen $(V, E, v_{start}, \{v_{ziel}\})$ beschrieben werden. Somit stehen die Relationen $vcov_s$, $ecov_s$ und $pcov_s$ der Knoten-, Kanten- und Pfadüberdeckung durch Eingabedaten zur Verfügung. Auf dieser Basis können die dynamischen Testverfahren für Syntaxgraphen formuliert werden. Als grundlegende Forderung für den dynamischen Test wird in [Bei90] und [Lig02] die Erreichung aller terminalen und nicht-terminalen Symbole des Syntaxgraphen verlangt. Diese kann analog zu Verfahren 4-9 formuliert werden, wobei lediglich die Knoten zu berücksichtigen sind, die den Symbolen der Grammatik entsprechen.

Verfahren 4-14: Überdeckung aller terminalen und nicht-terminalen Symbole eines Syntaxgraphen

(a) Voraussetzung:

Vorausgesetzt wird die Beschreibung einer Spezifikation $s \in \Sigma_{I, O}$ mit

$I = (A_{Ein})^*$

$O = \{ 0, 1, 2, \ldots, k \} \quad (k \in \aleph)$

durch Angabe einer kontextfreien Grammatik $gr = (Symb_{term}, Symb_{nterm}, Prod, symb_{start})$, deren Produktionsregeln mit Syntaxgraphen beschrieben sind. Betrachtet wird ein Syntaxgraph $(V, E, v_{start}, v_{ziel}, V_{term}, V_{nterm}, Symb_{term}, Symb_{nterm}, Abb_{term}, Abb_{nterm})$ als Teilspezifikation.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller terminalen und nicht-terminalen Symbole des Syntaxgraphen wird beschrieben durch

$$X^{StxSymb}: \quad \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(v \in (V_{term} \cup V_{nterm})) \exists(t \in T) ((t, v) \in vcov_s).$$

Die Überdeckung der terminalen und nicht-terminalen Symbole im dynamischen Test eines Syntaxgraphen stellt nicht das Durchlaufen aller Kanten sicher, da Kanten auch zwischen Knoten ohne die Angabe von terminalen oder nicht-terminalen Symbolen gezogen werden können. Eine Überdeckung aller Kanten wird direkt durch Anwendung von Verfahren 4-10 erreicht.

Verfahren 4-15: Überdeckung aller Kanten eines Syntaxgraphen

(a) Voraussetzung:

Vorausgesetzt wird die Beschreibung einer Spezifikation $s \in \Sigma_{I, O}$ mit

$I = (A_{Ein})^*$

$O = \{ 0, 1, 2, \ldots, k \}$ $(k \in \aleph)$

durch Angabe einer kontextfreien Grammatik $gr = (Symb_{term}, Symb_{nterm}, Prod, symb_{start})$, deren Produktionsregeln mit Syntaxgraphen beschrieben sind. Betrachtet wird ein Syntaxgraph $(V, E, v_{start}, v_{ziel}, V_{term}, V_{nterm}, Symb_{term}, Symb_{nterm}, Abb_{term}, Abb_{nterm})$ als Teilspezifikation.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Kantenüberdeckung des Syntaxgraphen wird durch Anwendung von Verfahren 4-10 auf den gerichteten Graphen $(V, E, v_{start}, \{ v_{ziel} \})$ erreicht.

Die Überdeckung aller Kanten eines Syntaxgraphen kann oft mit wenigen dynamischen Testfällen sichergestellt werden. Weitere dynamische Testfälle fordert der Versuch einer Pfadüberdeckung. Eine Pfadüberdeckung ist nur dann erreichbar, wenn der Syntaxgraph keine Zyklen hat. In diesem Fall wird die Pfadüberdeckung wird durch Anwendung von Verfahren 4-11 erreicht.

Verfahren 4-16: Überdeckung aller Pfade eines Syntaxgraphen

(a) Voraussetzung:

Vorausgesetzt wird die Beschreibung einer Spezifikation $s \in \Sigma_{I, O}$ mit

$I = (A_{Ein})^*$

$O = \{ 0, 1, 2, \ldots, k \}$ $(k \in \aleph)$

durch Angabe einer kontextfreien Grammatik $gr = (Symb_{term}, Symb_{nterm}, Prod, symb_{start})$, deren Produktionsregeln mit Syntaxgraphen beschrieben sind. Betrachtet wird ein Syntaxgraph $(V, E, v_{start}, v_{ziel}, V_{term}, V_{nterm}, Symb_{term}, Symb_{nterm}, Abb_{term}, Abb_{nterm})$ als Teilspezifikation.

(b) Verfahrensbeschreibung:

Eine Überdeckung aller Pfade des Syntaxgraphen wird durch Anwendung von Verfahren 4-11 auf den gerichteten Graphen $(V, E, v_{start}, \{ v_{ziel} \})$ erreicht.

Eine auch für Syntaxgraphen mit Zyklen anwendbare Variante fordert in Anlehnung an Verfahren 4-12 den dynamischen Test aller Pfade, die alle Kreise bis zu einer gegebenen maximalen Anzahl von Malen durchlaufen.

Verfahren 4-17: Überdeckung aller Pfade eines Syntaxgraphen, die alle enthaltenen Kreise höchstens n Mal durchlaufen $(n \in \aleph)$

(a) Voraussetzung:

Vorausgesetzt wird die Beschreibung einer Spezifikation $s \in \Sigma_{I, O}$ mit

$I = (A_{Ein})^*$

$O = \{ 0, 1, 2, \ldots, k \}$ $(k \in \aleph)$

durch Angabe einer kontextfreien Grammatik gr = ($Symb_{term}$, $Symb_{nterm}$, Prod, $symb_{start}$), deren Produktionsregeln mit Syntaxgraphen beschrieben sind. Betrachtet wird ein Syntaxgraph (V, E, v_{start}, v_{ziel}, V_{term}, V_{nterm}, $Symb_{term}$, $Symb_{nterm}$, Abb_{term}, Abb_{nterm}) als Teilspezifikation.

(b) Verfahrensbeschreibung:

Eine Überdeckung aller Pfade des Syntaxgraphen, die alle enthaltenen Kreise höchstens n Mal durchlaufen ($n \in \aleph$), wird durch Anwendung von Verfahren 4-12 auf den gerichteten Graphen (V, E, v_{start}, { v_{ziel} }) erreicht.

Weniger umfangreich gestaltet sich die Forderung, dass alle im Syntaxgraphen beschriebenen Kreise im dynamischen Test bis zu n-fach ($n \in \aleph \setminus \{ 0 \}$) ausgeführt werden. Diese Anforderung wird durch Anwendung von Verfahren 4-13 realisiert.

Verfahren 4-18: Überdeckung der bis zu n-fachen Ausführung aller Kreise eines Syntaxgraphen ($n \in \aleph \setminus \{ 0 \}$)

(a) Voraussetzung:

Vorausgesetzt wird die Beschreibung einer Spezifikation $s \in \Sigma_{I, O}$ mit

$I = (A_{Ein})^*$

$O = \{ 0, 1, 2, \ldots, k \}$ $\quad (k \in \aleph)$

durch Angabe einer kontextfreien Grammatik gr = ($Symb_{term}$, $Symb_{nterm}$, Prod, $symb_{start}$), deren Produktionsregeln mit Syntaxgraphen beschrieben sind. Betrachtet wird ein Syntaxgraph (V, E, v_{start}, v_{ziel}, V_{term}, V_{nterm}, $Symb_{term}$, $Symb_{nterm}$, Abb_{term}, Abb_{nterm}) als Teilspezifikation.

(b) Verfahrensbeschreibung:

Eine Überdeckung der bis zu n-fachen Ausführung ($n \in \aleph \setminus \{ 0 \}$) aller Kreise des Syntaxgraphen wird durch Anwendung von Verfahren 4-13 auf den gerichteten Graphen (V, E, v_{start}, { v_{ziel} }) erreicht.

Beizer schlägt in [Bei90] zum dynamischen Test der in einem Syntaxgraphen spezifizierten Funktionalität eine Strategie vor, die sich an der Anzahl der Kreisdurchläufe orientiert. Zunächst schlägt er den Durchlauf aller Pfade des Syntaxgraphen vor, die keinen Kreis durchlaufen. Dies entspricht der Anwendung von Verfahren 4-17 für n = 0. Anschließend schlägt der vor, die Kreise keinmal, einmal und zweimal zu durchlaufen, was durch ergänzende Anwendung von Verfahren 4-18 mit n = 2 erreicht werden kann. Darüber hinaus schlägt er vor, zu jedem Kreis die maximale Anzahl zu ermitteln, mit der er durchlaufen werden kann. Diese Anzahl ist aus dem Syntaxgraphen für kontextfreie Grammatiken nicht ablesbar, sie muss sich aus dem Kontext der Spezifikation ergeben, kann aber auch auf Basis einer Fehlererwartung festgelegt werden. Setzt man in der Spezifikation die Angabe $max(k) \in \aleph$ der maximalen Durchlaufzahl für jeden enthaltenen Kreis k voraus, so kann durch Modifikation von Verfahren 4-13 ein weiteres funktionsorientiertes dynamisches Testverfahren für den Syntaxgraphen beschrieben werden, das Kreisdurchläufe durch jeden Kreis k mit der von Beizer geforderten Anzahl max(k)-1, max(k) und max(k)+1 fordert. Wird in einem Syntaxgraphen zu einem nicht-terminalen Symbol das Symbol selbst als Knoten verwendet, so verlangt Beizer zusätzlich eine Prüfung dieser

Rekursion. Diese kann im dynamischen Test durch eine weitere Modifikation der Darstellung des Syntaxgraphen erreicht werden, indem anstelle der Verwendung des Knotens mit dem nicht-terminalen Symbol ein weiterer Hilfsknoten eingefügt wird, der mit einer weiteren „Instanz“ des Syntaxgraphen durch eine abgehenden Kante zu dessen Startknoten und einer eingehenden Kante von dessen Zielknoten verbunden ist. Diese Kante schließt die Zyklen, die sich durch die direkte Rekursion ergeben. Indirekte Rekursionen, die weitere Syntaxgraphen mit einbeziehen, werden weder in [Bei90] noch in [Lig02] berücksichtigt.

Zusätzlich legt Beizer besonderes Gewicht auf dynamische Testfälle, die nicht durch den Syntaxgraphen akzeptiert werden. Bei der Auswahl entsprechender Testdaten orientiert er sich am Syntaxgraphen. Er wählt Eingabedaten aus, die bis zu dem betrachteten Strukturelement der Syntax entsprechen, jedoch nicht dem weiteren Verlauf der möglichen Pfade. Dies entspricht der Forderung, auf Basis der Spezifikation zu jedem Knoten einen dynamischen Testfall zu verlangen, der bis zu diesem Knoten einem Pfad durch den Syntaxgraph durchläuft, anschließend jedoch zu einem Abweisen des Eingabedatums führt. Für spezielle Formen von Grammatiken wie beispielsweise reguläre Grammatiken kann dieser Ansatz systematisiert werden. Diese Grammatiken können durch einen deterministischen endlichen Automaten erkannt werden [Sch01], in dem in jedem Zustand für ungültige Zeichen ein Übergang in einen zusätzlichen Zustand des Automaten eingetragen ist. Dieser Zustand entspricht einer Fehlererkennung mit Abbruch der weiteren Analyse. Die von Beizer formulierte Prüfung kann dann durch Prüfung aller Übergänge zwischen den Zuständen des Automaten beschrieben werden. Eine Formulierung dieser Strategie für allgemeine Syntaxgraphen ist nicht möglich, da hier der fehlerhafte Zustand nicht immer eindeutig erkannt werden kann.

Liggesmeyer regt in [Lig02] die Verwendung weiterer dynamischer Testfälle an, die nicht aus der kontextfreien Grammatik abgeleitet werden können, sondern die sich aus der weiteren Spezifikationssemantik ergeben. Als Beispiel führt er die Bereichgrenzen eine Gleitkommazahl an, die nicht in einer syntaktischen Beschreibung wie in Abbildung 4-3 berücksichtigt werden können. Ergeben sich kontextabhängige Forderungen aus weiteren Teilen der Spezifikation, so werden diese durch die kombinierte Anwendung verschiedener dynamischer Testverfahren auf einzelne Teilspezifikationen berücksichtigt.

4.3.3.2 Aktivitätsdiagramme

Aktivitätsdiagramme werden zur Spezifikation und Modellierung im Rahmen der Unified Modeling Language (UML) [BooRumJac98, Oes98, OMG03] empfohlen. Auch in anderen Entwicklungswerkzeugen werden entsprechende Modelle verwendet [HarEA90]. In der Praxis finden die Aktivitätsdiagramme breite Anwendung. Mit ihrer Hilfe werden die Aktivitäten eines Systems, ihre Durchführungsreihenfolge und die für ihre Durchführung notwendigen Bedingungen spezifiziert. Sie geben so eine ablauforientierte Beschreibung des geforderten Systemverhaltens. Eine umfassende Erläuterung der Syntax und Semantik von Aktivitätsdiagrammen kann in [Oes98] eingesehen werden, ihre Kenntnis wird im Folgenden vorausgesetzt.

Aktivitätsdiagramme werden in graphischer Form spezifiziert und können mit Hilfe der graphentheoretischen Ansätze des Abschnitts 4.3.2 beschrieben werden. Hierzu wird ein Aktivitätsdiagramm als gerichteter Graph mit Zusatzinformationen aufgefasst. Die Knotenmenge wird durch die Aktivitäten, die Entscheidungen, die Synchronisations- und die Splittingknoten bestimmt. Die Kantenmenge wird durch Transitionen bestimmt, die zwischen Aktivitäten, Entscheidungen, Splitting- und Synchronisationsknoten im Aktivitätsdiagramm gezogen sind. Start- und Zielknoten des Diagramms werden direkt

übernommen. Die Bedingungen, die an Transitionen angetragen werden und die bei mehreren ausgehenden Transitionen aus einer Aktivität oder Entscheidung über die zu durchlaufende Transition und damit weitere Aktivitätenfolge entscheiden, werden als Prädikate über dem Eingabebereich aufgefasst und den Transitionen zugeordnet. Eine mögliche Einteilung in Verantwortungsbereiche, so genannte „swim lanes“, kann in Form einer Zuordnung des Verantwortungsbereichs zu jedem Knoten in der graphentheoretischen Darstellung aufgenommen werden. Es ergibt sich die folgende graphentheoretische Notation eines Aktivitätsdiagramms.

Definition 4-26: Graphentheoretische Notation eines Aktivitätsdiagramms

Ein Aktivitätsdiagramm wird beschrieben durch einen pfadzusammenhängenden Graphen (V, E, v_{start}, V_{ziel}) mit einelementiger Zielknotenmenge $V_{ziel} = \{ v_{ziel} \}$, der den folgenden Bedingungen genügt:

(a) Die Knotenmenge V ist in disjunkte Teilmengen $V_{act} \subseteq V$, $V_{dec} \subseteq V$, $V_{sync} \subseteq V$ und $V_{split} \subseteq V$ eingeteilt, die die Bedingung

$$V = V_{act} \cup V_{dec} \cup V_{sync} \cup V_{split} \cup \{ v_{start} \} \cup \{ v_{ziel} \}$$

erfüllen. V_{act} beschreibt die Menge der Aktivitäten, V_{dec} die Menge der Entscheidungen, V_{split} die Menge der Splittingknoten und V_{sync} die Menge der Synchronisationsknoten.

(b) Die Einteilung der Knoten in k Zuständigkeitsbereiche ($k \in \mathbb{N} \setminus \{ 0 \}$), die „swim lanes“, wird durch die Zuordnung

$$\begin{array}{lccl} sl: & V & \rightarrow & \{ 1, \ldots, k \} \\ & v & \mapsto & sl(v) \end{array}$$

notiert.

(c) Eine Bedingung für den Durchlauf einer Kante kann durch die Zuordnung

$$\begin{array}{lccl} c: & E & \rightarrow & \wp(I) \\ & e & \mapsto & c(e) \end{array}$$

eines einstelligen Prädikats über dem Eingabebereich zur Kante angegeben sein.

(d) Die weiteren Bedingungen für die syntaktisch korrekte Definition eines Aktivitätsdiagramms [OMG03] sind einzuhalten.

Aktivitätsdiagramme werden im Folgenden durch das Tupel (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c) beschrieben. Die Kanten eines Aktivitätsdiagramms werden als Transitionen bezeichnet.

Knoten und Kanten, die im Aktivitätsdiagramm zur Kennzeichnung von Zustandsänderungen von Realisierungsobjekten verwendet werden, sind in der oben beschriebenen Darstellung nicht enthalten. Sie können bei Bedarf zu Informationszwecken ergänzt werden, bilden jedoch keine systematische Grundlage für den dynamischen Test. Grund hierfür ist, dass diese Knoten und Kanten keine Aktivitäten spezifizieren, sondern punktuell auf Aspekte eines zustandsbasierten Verhaltens verweisen. Dies geschieht jedoch nicht systematisch, so dass die Aspekte zwar im dynamischen Test überprüft werden können, aber nicht zur Testdatenselektion oder zur Prüfung der Testdatenadäquatheit verwendet werden sollten. Es ist daher grundsätzlich sinnvoller, Aspekte der Zustandsänderung von Realisierungsobjekten vollständig in einer zustandsbasierten Beschreibungsform zu spezifizieren. Sie können dann mit den Verfahren aus Abschnitt 4.4 geprüft werden und bleiben daher in diesem Abschnitt unberücksichtigt.

Die graphentheoretische Beschreibung der Aktivitätsdiagramme in Definition 4-26 legt eine Anwendung der dynamischen Testverfahren aus Abschnitt 4.3.2 nahe. Um die dafür notwendige Pfadassoziation zu den Eingabedaten zu beschreiben, muss zunächst der Begriff eines Pfades durch ein Aktivitätsdiagramm definiert werden. Dieser ist weiter zu fassen als in Definition 4-20, da in Aktivitätsdiagrammen die Darstellung von Nebenläufigkeiten mit Hilfe von Splitting- und Synchronisationsknoten möglich ist. Eine sequenzielle Beschreibung von Pfaden mit Hilfe von Knotenfolgen wie in Definition 4-20 ist für Aktivitätsdiagramme daher nicht geeignet. Aus diesem Grund wird eine graphentheoretische Struktur zur Definition eines erweiterten Pfades eingeführt.

Definition 4-27: Erweiterte Pfade durch ein Aktivitätsdiagramm, Menge der erweiterten Pfade durch ein Aktivitätsdiagramm, Menge der erweiterten Pfade mit höchstens n-maligem Durchlauf der im Aktivitätsdiagramm enthaltene Kreise ($n \in \aleph$)

Zu einem Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$ wird ein erweiterter Pfad $(p, Vref_p)$ beschrieben durch einen zyklenfreien pfadzusammenhängenden Graphen

$p = (V', E', v'_{start}, v'_{ziel})$

und eine Referenzfunktion

$Vref_p: \quad V' \quad \rightarrow \quad V,$

die die Knoten von p auf Knoten von a abbildet. Graph und Referenzfunktion müssen die Bedingungen

(a) $Vref_p(v'_{start}) = v_{start}$

(b) $Vref_p(v'_{ziel}) = v_{ziel}$

(c) $\forall(e \in E') ((Vref_p(anf(e)), Vref_p(end(e))) \in E)$

(d) $\forall(e_1 \in E') \forall(e_2 \in E') (((e_1 \neq e_2) \wedge (anf(e_1) = anf(e_2))) \Rightarrow (Vref_p(anf(e_1)) \in V_{split}))$

(e) $\forall(e_1 \in E') \forall(e_2 \in E') (((e_1 \neq e_2) \wedge (end(e_1) = end(e_2))) \Rightarrow (Vref_p(end(e_1)) \in V_{sync}))$

(f) $(V', E', v'_{start}, v'_{ziel})$ ist zyklenfrei

erfüllen.

Die Menge der erweiterten Pfade durch das Aktivitätsdiagramm a wird mit ePfade(a) bezeichnet. Die Menge aller erweiterten Pfade zu einem Aktivitätsdiagramm, die die im Aktivitätsdiagramm enthaltenen Knoten höchstens von n + 1 Knoten des erweiterten Pfades referenzieren ($n \in \aleph$), wird mit $ePfade_n(a)$ bezeichnet. Als Menge der erweiterten Pfade zu allen zyklenfreien Pfaden durch das Aktivitätsdiagramm ergibt sich $ePfade_0(a)$.

Zur Beschreibung der auf einem erweiterten Pfad besuchten Knoten des Aktivitätsdiagramms wird in der obigen Definition ein gerichteter Graph verwendet. Die Referenzfunktion ermöglicht die Identifikation der besuchten Knoten. Die Bedingungen (a) und (b) stellen sicher, dass ein erweiterter Pfad immer mit Referenz auf den Startknoten des Aktivitätsdiagramms beginnt und mit Referenz auf den Zielknoten endet. Bedingung (c) gewährleistet, dass die Transitionen des erweiterten Pfades denen des Aktivitätsdiagramms entsprechen. Die Bedingungen (d) und (e) lassen die Verzweigungen im erweiterten Pfad nur im Anschluss an Knoten mit Referenz auf einen Splittingknoten und Synchronisation nur vor Knoten mit Referenz auf einen Synchronisationsknoten zu. Sie sorgen dafür, dass ein erweiterter Pfad in einem dynamischen Testlauf komplett durchlaufen werden kann. Die Verzweigungen im erweiterten Pfad ermöglichen die Darstellung der Nebenläufigkeiten. Die in (f) geforderte Zyklenfrei-

heit des erweiterten Pfades stellt sicher, dass Zyklen beim Durchlauf des Aktivitätsdiagramms zu einer sequentiellen Darstellung im erweiterten Pfad führen. Die Anzahl der Durchläufe ist damit im erweiterten Pfad eindeutig festgelegt; Knoten, die durch einen Zyklus mehrfach besucht werden, werden mehrfach referenziert.

Mit Hilfe der obigen Definition der erweiterten Pfade eines Aktivitätsdiagramms kann die Überdeckung der Strukturelemente des Diagramms beschrieben werden. Hierzu ist zunächst die Assoziation der Eingabedaten zu den erweiterten Pfaden in Analogie zu Definition 4-21 zu beschreiben.

Definition 4-28: Assoziation der erweiterten Pfade zu den Eingabedaten

Vorausgesetzt wird die Beschreibung der Spezifikation $s \in \Sigma_{I,\ O}$ durch ein Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$.

Ein erweiterter Pfad $(p, Vref_p) \in ePfade(a)$ durch das Aktivitätsdiagramm wird genau dann als mit einem Eingabedatum $i \in I$ assoziiert bezeichnet, wenn laut Spezifikation für das Eingabedatum die durch den erweiterten Pfad beschriebene Funktionalität anzuwenden ist. Dies wird durch die Zuordnung

$$\begin{array}{lccc} ePfadAssoz_s: & I & \rightarrow & \wp(ePfade(a)) \\ & i & \mapsto & ePfadAssoz_s(i) \end{array}$$

ausgedrückt, die jedem Eingabedatum die Menge der laut Spezifikation mit ihm assoziierten erweiterten Pfade zuweist. Die Zuordnung wird als Assoziation erweiterter Pfade zu einem Eingabedatum bezeichnet.

Kann eine korrekte und vollständige Beschreibung c der Bedingungen für den Durchlauf der Transitionen im erweiterten Pfad $(p, Vref_p)$ mit $p = (V', E', v'_{start}, v'_{ziel})$ vorausgesetzt werden, so wird p zu genau den Eingabedaten assoziiert, die alle Prädikate an den Kanten von p erfüllen. In diesem Fall kann die Assoziation durch

$$((p, Vref_p) \in ePfadAssoz_s(i)) :\Leftrightarrow (\forall(e \in E')\ (i \in c(e)))$$

definiert werden.

Mit dieser Definition der Assoziation erweiterter Pfade zu den Eingabedaten kann die Überdeckung der Strukturelemente des Aktivitätsdiagramms für die dynamischen Testfälle analog zu Definition 4-22 beschrieben werden.

Definition 4-29: Überdeckung der Strukturelemente eines Aktivitätsdiagramms durch einen dynamischen Testfall

Vorausgesetzt wird die Beschreibung der Spezifikation $s \in \Sigma_{I,\ O}$ durch ein Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$.

(a) Die Überdeckung $pcov_s \subseteq I \times ePfade(a)$ von erweiterten Pfaden des Aktivitätsdiagramms a durch Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(p \in ePfade(a))$$
$$(((i, p) \in pcov_s) :\Leftrightarrow (p \in ePfadAssoz_s(i)))$$

über der Menge der Eingabedaten und der erweiterten Pfade durch das Aktivitätsdiagramm.

(b) Die Überdeckung $ecov_s \subseteq I \times E$ von Transitionen des Aktivitätsdiagramms a durch Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(e \in E)$$
$$(((i, e) \in ecov_s) :\Leftrightarrow$$
$$(\exists((p, Vref_p) \in ePfadAssoz_s(i))\ \exists(e' \in Kanten(p))$$
$$(e = (Vref(anf(e')), Vref(end(e'))))))$$

über der Menge der Eingabedaten und der Kanten des Aktivitätsdiagramms.

(c) Die Überdeckung $vcov_s \subseteq I \times V$ von Knoten des Aktivitätsdiagramms a durch Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(v \in V)$$
$$(((i, v) \in vcov_s) :\Leftrightarrow (\exists((p, Vref_p) \in ePfadAssoz_s(i))\ \exists(v' \in Knoten(p))\ (v = Vref_p(v'))))$$

über der Menge der Eingabedaten und der Knoten des Aktivitätsdiagramms.

Mit diesen Definitionen besteht die Möglichkeit, im dynamischen Test die Überdeckung aller Knoten, Kanten oder Pfade durch das Aktivitätsdiagramm zu fordern. So beschreibt das folgenden Verfahren die Überdeckung aller Knoten eines Aktivitätsdiagramms.

Verfahren 4-19: Überdeckung aller Knoten eines Aktivitätsdiagramms

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I,\,O}$ durch ein Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller Knoten des Aktivitätsdiagramms a wird beschrieben durch

$$X^{A\text{-}Vcov}: \quad \Sigma_{I,\,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(v \in V)\ \exists(t \in T)\ ((t, v) \in vcov_s).$$

Entsprechend kann auch eine selektive Überdeckung aller Aktivitäten, Entscheidungen, Synchronisations- oder Splittingknoten gefordert werden, in dem das Prädikat $P_s(T)$ auf die Mengen V_{act}, V_{dec}, V_{sync} oder V_{split} eingeschränkt wird. Darüber hinaus kann wie im Folgenden ein dynamisches Testverfahren beschrieben werden, dass die Überdeckung aller Transitionen eines Aktivitätsdiagramms im dynamischen Test fordert.

Verfahren 4-20: Überdeckung aller Transitionen eines Aktivitätsdiagramms

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I,\,O}$ durch ein Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller Transitionen des Aktivitätsdiagramms a wird beschrieben durch

$$X^{A\text{-}Ecov}: \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(e \in E)\, \exists(t \in T)\, ((t, e) \in ecov_s).$$

Weiterhin ist es möglich, wie im folgenden Verfahren den dynamischen Test aller erweiterten Pfade durch das Aktivitätsdiagramm zu fordern.

Verfahren 4-21: Überdeckung aller erweiterten Pfade durch ein Aktivitätsdiagramm

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I,O}$ durch ein Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller erweiterten Pfade durch das Aktivitätsdiagramm a wird beschrieben durch

$$X^{A\text{-}Pcov}: \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(p \in ePfade(a))\, \exists(t \in T)\, ((t, p) \in pcov_s).$$

Falls Zyklen im Aktivitätsdiagramm auftreten, können wie in Verfahren 4-12 und Verfahren 4-13 eingeschränkte Forderungen an die Überdeckung der Zyklen gestellt werden. Das folgende Verfahren verlangt den dynamischen Test aller Pfade durch das Aktivitätsdiagramm, die enthaltene Kreise bis zu n Mal durchlaufen ($n \in \aleph$).

Verfahren 4-22: Überdeckung aller erweiterten Pfade eines Aktivitätsdiagramms, die die enthaltenen Kreise höchstens n Mal durchlaufen ($n \in \aleph$)

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I,O}$ durch ein Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller erweiterten Pfade des Aktivitätsdiagramms, die alle enthaltenen Kreise höchstens n Mal durchlaufen, wird beschrieben durch

$$X^{A\text{-}P(n)Cov}: \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(p \in ePfade_n(a))\, \exists(t \in T)\, ((t, p) \in pcov_s).$$

Mit weniger Aufwand verbunden ist die Forderung der bis zu n-fachen Durchläufe aller im Aktivitätsdiagramm enthaltenen Kreise ($n \in \aleph \setminus \{ 0 \}$). Zur einfachen Notation wird eine Funktion eAnz verwendet, die einem Aktivitätsdiagramm a, einem zugehörigen erweiterten Pfad p und einem Weg $w = (v_1, v_2, \ldots, v_k)$ durch das Aktivitätsdiagramm die Anzahl eAnz(a, p, w) der Vorkommen der Knotenfolge w im erweiterten Pfad p zuordnet.

Verfahren 4-23: Überdeckung der bis zu n-fachen Ausführung aller Kreise eines Aktivitätsdiagramms ($n \in \aleph \setminus \{ 0 \}$)

(a) Voraussetzungen:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch ein Aktivitätsdiagramm $a = (V, E, v_{start}, v_{ziel}, V_{act}, V_{dec}, V_{sync}, V_{split}, sl, c)$.

Kreise(a) = $\{ k_1, \ldots, k_l \}$ ($l \in \aleph$) bezeichne die Menge aller Kreise im dem Aktivitätsdiagramm unterliegenden gerichteten Graphen.

Die Funktion eAnz gebe die Anzahl eAnz(a, p, w) der Vorkommen der einer Knotenfolge w in einem erweiterten Pfad p zu einem Aktivitätsdiagramm a an.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung der bis zu n-fachen Ausführung aller Kreise eines Aktivitätsdiagramms wird beschrieben durch

$$X^{A\text{-}K(n)Cov}: \quad \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(k \in K)\, \forall(m \in \{ 0, \ldots, n \})$$
$$\exists(t \in T)\, \exists(p \in ePfadAssoz_s(t))\, (eAnz(a, p, k) = m).$$

Die Notation der dynamischen Testverfahren dieses Abschnitts zeigt, dass der dynamische Test auf Basis von Aktivitätsdiagrammen weitgehend formal beschrieben werden kann. Dies ist begründet durch die den Aktivitätsdiagrammen zugrunde liegende graphentheoretische Struktur, die als Basis der Testdatenselektion oder als Basis der Prüfung der Testdatenadäquatheit verwendet werden kann. Auf Basis dieser Struktur ist auch eine Standardisierung und Automatisierung der Durchführung des dynamischen Tests von Aktivitätsdiagrammen denkbar. Dies ist insbesondere dann der Fall, wenn die Bedingungen an den Transitionen des Aktivitätsdiagramms vollständig in Form von Prädikaten über dem Eingabebereich angegeben sind.

Bei der Verwendung von Aktivitätsdiagrammen in einer Spezifikation ist zu beachten, dass durch die Diagramme nicht unbedingt eine vollständige Beschreibung der Spezifikation im Sinne von Definition 3-2 erreicht wird. So beschreiben die Aktivitätsdiagramme lediglich den Ablauf und Reihenfolge von Aktivitäten, die für die Eingabedaten gefordert sind. Die Aktivitäten selbst werden oft nur durch die Aktivitätsnamen beschrieben, die oft nur intuitiv gedeutet werden können. In diesem Fall sind die Kreativität und Intuition des Entwicklers oder Testers gefragt, um ein umfassendes Verständnis der Spezifikation aus dem Diagramm abzuleiten. Werden die im Aktivitätsdiagramm aufgeführten Aktivitäten außerhalb des Aktivitätsdiagramms umfassend beschrieben, kann eine vollständige Beschreibung der Spezifikation gegeben sein. Für eine solche Beschreibung der Aktivitäten eignet sich beispielsweise eine funktionale oder relationale Darstellung über Teilmengen des Ein- und Ausgabebereichs. Ist eine funktionale Darstellung gegeben, so kann das Aktivitätsdiagramm als Beschreibung einer Verkettung

partieller Funktionen aufgefasst werden. Der Definitionsbereich der partiellen Funktionen über dem Eingabebereich wird durch die Transitionsbedingungen eingeschränkt, die auch im Hinblick auf Ergebnisse von bereits abgeschlossenen Aktivitäten formuliert sein können. Eine solche Verkettung partieller Funktionen kann ein geeignetes Mittel zur vollständigen Beschreibung der Spezifikation sein. Das Aktivitätsdiagramm stellt in diesen Fällen ein mächtiges und gleichzeitig intuitiv eingängiges Werkzeug zur Spezifikation dar; die oben formulierten dynamischen Testverfahren beschreiben eine systematische Analyse der spezifizierten relationalen oder funktionalen Verkettungen. Sind einzelne Aktivitäten in Form von Teilspezifikationen beschrieben, die eine Anwendung weiterer funktionsorientierter Testverfahren erlauben, ist eine umfassende Prüfung durch Verfahrenskombination möglich.

4.3.3.3 Sequenz- und Kollaborationsdiagramme

Sequenzdiagramme und Kollaborationsdiagramme stellen eine weitere graphische Spezifikationsform dar, die im Rahmen der Unified Modeling Language (UML) [BooRumJac98, Oes98, OMG03] angeboten wird. In der Praxis sind insbesondere die Sequenzdiagramme gut akzeptiert und weit verbreitet. Beide Diagrammtypen beschreiben einen Kommunikations- oder Transaktionsfluss zwischen Systemkomponenten und ermöglichen so eine komponentenbasierte und verhaltensorientierte Systemspezifikation. Sequenzdiagramme stellen dabei insbesondere den zeitlichen Verlauf des Kommunikations- oder Transaktionsflusses zwischen Systemkomponenten dar, während Kollaborationsdiagramme einen Schwerpunkt auf die Visualisierung der Kommunikationsstruktur und Komponentenzusammenhänge legen. Eine umfassende Erläuterung der Syntax und Semantik der Sequenz- und Kollaborationsdiagramme kann in [Oes98] eingesehen werden, so dass hier darauf verzichtet wird.

Der Gegenstand der Darstellung ist in beiden Spezifikationsformen identisch: Visualisiert wird ein Kommunikations- oder Transaktionsfluss im System und die daran beteiligten Systemkomponenten. Dabei wird in beiden Spezifikationsformen die Zeit berücksichtigt, die im Sequenzdiagramm als Kontinuum an einer (gedachten) vertikalen Achse fortschreitet und die sich im Kollaborationsdiagramm in aufsteigenden Transaktionsnummern niederschlägt. Diese unterschiedlichen Darstellungen des zeitlichen Verlaufs entstehen durch die Einteilung der Zeichenfläche in den Diagrammen: So nutzt das Kollaborationsdiagramm beide Dimensionen der Zeichenfläche zur Platzierung der Systemkomponenten, so dass für die Darstellung des zeitlichen Aspektes lediglich die Nummerierung der Transaktions- oder Kommunikationsflüsse bleibt. Die partiell geordneten Transaktionsnummern können als Notation der temporalen und kausalen Zusammenhänge zwischen den Nachrichten oder als diskrete Darstellung der Zeit aufgefasst werden. Das Sequenzdiagramm hingegen platziert die Systemkomponenten lediglich in einer Dimension, der horizontalen, so dass die zweite, vertikale Dimension der Zeichenfläche zur Darstellung des zeitlichen Aspektes genutzt werden kann. Temporale und kausale Zusammenhänge werden dabei zwischen der Akzeptanz einer Nachricht oder Transaktion und der Versendung weiterer Nachrichten angenommen. So können im Sequenzdiagramm Aspekte des zeitlichen Verlaufs mit graphischen Mitteln dargestellt werden, die im Kollaborationsdiagramm durch textuelle Beschreibungselemente angegeben werden müssen. Dies schlägt sich neben der Beschreibung des zeitlichen Ablaufs des Kommunikations- oder Transaktionsflusses auch auf die Darstellung der Erzeugung oder Vernichtung beteiligter Systemkomponenten nieder. Trotz der Unterschiede in der Darstellungsform wird aus diesen Überlegungen deutlich, dass die Inhalte eines Sequenzdiagramms in einem äquivalenten Kollaborationsdiagramm dargestellt werden können und umgekehrt. Diese Äquivalenz der Darstellungsformen ist inhaltlicher Natur, eine Bewertung der Diagrammtypen im Hinblick auf ihre kognitive Erfassbarkeit soll hier nicht gegeben werden.

Die inhaltliche Äquivalenz von Sequenz- und Kollaborationsdiagrammen ermöglicht eine gleichartige formale Beschreibung ihres Inhalts mit graphentheoretischen Mitteln. Diese orientiert sich an der Form des Kollaborationsdiagramms, das einfach mit den Ansätzen des Abschnitts 4.3.2 beschrieben werden kann: Hierzu werden die Systemkomponenten als Knoten eines Graphen aufgefasst werden, die Transaktions- oder Kommunikationsflüsse als Kanten. Als Zusatzinformation werden an den Knoten der Zeitpunkt ihrer Erzeugung und Vernichtung vermerkt und an den Kanten die Zeitpunkte oder die Ordnungsnummern des beschriebenen Transaktions- oder Kommunikationsflusses angetragen. Hierzu können alternativ – je nach dem, ob der Inhalt eines Sequenz- oder eines Kollaborationsdiagramms wiedergegeben wird – Zeitangaben über dem Kontinuum der Systemzeit oder partiell geordnete Transaktionsnummern angegeben werden. Für nicht-transiente Systemkomponenten kann die Nennung der Zeitpunkte für Erzeugung und Vernichtung entfallen. Somit ergibt sich die folgende graphentheoretische Notation des in einem Sequenz oder Kollaborationsdiagramm dargestellten Transaktions- oder Kommunikationsflusses.

Definition 4-30: Graphentheoretische Notation eines Sequenz- oder Kollaborationsdiagramms

Das Paar $(S, \leq)$ bezeichnet im Folgenden eine partielle Ordnung, das heißt $\leq$ bezeichne eine reflexive, identitive und transitive zweistellige Relation über S [ReiSoe01].

Ein Sequenz- oder Kollaborationsdiagramm wird beschrieben durch einen endlichen gerichteten Graphen (V, E), eine partielle Ordnung $(S, \leq)$ und die Abbildungen

$$\begin{array}{lccl} lz: & V & \rightarrow & S \times S \\ & v & \mapsto & (lz_1(v), lz_2(v)), \\ tz: & E & \rightarrow & S \\ & e & \mapsto & tz(e) \end{array}$$

und

$$\begin{array}{lccl} c: & E & \rightarrow & \wp(I) \\ & e & \mapsto & c(e), \end{array}$$

die den Bedingungen

$$\forall(v \in V)\ (lz_1(v) \leq lz_2(v))$$

und

$$\forall(e \in E)$$
$$((lz_1(anf(e)) \leq tz(e)) \wedge (tz(e) \leq lz_2(anf(e))) \wedge (lz_1(end(e)) \leq tz(e)) \wedge (tz(e) \leq lz_2(end(e)))$$

genügen.

Die Knoten des gerichteten Graphen repräsentieren die Systemkomponenten, die Abbildung lz gibt ihre Erzeugungszeit lz_1 und Vernichtungszeit lz_2 an. Die Kanten stellen Transaktions- beziehungsweise Informationsflüsse dar, die Abbildung tz gibt den Transaktionszeitpunkt an, die Abbildung c die Transaktionsbedingung. Sequenz- oder Kollaborationsdiagramme werden im Folgenden durch das Tupel $(V, E, S, \leq, lz, tz, c)$ beschrieben.

Diese Darstellung gibt die wesentlichen Inhalte eines Sequenz- und Kollaborationsdiagramms wieder und kann bei Bedarf um weitere Informationen ergänzt werden. Als Ergänzungen können Übergabe- und Antwortparameter, Iterationsangaben für das Versenden von Nachrichten, Sichtbarkeitsangaben der Systemkomponenten oder Synchronisationsmerkmale der Kommunikation eingefügt werden. Die-

se Angaben enthalten Informationen zur Realisierung und sind für die Auswahl der dynamischen Testfälle meist weniger relevant, sie werden daher in der obigen Darstellung nicht berücksichtigt.

Grundsätzlich illustriert ein Sequenz- oder Kollaborationsdiagramm eine geforderte Funktionalität des Systems und liefert eine exemplarische Darstellung ihrer Nutzung. Daher wird im Folgenden angenommen, dass ein Sequenz- oder Kollaborationsdiagramm durch einen einzelnen dynamischen Testfall geprüft werden kann, der alle Kanten der graphentheoretischen Darstellung durchläuft. Ist diese Annahme nicht gerechtfertigt, kann das spezifizierte Sequenz- oder Kollaborationsdiagramm entsprechend zerlegt werden, die Teile werden dann als Teilspezifikationen betrachtet. Aus diesem Grund müssen für Sequenz- und Kollaborationsdiagramme keine Pfade identifiziert werden, da das Diagramm einem einzigen Pfad entspricht. Somit ergibt sich die folgende Definition der Pfadassoziation zu einem Sequenz- oder Kollaborationsdiagramm.

Definition 4-31: Assoziation eines Sequenz- oder Kollaborationsdiagramms zu einem Eingabedatum

Vorausgesetzt wird die Beschreibung einer Teilspezifikation der Spezifikation $s \in \Sigma_{I, O}$ durch ein Sequenz- oder Kollaborationsdiagramm $d = (V, E, S, \leq, lz, tz, c)$.

Das Sequenz- oder Kollaborationsdiagramm d wird genau dann als mit einem Eingabedatum $i \in I$ assoziiert bezeichnet, wenn laut Spezifikation für das Eingabedatum die durch das Sequenz- oder Kollaborationsdiagramm d beschriebene Funktionalität anzuwenden ist. Dies wird mit dem einstelligen und vom Diagramm d abhängigen Prädikat $DiagAssoz_d(i)$ bezeichnet.

Kann eine korrekte und vollständige Beschreibung c der Transaktionsbedingungen im Diagramm vorausgesetzt werden, so werden genau die Eingabedaten als zum Diagramm assoziiert bezeichnet, die alle Prädikate der Kanten von d erfüllen. In diesem Fall gilt

$$DiagAssoz_d(i) \Leftrightarrow (\forall (e \in E)\ (i \in c(e))).$$

Mit dieser Assoziation kann der dynamische Test der in einem Sequenz- oder Kollaborationsdiagramm dargestellten Funktionalität einfach beschrieben werden.

Verfahren 4-24: Überdeckung eines Sequenz- oder Kollaborationsdiagramms

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch ein Sequenz- oder Kollaborationsdiagramm $d = (V, E, S, \leq, lz, tz, c)$.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung eines Sequenz- oder Kollaborationsdiagramms wird beschrieben durch

$$X^{S/K\text{-}Cov}: \quad \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I)) \qquad s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \exists (t \in T)\ (DiagAssoz_d(t)).$$

Das Verfahren fordert zur Prüfung der in einem Sequenz- oder Kollaborationsdiagramm dargestellten Funktionalität lediglich einen dynamischen Testfall, der mit dem Diagramm assoziiert ist. Da Sequenz-

und Kollaborationsdiagramme häufig besonders wichtige Situationen beschreiben, ist diese Forderung trotz ihrer Einfachheit sinnvoll. So kann die Anwendung von Verfahren 4-24 in einer kombinierten Prüfung verschiedener Teilspezifikationen die Prüfung der in den Diagrammen dargestellten Funktionalität sicherstellen. Dies entspricht der Forderung in [Lig02], die den dynamischen Test der in Sequenzdiagrammen dargestellten Funktionalität als notwendig, aber nicht hinreichend bezeichnet.

Die Darstellung von Kollaborations- und Sequenzdiagrammen in der Spezifikation kann zwei unterschiedlichen Sichtweisen entsprechen. Einerseits kann das zu spezifizierende System als Komponente im Diagramm erscheinen, so dass das Diagramm die Systemnutzung in einem möglichen Umfeld beschreibt. In diesem Fall liefert das Diagramm eine Grundlage zur Ableitung eines Systemtestfalls. Andererseits ist es möglich, dass das Diagramm die geforderte interne Realisierung einer funktionalen Anforderung vor dem Hintergrund einer möglichen Systemarchitektur beschreibt. Die Komponenten stellen dann Teilsysteme des Systems dar. In diesem Fall beschreibt das Diagramm einen dynamischen Testfall zur Komponentenintegration. In beiden Fällen muss berücksichtigt werden, dass neben der beschriebenen Interaktion des Systems, seiner Umwelt beziehungsweise seiner Komponenten auch eine relationale Beschreibung der Spezifikation gemäß Definition 3-2 anzugeben ist. Diese kann beispielsweise durch einen weitere Spezifikation der dargestellten Reaktionen der Einzelkomponenten erreicht werden: Werden hier Zuordnungen beschrieben, die den eingehenden Informationen die zugehörigen Ausgaben zuordnen, so kann hierdurch eine partielle relationale Beschreibung über der durch die Transaktionsbedingungen beschriebenen Eingabepartition erreicht werden. Diese kann durch Zusammenfassung mehrerer Teilspezifikationen die in Definition 3-2 geforderte relationale Beschreibung darstellen. Zur Beschreibung der Zuordnungen in den einzelnen Komponenten können weitere Spezifikationstechniken verwendet werden, die mit geeigneten dynamischen Testtechniken zu prüfen sind, und die mit Verfahren 4-24 kombiniert werden können.

Die weitgehend formalisierte Beschreibung von funktionalen Anforderungen in Kollaborations- und Sequenzdiagrammen, die wie oben beschrieben in einer graphentheoretischen Beschreibung wiedergegeben werden kann, liefert eine formale Grundlage für den dynamischen Test. Eine automatisierte Adäquatheitsbewertung des dynamischen Tests kann erreicht werden, falls die Transaktionsbedingungen vollständig und in automatisiert auswertbarer Form angegeben sind.

4.3.3.4 Zustandsdiagramme

Zustandsdiagramme sind eine häufig verwendete graphische Spezifikationsform, die auch im Rahmen der Unified Modeling Language empfohlen wird [Har87, HarEA90, BooRumJac98, Oes98, OMG03]. Mit Zustandsdiagrammen wird eine zustandsorientierte Sicht des Systemverhaltens während der Systemnutzung beschrieben. Eine eingängige Beschreibung der Syntax und Semantik von Zustandsdiagrammen findet sich in [Oes98]. Sie ist Grundlage der folgenden Überlegung.

Die Syntax eines Zustandsdiagramms kann einfach mit graphentheoretischen Mitteln beschrieben werden. Dennoch sind die in Abschnitt 4.3.2 eingeführten dynamischen Testverfahren für Zustandsdiagramme nicht geeignet: Zustandsdiagramme beschreiben gedächtnisbehaftete Systeme, die über einen Speicher verfügen, der Informationen zwischen einzelnen dynamischen Ausführungen festhält. Dieser Speicher, wie auch immer er realisiert sein mag, gibt unterschiedliche Zustände vor, in denen sich der Prüfling bei Ausführung eines dynamischen Testfalls befinden kann. Zur Beschreibung des dynamischen Tests eines solchen Systems ist das in Kapitel 3.2 eingeführte Modell nicht ausreichend, da hier eine Berücksichtigung des Speicherzustands des Prüflings nicht vorgesehen ist. Eine Erweiterung der Theorie und damit die Bereitstellung einer Beschreibungsform für den dynamischen Test

zustandsbasierter Systeme wird in Abschnitt 4.4 gegeben. Dort werden dynamische Testverfahren eingeführt, die für zustandsbasierte Systeme geeignet sind und die ähnlichen Prinzipien wie denen der graphischen Strukturüberdeckung wie die Verfahren aus Abschnitt 4.3.2 folgen.

4.3.3.5 Petri-Netze

Petri-Netze [Pet62, Pet81] werden als weitere Darstellungsform zur graphischen Spezifikation des Verhaltens von Software und Systemen verwendet. Sie eignen sich insbesondere zur Darstellung asynchroner oder paralleler Prozesse sowie zur Analyse des Zeitverhaltens komplexer Systeme. Insbesondere wegen der definierten Semantik, deren Typ entsprechend dem zu spezifizierenden Verhalten gewählt werden kann, eignen sich Petri-Netze neben der graphischen Modellierung auch für eine analytische oder simulative Auswertung. Aus diesem Grund haben sie sich im Bereich der Spezifikation und Analyse von Betriebssystemen, Kommunikationssystemen und technischen Systemen etabliert. Petri-Netze werden zur Modellierung und Analyse von zeit- oder sicherheitskritischen Systemen verwendet. Sie ermöglichen den Nachweis sicherheitsrelevanter Bedingungen, gestatten eine Analyse des dynamischen Systemverhaltens im Hinblick auf Systemverklemmungen, Lebendigkeits- oder Erreichbarkeitseigenschaften, und können bei geeigneter Modellierung des Ausfallverhaltens auch Zuverlässigkeitsaussagen liefern [LevSto87, Bis90].

Die verschiedenen Ausprägungen von Petri-Netzen unterscheiden sich in Syntax, Semantik und Zeitverhalten. Ihre syntaktische Beschreibung beruht auf einem bipartiten, gerichteten Graphen, dessen Knoten als Stellen beziehungsweise als Transitionen bezeichnet werden. Die Stellen können Marken beinhalten, die entweder als gleichartig vorausgesetzt werden oder mit Hilfe beliebiger Prädikate beschrieben sind. Die Kapazität der Markenaufnahme einer Stelle kann unbeschränkt oder beschränkt sein. Transitionen bestimmen das dynamische Netzverhalten durch Erzeugung und Verbrauch von Marken in den verbundenen Stellen. Dieser Vorgang des Verbrauchens und Erzeugens von Marken wird als Schalten der Transition bezeichnet. Die Regeln zur Erzeugung und zum Verbrauch von Marken in Stellen beim Schalten einer Transition werden durch die verbindenden Kanten beschrieben: So sorgen die in eine Transition eingehenden, aus einer vorgelagerten Stelle ausgehenden Kanten für die Vernichtung von Marken in der vorgelagerten Stelle beim Schalten der Transition. Die aus der Transition ausgehenden, in eine nachgelagerte Stelle eingehenden Kanten bewirken hingegen die Erzeugung von Marken in der nachgelagerten Stelle. Erzeugung und Verbrauch können durch Kantenkapazitäten quantifiziert werden. Zusätzlich werden in manchen Netzen auch weitere Typen von Kanten als Eingangskanten in Transitionen verwendet: Abräumkanten verbrauchen beispielsweise beim Schalten der nachgelagerten Transition alle in der vorgelagerten Stelle vorhandenen Marken. Verbots- oder Aktivierungskanten haben eine prüfende Funktion. Sie verhindern beziehungsweise erlauben das Schalten der Transition, falls die vorgelagerte Stelle eine Markenbelegung größer oder gleich der Kantenkapazität aufweist, verändern die Markenbelegung aber nicht. Werden unterscheidbare Marken verwendet, so können die Kanten mit Variablennamen versehen werden, die in den Transitionen eine Referenzierung der Marken der vorgelagerten Stellen erlauben. Diese Referenzierung ermöglicht die Formulierung von markierungsabhängigen Schaltbedingungen sowie eine erweiterte Beschreibung der Markenerzeugung. Zur Modellierung eines zeitlichen Verhaltens im Netz existieren unterschiedliche Ansätze, deren Auswahl im Hinblick auf die konkurrierenden Ziele einer umfangreichen Modellierungssemantik oder einer aussagekräftigen Auswertungsmöglichkeit getroffen werden muss. So können Zeitangaben mit Stellen, Transitionen oder Kanten verbunden sein, Zeitpunkte oder Zeitdauern beschreiben, minimale und maximale Verweildauern angeben und deterministisch oder stochastisch

verteilt sein. Methoden der Stochastik können zur Lösung von Konflikten durch die gleichzeitige Aktivierung mehrerer Transitionen verwendet werden. So beschreiben beispielsweise zeitbehaftete, stochastische Varianten der Petri-Netze das Systemverhalten durch stochastische Prozesse [AjmEA95].

Trotz der umfangreichen Modellierungssemantik und der vielfältigen Ausprägungen in Praxis und Forschung beschreiben alle Typen von Petri-Netzen letztlich Zustandsübergangsmodelle. So ergeben sich die Zustände aus den möglichen Verteilungen der Marken in den Stellen, die Zustandsübergänge werden durch die Transferwege und Transferbedingungen der Marken bestimmt. Das Zustandsübergangsmodell zu einem Petri-Netz wird als Erreichbarkeitsgraph bezeichnet.

Betrachtet man die Vielzahl der Zustände, in denen sich ein Petri-Netz mit wenigen Stellen und Transitionen befinden kann, so wird die Mächtigkeit dieser Beschreibungsform deutlich. Die intuitive Erfassung dieses Zustandsraums durch den Menschen wird beispielsweise durch die manuelle, simulative Analyse des dynamischen Netzverhaltens im so genannten „Markenspiel“ unterstützt.

Ein vollständiger Überblick über die Syntax, Semantik und Einsatzmöglichkeiten von Petri-Netzen sowie eine Einführung ihrer graphischen Notation kann an dieser Stelle nicht gegeben werden. Stattdessen sei für eine generelle, modellierungsorientierte Einführung auf [Bau96] und für eine Einführung von Petri-Netzen mit stochastisch beschriebenem Zeitverhalten auf [AjmEA95] verwiesen. Mathematische Analysen der mit den verschiedenen Formalismen beschriebenen stochastischen Prozesse werden beispielsweise in [BerDia91, CiaGerLin94] ausgeführt.

Die intuitive Erfassbarkeit von Petri-Netzen bei gleichzeitig mathematisch auswertbarer Semantik hat zu einer intensiven Erforschung ihrer Eigenschaften und zu Anwendungen in der Praxis geführt. Dennoch steht eine Untersuchung der Eignung von Petri-Netzen als Grundlage des dynamischen Tests noch aus. So wird der dynamische Test einer mit Hilfe von Petri-Netzen spezifizierten Funktionalität in Standardwerken im Allgemeinen nicht betrachtet. Auch in der Literatur existieren wenige Beiträge zu diesem Thema. Ausnahmen stellen die Artikel [DesEA97] und [Ram00] dar: In [DesEA97] ist das Petri-Netz selbst der Prüfling, der gegen eine vorliegende Spezifikation mit einem an der Ursache-Wirkungs-Analyse orientierten Verfahren geprüft wird. Die Ursachen, Wirkungen und ihre Zusammenhänge werden aus der Spezifikation abgeleitet und in Form von weiteren Petri-Netzen dargestellt. Ursachen und Wirkungen sind hier durch Stellen repräsentiert, die Ursache-Wirkungs-Zusammenhänge werden durch verbindende Transitionen und Kanten modelliert. Eine Darstellung der Negation ist nicht vorgesehen, kann aber einfach mit Hilfe von Verbotskanten ergänzt werden. Zur Ableitung der Ursachenkombinationsforderungen wird ein simulativer Ansatz gewählt, der keine Beschränkung der Anzahl der dynamischen Testfälle wie in Algorithmus 4-1 oder Algorithmus 4-2 gestattet. Die Ursachenkombinationsforderungen werden durch dynamische Testfälle geprüft, die durch Startmarkierungen des Prüflings beschrieben werden. Anschließend wird geprüft, ob der Prüfling von der Startmarkierung die durch die Ursache-Wirkungs-Zusammenhänge beschriebene Zielmarkierung erreicht. Der Artikel [Ram00] verwendet erweiterte hierarchische Petri-Netze für die Spezifikation verteilter, kommunizierender Prozesse, die sich in einem gegenseitigen Informationsaustausch über ihren aktuellen Zustand informieren und als Agenten bezeichnet werden. Mit Hilfe eines Algorithmus wird eine Menge von Transitionen bestimmt, die als kritisch für die Kommunikation der Agenten angesehen wird und für die eine Überdeckung im funktionsorientierten Test gefordert wird.

Die formale Beschreibung von Petri-Netzen legt eine über diese Vorschläge hinausgehende Betrachtung im funktionsorientierten Test nahe. So ermöglicht die graphentheoretische Darstellung die Definition von Überdeckungskriterien für den dynamischen Test, die sich an den Strukturelementen des Netzes orientieren. Es kann beispielsweise geprüft werden, ob durch die gewählten dynamischen

Testfälle jede Stelle mit einer Marke belegt wird, oder ob jede Transition schaltet. Die Anwendbarkeit der Verfahren, die sich aus solchen Kriterien ergeben, hängt von der Gestalt des Petri-Netzes ab und kann in vielen Fällen durch Analyse des Erreichbarkeitsgraphen geprüft werden. Auch diese zustandsbasierte Darstellung kann zur Auswahl und Bewertung von dynamischen Testläufen verwendet werden: Hier können bei geeigneter Semantik die Verfahren des Abschnitts 4.4.3 zum dynamischen Test des beschriebenen Zustandsübergangsmodells zur Anwendung kommen. Da die mit Hilfe von Petri-Netzen beschriebenen Zustandsübergangsmodelle meist eine sehr große, wenn nicht unendliche Zahl von Zuständen enthalten, ist ein dynamischer Test selbst mit einfachen zustandsbasierten Verfahren möglicherweise zu umfangreich. Eine Prüfung nach schwächeren Kriterien, die direkt auf den Strukturelementen des Petri-Netzes operieren, bietet sich in diesen Fällen an.

Um exemplarisch eine einfache Gestaltungsmöglichkeit des funktionsorientierten Tests auf Basis von Petri-Netzen aufzuzeigen, sei im Folgenden angenommen, dass ein als Teilspezifikation beschriebenes Petri-Netz Aspekte der Funktionalität eines Prüflings beschreibe. Hierzu werde ein Stellen-Transitions-Netz verwendet, dessen Transitionen mit einer maximalen Schaltdauer versehen sind. Eine graphentheoretische Notation kann wie folgt gegeben werden.

Definition 4-32: Graphentheoretische Notation eines Stellen-Transitions-Netzes mit Angabe der maximalen Schaltzeiten der Transitionen, zeitbehaftetes Stellen-Transitions-Netz

Ein Stellen-Transitions-Netz mit Angabe der maximalen Schaltzeiten der Transitionen wird durch ein Tupel (St, Tr, E, Gewicht, Schaltzeit) beschrieben, das die folgenden Bedingungen erfüllt:

(a) (V, E) mit Vereinigungsmenge $V = St \cup Tr$ der disjunkten Mengen St und Tr bildet einen gerichteten Graphen.

(b) Für die Menge der Kanten gilt $E \subseteq (St \times Tr) \cup (Tr \times St)$, so dass (St, Tr, E) einen bipartiten gerichteten Graphen darstellt.

(c) Die Abbildung

$$\text{Gewicht}: \quad E \quad \rightarrow \quad \aleph \setminus \{ 0 \}$$

weist den Kanten eine Gewichtung zu, die den Markenverbrauch und die Markenerzeugung in den verbundenen Stellen beim Schalten der verbundenen Transition beschreibt.

(d) Die Abbildung

$$\text{Schaltzeit}: \quad Tr \quad \rightarrow \quad \Re^{+}$$

weist den Transitionen eine maximal zulässige Schaltzeit nach ihrer Aktivierung zu.

Ein Stellen-Transitions-Netzes mit Angabe der maximalen Schaltzeiten der Transitionen wird im Folgenden abkürzend als zeitbehaftetes Stellen-Transitions-Netz bezeichnet.

Diese hier verwendete einfache Beschreibung des Zeitverhaltens eignet sich zur Angabe von Performanzbedingungen und hat gleichzeitig den Vorteil, dass der Erreichbarkeitsgraph dem des entsprechenden, nicht zeitbehafteten Stellen-Transitions-Netzes entspricht. Die Erreichbarkeit beliebiger Zielmarkierungen ist somit entscheidbar [Kos82], was bei Petri-Netzen mit einer erweiterten zeitlichen Semantik oft nicht garantiert werden kann [JonLanLie77, BerDia91].

Eine Markierung der Stellen ist in der obigen Definition noch nicht beschrieben. Diese kann bei vorausgesetzter Gleichartigkeit der Marken als Zuordnung von natürlichen Zahlen zu den Stellen beschrieben werden.

Definition 4-33: Markierung eines zeitbehafteten Stellen-Transitions-Netzes

Als Markierung eines zeitbehafteten Stellen-Transitions-Netzes pn = (St, Tr, E, Gewicht, Schaltzeit) wird die Abbildung

Mark: St $\rightarrow$ $\aleph$

bezeichnet. Die Menge aller Markierungen wird im Folgenden mit M_{pn} bezeichnet.

Da im hier betrachteten Stellen-Transitions-Netz die Kapazität der Stellen nicht beschränkt wurde, besteht auch keine Einschränkung bezüglich der Markierungen, die Mächtigkeit der Menge M_{pn} ist damit unendlich.

Das dynamische Verhalten des Petri-Netzes ergibt sich nach Angabe einer Startmarkierung durch die Schaltregeln. Es wird durch den Erreichbarkeitsgraphen dargestellt. Dieser ist wegen der gewählten Netzsemantik berechenbar und enthält alle Informationen über erreichte Markierungen, geschaltete Transitionen und belegte Stellen. Für eine vollständige Definition des Erreichbarkeitsgraphen sei auf [Bau96] verwiesen. Die in der weiteren Betrachtung benötigten Informationen werden wie folgt referenziert.

Definition 4-34: Erreichte Markierungen, erreichte Transitionen und erreichte Stellen zu einem zeitbehafteten Stellen-Transitions-Netz und einer Startmarkierung

Zu einem zeitbehafteten Stellen-Transitions-Netz pn = (St, Tr, E, Gewicht, Schaltzeit) wird im Folgenden der Erreichbarkeitsgraph zu einem Startzustand start $\in M_{pn}$ mit Err_{pn}(start) bezeichnet.

Die im Erreichbarkeitsgraphen Err_{pn}(start) enthaltenen Markierungen werden im Folgenden als Markierungen(Err_{pn}(start)) bezeichnet.

Die im Erreichbarkeitsgraphen Err_{n}(start) geschalteten Transitionen werden im Folgenden als Transitionen(Err_{pn}(start)) bezeichnet.

Die im Erreichbarkeitsgraphen Err_{pn}(start) mit mindestens einer Marke belegten Stellen werden im Folgenden als Stellen(Err_{pn}(start)) bezeichnet.

Um ein Petri-Netz als Teil einer Spezifikation verstehen zu können und seine strukturelle Überdeckung im dynamischen Test zu formulieren, müssen Annahmen darüber getroffen werden, wie der semantische Bezug zwischen Spezifikation und Petrinetz zu verstehen ist. Die im Folgenden exemplarisch verwendeten Annahmen orientieren sich an der in [DesEA97] beschriebenen Verwendung von Petri-Netzen. Sie sind restriktiv, gestatten aber eine einfache semantische Interpretation des Petri-Netzes als Spezifikation. So sei im Folgenden angenommen, dass die Eingabedaten wie folgt mit den Startmarkierungen assoziiert werden können.

Definition 4-35: Startmarkierungsassoziation über dem Eingabedatenbereich

Zu einem zeitbehafteten Stellen-Transitions-Netz pn = (St, Tr, E, Gewicht, Schaltzeit) wird eine Zuordnung

$$\begin{array}{lccl} StartAssoz_s: & I & \rightarrow & M_n \\ & i & \mapsto & StartAssoz_s(i), \end{array}$$

die jedem Eingabedatum eine Startmarkierung zuweist, als Startmarkierungsassoziation über dem Eingabedatenbereich bezeichnet.

Die Annahme der Vollständigkeit der Startmarkierungsassoziation über dem Eingabebereich I ist auf die Vollständigkeit der Spezifikation nach Definition 3-2 zurückzuführen; ungültige Eingaben, die eine Fehlerbehandlung erfordern, können ohne Beschränkung der Allgemeinheit beispielsweise durch eine entsprechende Markierung spezieller Stellen abgebildet werden. Über die Annahme der Startmarkierungsassoziation hinaus kann angenommen werden, dass der zu den Eingabedaten assoziierten Startmarkierungen gehörende Erreichbarkeitsgraph endlich und zyklenfrei ist. In diesem Fall werden den Startmarkierungen durch das Petri-Netz verschiedene Endmarkierungen zugewiesen, die dadurch gekennzeichnet sind, dass sie keiner weiteren Transition das Schalten ermöglichen. Dies ermöglicht die eindeutige Identifikation der zugeordneten Ausgabedaten und die Berechung der maximal zulässigen Berechnungsdauer. Diese Annahmen ermöglichen ein einfaches Verständnis des Petri-Netzes als Spezifikation über dem Ein- und Ausgabebereich. Bei Bedarf können diese Annahmen erweitert werden; die folgenden Definitionen und Verfahren sind dann entsprechend anzupassen.

Mit Hilfe der Startmarkierungsassoziation kann der Begriff einer Überdeckung von Strukturelementen eines zeitbehafteten Stellen-Transitions-Netzes formuliert werden.

Definition 4-36: Überdeckung der Strukturelemente eines zeitbehafteten Stellen-Transitions-Netzes durch dynamische Testfälle

Vorausgesetzt wird die Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch ein zeitbehaftetes Stellen-Transitions-Netz pn = (St, Tr, E, Gewicht, Schaltzeit) und die Vorgabe einer Startmarkierungsassoziation $StartAssoz_s$.

(a) Die Überdeckung $trcov_s \subseteq I \times Tr$ von Transitionen eines zeitbehafteten Stellen-Transitions-Netzes n durch die Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(tr \in Tr)\ (((i, tr) \in trcov_s) :\Leftrightarrow (tr \in Transitionen(Err_{pn}(StartAssoz_s(i)))))$$

über der Menge der Eingabedaten und der Transitionen.

(b) Die Überdeckung $stcov_s \subseteq I \times St$ von Stellen eines zeitbehafteten Stellen-Transitions-Netzes n durch die Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(st \in St)\ (((i, st) \in stcov_s) :\Leftrightarrow (tr \in Stellen(Err_{pn}(StartAssoz_s(i)))))$$

über der Menge der Eingabedaten und der Stellen.

(c) Die Überdeckung $mcov_s \subseteq I \times M_{pn}$ von Markierungen eines zeitbehafteten Stellen-Transitions-Netzes n durch die Eingabedaten wird definiert durch die zweistellige Relation

$$\forall(i \in I)\ \forall(m \in M_{pn})\ (((i, m) \in mcov_s) :\Leftrightarrow (m \in Markierungen(Err_{pn}(StartAssoz_s(i)))))$$

über der Menge der Eingabedaten und der möglichen Markierungen.

Diese einfachen Überdeckungsbegriffe erlauben die Formulierung von Kriterien für den dynamischen Test, die eine Überdeckung der Strukturelemente des spezifizierten zeitbehafteten Stellen-Transitions-Netzes prüfen. So zielt das folgende Verfahren auf eine Überdeckung der Transitionen des spezifizierten Netzes im dynamischen Test ab.

Verfahren 4-25: Transitionsüberdeckung eines zeitbehafteten Stellen-Transitions-Netzes

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch ein zeitbehaftetes Stellen-Transitions-Netz pn = (St, Tr, E, Gewicht, Schaltzeit) und eine Startmarkierungsassoziation $StartAssoz_s$.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Transitionsüberdeckung eines zeitbehafteten Stellen-Transitions-Netzes wird beschrieben durch

$$X^{Tcov}: \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(tr \in Tr)\, \exists(t \in T)\, ((t, tr) \in trcov_s).$$

Entsprechend kann die Markierung einer jeden Stelle im dynamischen Test durch das folgende Verfahren verlangt werden.

Verfahren 4-26: Stellenüberdeckung eines zeitbehafteten Stellen-Transitions-Netzes

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch ein zeitbehaftetes Stellen-Transitions-Netz pn = (St, Tr, E, Gewicht, Schaltzeit) und eine Startmarkierungsassoziation $StartAssoz_s$.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Stellenüberdeckung eines zeitbehafteten Stellen-Transitions-Netzes wird beschrieben durch

$$X^{Scov}: \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I))$$
$$s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall(st \in St)\, \exists(t \in T)\, ((t, st) \in stcov_s).$$

Darüber hinaus kann die Erreichung gewisser vorgegebener Markierungen im dynamischen Test verlangt werden.

Verfahren 4-27: Markierungsüberdeckung eines zeitbehafteten Stellen-Transitions-Netzes

(a) Voraussetzung:

Gegeben sei eine Beschreibung der Spezifikation $s \in \Sigma_{I, O}$ durch ein zeitbehaftetes Stellen-Transitions-Netz pn = (St, Tr, E, Gewicht, Schaltzeit) und eine Startmarkierungsassoziation $StartAssoz_s$. Weiterhin sei eine endliche Menge von Markierungen $Emark \subseteq M_{pn}$ beschrieben, die im dynamischen Test erreicht werden soll.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Markierungsüberdeckung eines zeitbehafteten Stellen-Transitions-Netzes wird beschrieben durch

$$X^{Mcov,\,Emark}: \quad \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I)), \quad s \mapsto \{ T \in \wp^{fin}(I): P_s(T) \}$$

mit

$$P_s(T) := \forall (m \in Emark)\ \exists (t \in T)\ ((t, m) \in mcov_s).$$

Eine Erweiterung von Verfahren 4-27 auf alle Markierungen der Menge M_n ist nicht sinnvoll, da diese Menge unendliche Mächtigkeit hat, während unter den hier zugrunde gelegten Annahmen nur eine Erreichung endlich vieler Markierungen in einem dynamischen Testfall und somit auch in einem dynamischen Testlauf erreicht werden kann. Werden im Stellen-Transitions-Netz zusätzlich Stellenkapazitäten für jede Stelle angegeben, so sind nur endlich viele Markierungen möglich, deren vollständige Überdeckung im dynamischen Test gefordert werden kann. Die Anwendbarkeit des resultierenden Verfahrens hängt dann von der Erreichbarkeit der Markierung ausgehend von den zu Eingabedaten assoziierten Startmarkierungen ab.

Die Anwendbarkeit ist auch für die drei oben angegebenen Verfahren explizit zu prüfen. So muss bei der Angabe der zu überdeckenden Strukturelemente deren theoretische Erreichbarkeit stets berücksichtigt werden. Dies stellt, wie oben schon bemerkt, für das hier angenommene Zeitverhalten ein entscheidbares Problem dar.

Eine Herleitung weiterer Überdeckungskriterien in Analogie zu den oben beschriebenen Verfahren ist einfach möglich. So kann beispielsweise für jede Stelle geprüft werden, ob sie während des dynamischen Tests gewisse vorgegebene Markierungen enthalten hat. Beispielsweise kann es sinnvoll sein zu prüfen, ob gewisse Stellen im dynamischen Testlauf leer gewesen sind. Bei zusätzlicher Angabe von Stellenkapazitäten kann geprüft werden, ob die Maximalkapazität einer Stelle erreicht wurde. Die Ableitung solcher Kriterien kann im Hinblick auf Fehlererwartungen bezüglich der Implementierung vorgenommen werden.

Die hier angegebenen dynamischen Testverfahren sind lediglich als Beispiele zu verstehen. Sie sollen an der hier gewählten Spezifikationssemantik aufzeigen, dass die Formulierung von dynamischen Testverfahren zur Prüfung der in einem Petri-Netz beschriebenen Funktionalität orientiert an den Strukturelementen des Netzes möglich ist. Vor dem Hintergrund dieser beispielhaften Beschreibungen ist es nahe liegend, weitere Verfahren für die Prüfung der in einem Petri-Netz spezifizierten Funktionalität zu entwickeln. Da der Einsatz von Petri-Netzen in Spezifikationen der Praxis sehr unterschiedlichen Charakter haben kann, sind insbesondere die möglichen Interpretationen eines Petri-Netzes als Spezifikation über einem gegebenen Ein- und Ausgabebereich zu untersuchen. Diese haben Einfluss auf die Assoziationsrelation aus Definition 4-35 und damit auf die Überdeckungsrelationen aus

Definition 4-36. Möglichkeiten zur Standardisierung und Automatisierung solcher Verfahren ergeben sich insbesondere dann, wenn eine Bezugnahme zwischen der syntaktischen Netzbeschreibung einerseits und dem Eingabe-, Ausgabe- und Zeitbereich andererseits erreicht werden kann.

Es ist zu vermuten, dass die praktische Einsetzbarkeit der hier exemplarisch entwickelten Verfahren des dynamischen Tests auf Basis einer Spezifikation mit Petri-Netzen gering ist, da die zahlreichen Voraussetzungen deutliche Modellierungseinschränkungen bedeuten. Es sollte daher in einer weiterführenden Untersuchung geprüft werden, ob eine Aufhebung der hier angenommenen Ein-Schritt-Verarbeitungssemantik möglich ist. Somit könnte eine Erweiterung der hier formulierten Überdeckungskriterien auf zustandbasierte Systeme, die mit Hilfe von Petri-Netzen spezifiziert wurden, erreicht werden.

4.4 Verfahren zum Test zustandsbasierter Systeme

In den vorangegangen Abschnitten wurden Verfahren analysiert, die den dynamischen Test spezieller, nach funktionalen Gesichtspunkten ausgewählter Repräsentanten des Ein- oder Ausgabebereich fordern. Alle diese Verfahren betrachten den Prüfling als „gedächtnislos“, gehen also davon aus, dass keinerlei Informationsfluss zwischen den dynamischen Testfällen stattfindet. Diese Annahme ist in vielen Fällen nicht erfüllt: So verfügen Softwareprogramm im Allgemeinen über einen Speicher, der während der Programmlaufzeit und darüber hinaus Informationen festhält. Dieses „Gedächtnis“ ist meist ein wesentlicher Bestandteil der Realisierung und somit auch im dynamischen Test zu berücksichtigen. Es wird oft im Hauptspeicher verwaltet, kann aber auch als Datei oder in Form einer Datenbank abgelegt sein. In technischen Systemen wird das Gedächtnis eines Systems auch durch die zugehörige Hardware beeinflusst, beispielsweise durch die Position von Schalten, Hebeln oder sonstigen Komponenten im Raum. Falls ein solches Gedächtnis Auswirkungen auf das funktionale Verhalten des Prüflings hat, muss es im dynamischen Test berücksichtigt werden. Eine systematische Analyse des Verhaltens des Prüflings unter Berücksichtigung seines Gedächtnisses leisten zustandsbasierte dynamische Testtechniken, die im Folgenden diskutiert werden.

Als Grundlage zur Betrachtung der zustandbasierten dynamischen Testtechniken werden in Abschnitt 4.4.1 die Definitionen aus Abschnitt 3.2 für gedächtnisbehaftete Prüflinge und Spezifikationen erweitert. In Abschnitt 4.4.2 werden Beschreibungsformen für zustandsbasierte Spezifikationen vorgestellt, auf denen die in Abschnitt 4.4.3 diskutierten zustandsbasierten Testtechniken operieren.

4.4.1 Formale Beschreibung des dynamischen Tests zustandsbasierter Systeme

Um das Gedächtnis eines Prüflings und damit seinen Zustand vor und nach Durchführung eines dynamischen Testfalls in einem dynamischen Testverfahren berücksichtigen zu können, sind Anpassungen der Definitionen aus Abschnitt 3.2 vorzunehmen. Hierzu gibt es verschiedene Möglichkeiten, die bereits in Abschnitt 3.3.1 genannt wurden. Im Folgenden wird der in [TurRob93] und [Voa99] beschriebene Ansatz verfolgt, der eine Erweiterung des Ein- und Ausgabebereichs um den Systemzustand vorschlägt: So wird der Zustand des Prüflings vor Ausführung eines dynamischen Testfalls als weiterer Eingabeparameter behandelt, und ebenso wird der Zustand nach Ausführung des dynamischen Testfalls als Teil des Testergebnisses und damit als Teil des Ausgabebereichs betrachtet. Diese Erweiterung ermöglicht die Beibehaltung des Grundverständnisses des Testprozesses aus Abschnitt 3.2.2 bei einer geringfügigen Modifikation einzelner Definitionen. Weiterhin eignet sie sich sehr

gut, um das Vorgehen im zustandsbasierten Test in der Praxis zu beschrieben, bei der der erreichte Systemzustand zwischen den Testfällen nach Möglichkeit auf Übereinstimmung mit der Spezifikation geprüft wird.

Um die Spezifikation des Gedächtnisses und ihre Realisierung im Prüfling adäquat modellieren zu können, wird die folgende Definition eines Speicherbereichs gegeben.

Definition 4-37: Speicherbereich, Speicherparameterbereich, Speicherparameter

Als Speicherbereich zu einem Prüfling oder einer Spezifikation wird eine beliebige Menge H zugelassen. Falls der Speicherbereich als kartesisches Produkt

$$H = H_1 \times H_2 \times \ldots \times H_m \qquad (m \in \aleph)$$

einer endlichen Zahl beliebiger Mengen angegeben ist, werden die Mengen H_1, H_2, …, H_m als Speicherparameterbereiche bezeichnet und die Elemente h von H als Tupel

$$h = (h_1, h_2, \ldots, h_m) \qquad (h_1 \in H_1, h_2 \in H_2, \ldots, h_m \in H_m, m \in \aleph)$$

der Speicherparameter h_1, h_2, …, h_m beschrieben.

Die Definition beschreibt den Speicherbereich zur Spezifikation oder Realisierung eines Systemgedächtnisses möglichst allgemein durch die Menge der möglichen Inhalte. Dadurch bestehen keinerlei Einschränkungen bezüglich der darstellbaren Speicherzustände. Die Darstellung ermöglicht eine Betrachtung einzelner Parameter, die beispielsweise mit Speicheradressen im System oder mit Hardwarekomponenten assoziiert sein können. Die Erweiterung des Ein- und Ausgabebereichs um einen Speicherbereich wird nun wie folgt vorgenommen.

Definition 4-38: Erweiterter Ein- und Ausgabebereich

Zu einem beliebigen Eingabebereich I, Ausgabebereich O und Speicherbereich H werden die kartesischen Produkte $I \times H$ und $O \times H$ als erweiterter Ein- beziehungsweise Ausgabebereich bezeichnet.

Diese Definitionen ermöglicht die Anpassung von Definition 3-2 einer Spezifikation im Hinblick auf das Systemgedächtnis.

Definition 4-39: Gedächtnisbehaftete Spezifikation über einem erweiterten Ein- und Ausgabebereich

(a) Eine Spezifikation s über einem erweiterten Eingabebereich $I \times H$ und erweiterten Ausgabebereich $O \times H$ wird durch eine Relation über dem erweiterten Ein- und Ausgabebereich und den positiven reellen Zahlen beschrieben, also durch eine Teilmenge des kartesischen Produktes $I \times H \times O \times H \times \Re^+$, die zusätzlich der Bedingung

$$\forall((i, h_1) \in I \times H)\ \exists((o, h_2) \in O \times H)\ \exists(r \in \Re^+)\ (i, h_1, o, h_2, r) \in s \qquad \text{(Vollständigkeit)}$$

genügt. Sie wird im Folgenden als gedächtnisbehaftete Spezifikation oder als Spezifikation über einem erweiterten Ein- und Ausgabebereich bezeichnet.

(b) $\Sigma_{I, O, H}$ bezeichnet im Folgenden die Menge aller Spezifikationen über dem erweiterten Eingabebereich $I \times H$ und dem erweiterten Ausgabebereich $O \times H$.

Häufig wird in der Spezifikation eines gedächtnisbehafteten Systems über einem erweiterten Ein- und Ausgabebereich nicht jedes Element des Speicherbereichs gesondert betrachtet, sondern eine vereinfachte Beschreibung im Hinblick auf spezielle Teilmengen des Speicherbereichs gegeben. Hierzu werden die Elemente des Speicherbereichs nach funktionalen Aspekten zu Teilmengen zusammengefasst, für die ein äquivalentes funktionales Verhalten des Prüflings gefordert wird. Die folgende Definition beschreibt eine solche zustandsbasierte Spezifikation.

Definition 4-40: Zustand über dem Speicherbereich, zustandsbasierte und zustandsendliche Spezifikation

(a) Falls in einer gegebenen gedächtnisbehafteten Spezifikation $s \in \Sigma_{I, O, H}$ über dem erweiterten Ein- und Ausgabebereich für verschiedene Elemente der Speicherbereichs eine äquivalente funktionale Behandlung der Eingabedaten aus I gefordert ist, so impliziert dies eine Äquivalenzrelation $\sim_{(fäq)}$ zwischen den Elementen des Speicherbereichs, also eine reflexive, symmetrische und transitive Relation [ReiSoe01]. Fasst man die funktional äquivalenten Elemente in Partitionen zusammen, so entsteht eine disjunkte Partitionierung des Speicherbereiches H, die im Folgenden mit $H / \sim_{(fäq)}$ bezeichnet wird. Die Partitionen von $H / \sim_{(fäq)}$ werden im Folgenden als Zustände über dem Speicherbereich bezeichnet.

(b) Ermöglicht eine Spezifikation über einem erweiterten Ein- und Ausgabebereich die Bildung von Zuständen mit mehr als einem Element, so wird sie im Folgenden als zustandsbasiert bezeichnet.

(c) Besitzt eine zustandsbasierte Spezifikation nur endlich viele Zustände, so wird sie im Folgenden als zustandsendlich bezeichnet.

Ebenso wie für die Zustände des Speicherbereichs kann auch eine funktionale Partitionierung des Eingabebereichs möglich sein. In einer zustandsbasierten Spezifikation können disjunkte funktionale Partitionen des Eingabebereichs, für die eine spezielle Systemreaktion abhängig vom aktuellen Zustand spezifiziert ist, als Ereignisse betrachtet werden. Ist eine funktionale Partitionierung des Eingabebereichs nicht möglich, kann jedes Eingabedatum als Ereignis verstanden werden.

Es kann darüber hinaus hilfreich sein, eine zustandsbasierte Spezifikation als Menge von Teilspezifikationen zu verstehen, die für jeden Zustand das funktionale Verhalten in Form einer Ein- und Ausgaberelation beschreibt. Bezüglich der Art der funktionalen Äquivalenz der Speicherdaten eines Zustands wird nichts vorausgesetzt. Sie muss lediglich eine einheitliche Beschreibung der Ein- und Ausgaberelation unabhängig vom vorliegenden Speicherdatum ermöglichen. Unterschiedliche Speicherdaten eines Zustands können in die Zuordnung der Ausgabedaten zu den Eingabedaten durchaus einfließen, solange dies einheitlich, beispielsweise in Form eines funktionalen Zusammenhangs, beschrieben werden kann. Aus diesen Überlegungen wird deutlich, dass die zustandsbasierte Formulierung der Spezifikation eine genaue Analyse des intendierten Systemverhaltens voraussetzt und als Modellierung einen ersten Schritt im Formalisierungsprozess der Softwareentwicklung darstellt.

Ebenso wie die Definition der Spezifikation kann die Definition 3-4 eines Prüflings an den erweiterten Ein- und Ausgabebereich angepasst werden.

Definition 4-41: Gedächtnisbehafteter Prüfling über einem erweiterten Eingabe- und Ausgabebereich

(a) Ein gedächtnisbehafteter Prüfling p über dem erweiterten Eingabebereich $I \times H$ und erweiterten Ausgabebereich $O \times H$ wird beschrieben durch eine partielle Funktion des Eingabebereichs auf das kartesische Produkt des Ausgabebereichs und der positiven reellen Zahlen

$$p: \quad I \times H \rightarrow (O \times H \times \Re^+) \cup \perp$$

$$(i, h) \mapsto \begin{cases} (p_{out}(i,h), p_{ged}(i,h), p_{perf}(i,h)) & \text{falls p für } (i,h) \text{ terminiert,} \\ \perp & \text{sonst.} \end{cases}$$

Dabei bezeichnet $p_{out}(i, h)$ die vom Prüfling berechneten Ausgabedaten, $p_{ged}(i, h)$ die vom Prüfling berechneten Speicherdaten und $p_{perf}(i, h)$ die benötigte Berechnungszeit, falls p für (i, h) terminiert. Der Prüfling p wird im Folgenden als gedächtnisbehaftet oder als Prüfling über einem erweiterten Ein- und Ausgabebereich bezeichnet.

(b) $\Pi_{I, O, H}$ bezeichnet im Folgenden die Menge aller Prüflinge über dem erweiterten Eingabebereich $I \times H$ und Ausgabebereich $O \times H$.

Für die Korrektheit eines Prüflings bezüglich einer Spezifikation über einem erweiterten Ein- und Ausgabebereich ergibt sich analog zu Definition 3-5 die folgende Definition.

Definition 4-42: Korrektheit von Prüflingen bezüglich Spezifikationen über einem erweiterten Ein- und Ausgabebereich

Die Korrektheit $corr_{I, O, H} \subseteq \Pi_{I, O, H} \times \Sigma_{I, O, H}$ wird definiert durch die zweistellige Relation

$\forall(p \in \Pi_{I, O, H}) \; \forall(s \in \Sigma_{I, O, H})$

$((p, s) \in corr_{I, O, H}) :\Leftrightarrow \forall((i, h) \in I \times H) \; ((p(i, h) \neq \perp) \wedge ((i, h, p_{out}(i, h), p_{ged}(i, h), p_{perf}(i, h)) \in s)$

über der Menge der Prüfling und der Spezifikationen über dem erweiterten Ein- und Ausgabebereich.

Auch die Durchführung und Auswertung eines dynamischen Testfalls, eines dynamischen Testlaufs und eines dynamischen Testverfahrens für Prüflinge und Spezifikationen über einem erweiterten Ein- und Ausgabebereich können analog zu den in Abschnitt 3.2 gegebenen Definitionen formuliert werden. So ergibt sich für einen dynamischen Testfall und sein Bestehen entsprechend Definition 3-6 das Folgende.

Definition 4-43: Dynamischer Testfall, Bestehen dynamischer Testfälle über einem erweiterten Ein- und Ausgabebereich

(a) Ein dynamischer Testfall für einen Prüfling $p \in \Pi_{I, O, H}$ bezüglich einer Spezifikation $s \in \Sigma_{I, O, H}$ über dem erweiterten Ein- und Ausgabebereich $I \times H$ beziehungsweise $O \times H$ wird durch die Anwendung des Prüflings auf ein Testdatum $t = (i, h) \in I \times H$ durchgeführt.

(b) Ein dynamischer Testfall über einem erweiterten Ein- und Ausgabebereich gilt dann als bestanden, wenn das Ergebnis der Testdurchführung der gedächtnisbehafteten Spezifikation entspricht. Das Bestehen von dynamischen Testfällen eines Prüflings bezüglich einer Spezifikation wird durch die Relation $ok_{I, O, H} \subseteq I \times H \times \Pi_{I, O, H} \times \Sigma_{I, O, H}$ mit

$\forall(i \in I)\ \forall(h \in H)\ \forall(p \in \Pi_{I, O, H})\ \forall(s \in \Sigma_{I, O, H})$

$((i, h, p, s) \in ok_{I, O, H}) :\Leftrightarrow ((p(i, h) \neq \perp) \wedge ((i, h, p_{out}(i, h), p_{ged}(i, h), p_{perf}(i, h)) \in s)$

beschrieben.

Für einen gegebenen gedächtnisbehafteten Prüfling und eine gegebene gedächtnisbehaftete Spezifikation ist somit ein dynamischer Testfall durch das Testdatum t = (i, h) vollständig beschrieben. Ein dynamischer Testlauf über dem erweiterten Ein- und Ausgabebereich kann nun, analog zu Definition 3-7, als nichtleere Menge von dynamischen Testfällen über dem erweiterten Ein- und Ausgabebereich definiert werden. Sein Bestehen wird mit einer Relation $ok'_{I, O, H}$ beschrieben, die durch das Bestehen aller enthaltenen dynamischen Testfälle bestimmt ist.

Definition 4-44: Dynamischer Testlauf, Bestehen dynamischer Testläufe über einem erweiterten Ein- und Ausgabebereich

(a) Ein dynamischer Testlauf für den Prüfling $p \in \Pi_{I, O, H}$ bezüglich einer Spezifikation $s \in \Sigma_{I, O, H}$ über dem erweiterten Ein- und Ausgabebereich $I \times H$ beziehungsweise $O \times H$ besteht aus einer nichtleeren, endlichen Menge dynamischer Testfälle. Er wird durch die zugrunde gelegte Menge der Testdaten $T \in \wp^{fin}(I \times H)$ beschrieben.

(b) Ein dynamischer Testlauf T über einem erweiterten Ein- und Ausgabebereich gilt dann als bestanden, wenn alle dynamischen Testfälle $t \in T$ bestanden wurden. Das Bestehen von dynamischen Testläufen durch einen Prüfling bezüglich einer Spezifikation wird durch die dreistellige Relation $ok'_{I, O, H} \subseteq \wp^{fin}(I \times H) \times \Pi_{I, O, H} \times \Sigma_{I, O, H}$ mit

$\forall(T \in \wp^{fin}(I \times H))\ \forall(p \in \Pi_{I, O, H})\ \forall(s \in \Sigma_{I, O, H})$

$(((T, p, s) \in ok'_{I, O, H}) :\Leftrightarrow (\forall(t \in T)\ ((t, p, s) \in ok_{I, O, H})))$

beschrieben.

Entsprechend Definition 3-8 leistet ein dynamisches Testverfahren über dem erweiterten Ein- und Ausgabebereich die Zuordnung einer Menge dynamischer Testläufe zu einem gegebenen gedächtnisbehafteten Prüfling und einer gegebenen gedächtnisbehafteten Spezifikation.

Definition 4-45: Dynamisches Testverfahren, Bestehen eines dynamischen Testverfahrens, Adäquatheit eines dynamischen Testlaufs über einem erweiterten Ein- und Ausgabebereich

(a) Ein dynamisches Testverfahren X über einem erweiterten Ein- und Ausgabebereich beschreibt eine Zuordnung

$$X: \quad \Pi_{I, O, H} \times \Sigma_{I, O, H} \rightarrow \wp(\wp^{fin}(I \times H))$$
$$(p, s) \mapsto X(p, s),$$

die einem beliebigen gedächtnisbehafteten Prüfling $p \in \Pi_{I, O, H}$ und einer beliebigen gedächtnisbehafteten Spezifikation $s \in \Sigma_{I, O, H}$ eine Menge dynamischer Testläufe $X(p, s) \in \wp(\wp^{fin}(I))$ zuordnet und gleichzeitig die Bedingung

$\forall(s \in \Sigma_{I, O, H})\ \forall(p \in \Pi_{I, O, H})\ \forall(T \in X(p, s))\ \forall(M \in \wp^{fin}(I \times H))$

$((M \supseteq T) \Rightarrow (M \in X(p, s)))$ (Monotonie)

erfüllt.

(b) Im Folgenden bezeichnet $\Xi_{I, O, H} := \{ X : \Pi_{I, O, H} \times \Sigma_{I, O, H} \rightarrow \wp(\wp^{fin}(I \times H)) \}$ die Menge aller dynamischen Testverfahren, die gedächtnisbehafteten Prüflingen aus $\Pi_{I, O, H}$ und Spezifikationen aus $\Sigma_{I, O, H}$ eine Menge von dynamischen Testläufen zuordnet.

(c) Ein dynamisches Testverfahren $X \in \Xi_{I, O, H}$ gilt dann vom gedächtnisbehafteten Prüfling $p \in \Pi_{I, O, H}$ und der gedächtnisbehafteten Spezifikation $s \in \Sigma_{I, O, h}$ als bestanden, wenn es ein mindestens einen dynamischen Testlauf $T \in X(p, s)$ gibt, der bestanden wird. Das Bestehen von dynamischen Testverfahren durch einen Prüfling bezüglich einer Spezifikation wird daher durch die dreistellige Relation $ok''_{I, O, H} \subseteq \Xi_{I, O, H} \times \Pi_{I, O, H} \times \Sigma_{I, O, H}$ mit

$$\forall(X \in \Xi_{I, O, H}) \; \forall(p \in \Pi_{I, O, H}) \; \forall(s \in \Sigma_{I, O, H})$$
$$((X, p, s) \in ok''_{I, O, H}) :\Leftrightarrow (\exists(T \in X(p, s)) \; ((T, p, s) \in ok'_{I, O, H}))$$

beschrieben.

(d) Alle dynamischen Testläufe, die durch das Verfahren einem gedächtnisbehafteten Prüfling und einer gedächtnisbehafteten Spezifikation zugeordnet werden, werden als adäquat für Prüfling und Spezifikation bezüglich des Verfahrens bezeichnet.

Auch hier wird die Monotonie des Verfahrens vorausgesetzt, die es erlaubt, beliebige dynamische Testfälle zu einem dynamischen Testlauf hinzuzunehmen, ohne dass die Adäquatheit bezüglich des dynamischen Testverfahrens gefährdet wird. Die Bestehensrelation $ok''_{I, O, H}$ für ein dynamisches Testverfahren wird aus Relation $ok'_{I, O, H}$ abgeleitet, so dass es für einen gedächtnisbehafteten Prüfling und eine gedächtnisbehaftete Spezifikation genau dann als bestanden gilt, falls mindestens einer der zugehörigen dynamischen Testläufe über dem erweiterten Ein- und Ausgabebereich bestanden wird. Dies ermöglicht die kritische Betrachtung des Bestehens von dynamischen Testfällen durch einen möglicherweise nicht bezüglich seiner Spezifikation korrekten gedächtnisbehafteten Prüfling. Die alternative Forderung, für ein Bestehen des dynamischen Testverfahrens das Bestehen aller adäquaten dynamischen Testläufe zu verlangen, würde mit der angenommenen Monotonie dazu führen, dass nur gedächtnisbehaftete Prüflinge, die korrekt sind bezüglich ihrer gedächtnisbehafteten Spezifikation, ein Verfahren bestehen können.

Auch die weiteren, in Anschnitt 3.2 beschriebenen Eigenschaften eines dynamischen Testverfahrens können auf gedächtnisbehaftete Prüflinge und Spezifikationen bezogen werden. So wird die Anwendbarkeit eines dynamischen Testverfahrens über einem erweiterten Ein- und Ausgabebereich entsprechend Definition 3-9 voraussetzen, dass für jeden gedächtnisbehafteten Prüfling und jede gedächtnisbehaftete Spezifikation mindestens ein dynamischer Testlauf existiert, der bezüglich des Verfahrens adäquat ist. Für ein dynamisches Testverfahren kann die Subsumption eines weiteren dynamischen Testverfahrens über demselben erweiterten Ein- und Ausgabebereich wie in Definition 3-10 durch eine Teilmengenbeziehung der dynamischen Testläufe für alle gedächtnisbehafteten Prüflinge und Spezifikationen beschrieben werden. Auch können dynamische Testläufe und dynamische Testverfahren über demselben erweiterten Ein- und Ausgabebereich analog zu Definition 3-11 beziehungsweise Definition 3-12 kombiniert werden. Das in Definition 3-13 beschriebene Testorakel kann auf den erweiterten Ein- und Ausgabebereich bezogen werden, indem es als Relation $or \subseteq I \times H \times O \times H \times \Re^+$ beschrieben werden. Die Vollständigkeit bezieht sich dann, ebenso wie in Definition 4-39, auf den erweiterten Eingabebereich $I \times H$. Die Akzeptanz dynamischer Testläufe kann für den erweiterten Ein- und Ausgabebereich entsprechend Definition 3-14 als Obermenge der Bestehensrelation $ok'_{I, O, H}$ be-

schrieben werden. Da sich diese Definitionen ohne Schwierigkeiten als Erweiterung der Definitionen aus Anschnitt 3.2 ergeben, werden sie hier nicht erneut formuliert.

Auch wenn sich die formale Erweiterung der Definitionen aus Kapitel 3 auf den erweiterten Ein- und Ausgabebereich soweit als unproblematisch erwiesen hat, wird in Definition 4-43 (b) eine besondere Schwierigkeit der Auswertung dynamischen Tests eines gedächtnisbehafteten Prüflings deutlich: So ist hier nicht nur die Ausgabe des Prüflings auf Konformität mit der Spezifikation zu überprüfen, sondern auch der erzeugte Inhalt des Speicherbereichs. Vereinfachend wird hierbei manchmal auch der Speicherzustand entsprechend Definition 4-40 geprüft, der sich durch Faktorisierung nach der funktionalen Äquivalenz ergibt. Grundsätzlich können zur Prüfung des Speicherzustands drei unterschiedliche Vorgehensweisen verfolgt werden: Wie in [Bei90] vorgeschlagen, ist eine Instrumentierung des Prüflings zur Ausgabe des erzeugten Speicherinhalts oder des erreichten Speicherzustands denkbar. Ist eine entsprechende Instrumentierung nicht möglich oder wird sie wegen des Eingriffs in die Realisierung des Prüflings abgelehnt, so kann – soweit möglich – ein externer Zugriff auf das Systemgedächtnis erfolgen. Je nach Realisierung des Systemgedächtnisses kann dies beispielsweise einen externer Zugriff auf einen elektronischen Speicherbereich oder eine beispielsweise visuelle Prüfung der Position von Hardwarekomponenten im Raum erfordern. Eine weitere Möglichkeit zur Prüfung des Zustands eines Prüflings kann aus der in [DooFra94] für abstrakte Datentypen definierten „beobachtbaren Äquivalenz“ abgeleitet werden, die auch für allgemeine gedächtnisbehaftete Systeme anwendbar ist: Doong und Frankl bezeichnen den Zustand zweier Systeme als beobachtbar äquivalent, wenn diese Systeme auf beliebige Sequenzen von Eingaben identisch reagieren, wenn also in ihrem potentiellen dynamischen Verhalten auf weitere Eingaben kein Unterschied beobachtet werden kann. Eine vollständige Prüfung der beobachtbaren Äquivalenz ist im Allgemeinen nicht möglich, doch legt sie die Feststellung des internen Zustands des Speicherbereichs durch Verhaltensprüfung beispielsweise im Vergleich mit einem zweiten System nahe, dessen Zustand bekannt ist.

Um im dynamischen Test eines gedächtnisbehafteten Prüflings bezüglich einer gedächtnisbehafteten Spezifikation nicht nur einzelne dynamische Testfälle beschreiben zu können, sondern ganze Sequenzen von Eingaben zum Test des dynamischen Verhaltens zu betrachten, kann die Definition 4-43 des dynamischen Testfalls zusätzlich erweitert werden. Die folgende induktive Definition führt den Begriff einer dynamischen Testsequenz ein und beschreibt ihre Durchführung und ihr Bestehen.

Definition 4-46: Dynamische Testsequenz, Durchführung einer dynamischen Testsequenz, Bestehen einer dynamischen Testsequenz über einem erweiterten Ein- und Ausgabebereich

Im Folgenden bezeichnet I^n ($n \in \aleph \setminus \{ 0 \}$) das n-fache kartesische Produkt des Eingabebereichs I mit sich selbst.

(a) Ein Element

$$(i_1, \ldots, i_n, h) \in I^n \times H \qquad (n \in \aleph \setminus \{ 0 \})$$

kann als dynamische Testsequenz der Länge n zu einem Prüfling $p \in \Pi_{I, O, H}$ und einer Spezifikation $s \in \Sigma_{I, O, H}$ über dem erweiterten Ein- und Ausgabebereich $I \times H$ beziehungsweise $O \times H$ bezeichnet werden.

(b) Die Durchführung einer dynamischen Testsequenz (i, h) der Länge 1 mit einem Prüfling $p \in \Pi_{I, O, H}$ wird als Anwendung

$$p(i, h) \in (O \times H \times \Re^+) \cup \bot \qquad \text{(Induktionsverankerung)}$$

des Prüflings p auf das Testdatum (i, h) gemäß Definition 4-43 definiert.

Bezeichnet $p(i_1, \ldots, i_n, h)$ die Durchführung der dynamischen Testsequenz $(i_1, \ldots, i_n, h)$ der Länge n für den Prüfling p, und kann deren Ergebnis gemäß Definition 4-41 mit

$$p(i_1, \ldots, i_n, h) = (p_{out}(i_1, \ldots, i_n, h), p_{ged}(i_1, \ldots, i_n, h), p_{perf}(i_1, \ldots, i_n, h)) \neq \perp$$

beschrieben werden, so wird die Durchführung einer dynamischen Testsequenz $(i_1, \ldots, i_n, i_{n+1}, h)$ der Länge n+1 durch

$$p(i_1, \ldots, i_n, i_{n+1}, h) := p(i_{n+1}, p_{ged}(i_1, \ldots, i_n, h)) \in (O \times H \times \Re^+) \cup \perp \quad \text{(Induktionsschritt 1)}$$

definiert.

Bezeichnet $p(i_1, \ldots, i_n, h)$ die Durchführung der dynamischen Testsequenz $(i_1, \ldots, i_n, h)$ der Länge n für den Prüfling p, und kann deren Ergebnis gemäß Definition 4-41 mit

$$p(i_1, \ldots, i_n, h) = \perp$$

beschrieben werden, so wird die Durchführung einer dynamischen Testsequenz $(i_1, \ldots, i_n, i_{n+1}, h)$ der Länge n+1 durch

$$p(i_1, \ldots, i_n, i_{n+1}, h) := \perp \quad \text{(Induktionsschritt 2)}$$

definiert.

(c) Eine dynamische Testsequenz (i, h) der Länge 1 gilt genau dann als vom gedächtnisbehafteten Prüfling p bezüglich der gedächtnisbehafteten Spezifikation s bestanden, wenn der dynamische Testfall t = (i, h) über dem erweiterten Eingabebereich durch den Prüfling p bezüglich der Spezifikation s bestanden wird. (Induktionsverankerung)

Eine dynamische Testsequenz $(i_1, \ldots, i_n, i_{n+1}, h)$ der Länge n+1 gilt genau dann als vom gedächtnisbehafteten Prüfling p bezüglich der gedächtnisbehafteten Spezifikation s bestanden, wenn die dynamische Testsequenz $(i_1, \ldots, i_n, h)$ vom Prüfling p bezüglich der Spezifikation s bestanden wird und zusätzlich der dynamische Testfall

$$t = (i_{n+1}, p_{ged}(i_1, \ldots, i_n, h)) \quad \text{(Induktionsschritt)}$$

vom Prüfling p bezüglich der Spezifikation s gemäß Definition 4-43 bestanden wird.

Eine dynamische Testsequenz entspricht somit einer endlichen Folge von dynamischen Testfällen, wobei sich der Gedächtniszustand eines dynamischen Testfalls aus dem Ergebnis des vorangegangenen Testfalls ergibt. Der initiale Zustand des Gedächtnisses des Prüflings ist anzugeben. Unter Rückgriff auf Definition 4-42 kann mittels vollständiger Induktion einfach gezeigt werden, dass für einen bezüglich einer gedächtnisbehafteten Spezifikation korrekten Prüfling jede Testsequenz bestanden wird.

Die Bestehensrelation für dynamische Testsequenzen ist in Definition 4-46 (c) bewusst streng formuliert. So wird nach jedem Testdatum der Sequenz die Ausgabe, die Antwortzeit sowie der Inhalt des Gedächtnisses überprüft. Da dies nicht immer möglich ist, können auch erweiterte Relationen für die Akzeptanz einer dynamischen Testsequenz definiert werden. Hierbei kann insbesondere eine Prüfung des Systemausgaben und der Antwortzeiten durchgeführt werden, die gleichzeitige Prüfung des Gedächtnisinhaltes aber unterbleiben. Dies ist insbesondere dann realistisch, wenn die Prüfung des Gedächtnisinhalts aus technischen Gründen schwierig oder nicht möglich ist. In diesem Fall ist es möglich, den am Ende der dynamischen Testsequenz erreichten Systemzustand beispielsweise durch Untersuchung des Systemverhaltens auf weitere Eingaben zu bestimmen, wie beispielsweise in

[Cho78] mit der charakteristischen Sequenzmenge oder in [DooFra94] mit der beobachtbaren Äquivalenz angeregt.

Die Definition der dynamischen Testsequenz erlaubt es nun, die Definitionen des dynamischen Testlaufs und des dynamischen Testverfahrens erneut zu erweitern. So kann ein dynamischer Testlauf nun anstelle von dynamischen Testfällen auf dynamische Testsequenzen einer vorgegebenen Länge bezogen werden; sein Bestehen ergibt sich aus dem Bestehen aller enthaltenen Testsequenzen. Der Begriff des dynamischen Testverfahrens kann als Zuordnung von dynamischen Testläufen bestehend aus dynamischen Testsequenzen zu Prüfling und Spezifikation verstanden werden. Dabei ist weiterhin wie in Definition 3-8 die Monotoniebedingung vorauszusetzen, die die Hinzunahme weiterer dynamischer Testsequenzen zu einem dynamischen Testlauf gestattet. Zum Bestehen eines dynamischen Testverfahrens genügt auch hier die Zuordnung eines einzigen Testlaufs, den der Prüfling bezüglich der Spezifikation besteht. Auch die weiteren, in Anschnitt 3.2 getroffenen Definitionen zum dynamischen Test können problemlos auf dynamische Testsequenzen erweitert werden. Da diese Erweiterungen ohne Schwierigkeiten aus den Definitionen aus Anschnitt 3.2 abgeleitet werden können, werden sie hier nicht einzeln aufgeführt. Es steht nun für den dynamischen Test über einem erweiterten Ein- und Ausgabebereich die gesamte formale Basis der Definitionen aus Abschnitt 3.2 in angepasster Form zur Verfügung.

4.4.2 Spezifikation zustandsbasierter Systeme

Ebenso wie bei anderen funktionsorientierten Testverfahren wird auch im Test der spezifizierten Funktionalität eines zustandsbasierten Systems die Spezifikation als Grundlage benötigt. Die in der Praxis verwendeten Beschreibungsformen von zustandsbasierten Systemen beruhen häufig auf der Idee eines endlichen Zustandsübergangsmodells, das explizit die Menge der Zustände angibt und die Modalität der Zustandsübergänge beschreibt [Cho78, Har87, Bei90]. Eine entsprechende Modellierungsform ist auch in der Unified Modeling Language UML enthalten [BooRumJac98, Oes98, OMG03]. Auf dieser Basis beruht der im Folgenden angegebene Formalismus zur Beschreibung einer zustandsendlichen Spezifikation als endliches Zustandsübergangsmodell, der als Grundlage für die Beschreibung der dynamischen Testverfahren für zustandsbasierte Systeme in Abschnitt 4.4.3 verwendet wird.

Zustandsübergangsmodelle werden in der Praxis meist graphisch oder in Form von Zustandsübergangstabellen beschrieben. Eine Einführung zu diesen Beschreibungsformen ist beispielsweise in [Lig02] zu finden, Überlegungen zur sinnvollen Gestaltung finden sich außerdem in [Bei90, Har87], Hinweise auf Werkzeugunterstützung liefert [HarEA90]. Eine den kognitiven Strukturen des Menschen gut angepasste und intuitiv erfassbare Beschreibungsform ist sehr wichtig, da die zustandsbasierte Spezifikation als Modellierung einen wichtigen Schritt im Formalisierungsprozess der Softwareentwicklung darstellt. In der vorliegenden Untersuchung der Formalisierung funktionsorientierter Testverfahren wird auf diese Beschreibungsformen jedoch nicht eingegangen, da hier lediglich eine mathematisch fassbare funktionale Beschreibung benötigt wird. Die folgende Definition beschreibt ein endliches Zustandsübergangsmodell als elementare Grundform einer zustandsendlichen Spezifikation.

Definition 4-47: Endliches Zustandsübergangsmodell

Ein Tripel (Z, E, f) bestehend aus einer endlichen Menge Z, die als Zustandsmenge bezeichnet wird, einer endlichen Menge E, die als Ereignismenge bezeichnet wird, und einer Zustandsübergangsfunktion

$$\begin{array}{lccc} f: & Z \times E & \rightarrow & Z \\ & (z, e) & \mapsto & f(z, e), \end{array}$$

die jedem Element der Zustandsmenge in Verbindung mit einem Element der Ereignismenge ein weiteres Element der Zustandsmenge zuordnet, wird im Folgenden als endliches Zustandsübergangsmodell bezeichnet. Der einem Zustands-/Ereignispaar zugeordnete Zustand f(z, e) ist als Folgezustand des ursprünglichen Zustandes z nach Auftreten des Ereignisses e zu verstehen.

Zentrale Eigenschaft eines Zustandsübergangsmodells ist, dass der Folgezustand allein durch den aktuellen Zustand und das auftretende Ereignis bestimmt ist. Die in der Definition vorausgesetzte Endlichkeit der Zustands- und Ereignismenge ist keine unbedingte Voraussetzung zur Beschreibung eines Zustandsübergangsmodells, es sind ebenso Modelle mit unendlich vielen Zuständen und Ereignissen denkbar. Für die Formulierung dynamischer Testverfahren sind solche Modelle jedoch problematisch, da mit endlich vielen dynamischen Testfällen auch elementare Vollständigkeitskriterien bezüglich des Modells nicht erfüllt werden können. Im Folgenden wird daher immer die Endlichkeit des Zustandsübergangsmodells vorausgesetzt. In der Praxis ist die Zustandsendlichkeit der Spezifikation fast immer gegeben, da zustandsbasierte Beschreibungsformen genutzt werden, um die Komplexität der Spezifikation beherrschbar zu machen und sich graphische und tabellarische Beschreibungsformen im Allgemeinen auf eine endliche Zahl von Zuständen und Ereignissen beschränken.

Die Definition eines endlichen Zustandsübergangsmodells ist an der Definition eines endlichen Automaten der Automatentheorie ausgerichtet [Sch01]. In der klassischen Automatentheorie wird zusätzlich für jedes endliche Zustandsübergangsmodell eine Markierung des Startzustandes in der Menge der Zustände und die Angabe einer Teilmenge akzeptierender Endzustände verlangt. Eine entsprechend erweiterte Definition wird als endlicher Automat bezeichnet und spielt beispielsweise bei der Definition von Grammatiken eine zentrale Rolle. Für die Beschreibung der Funktionalität oder des Verhaltens eines softwarebasierten oder technischen Systems ist die eingeschränkte obige Definition des endlichen Zustandübergangsmodells jedoch geeigneter, da ein Startzustand oft nicht eindeutig bestimmbar ist und das Konzept der akzeptierenden Endzustände keine allgemeingültige Entsprechung hat. Falls nötig, kann das endliche Zustandsübergangsmodell um die entsprechenden Angaben erweitert werden.

Wichtig für die Funktionalitäts- und Verhaltensbeschreibung eines softwarebasierten Systems ist die Modellierung des unterlagerten Zeit- und Antwortverhaltens. In einem einfachen Ansatz kann angenommen werden, dass als Systemreaktion auf ein Ereignis ein Zustandsübergang und eine zusätzliche Ausgabe des Systems spezifiziert werden. Gleichzeitig wird für den Zustandsübergang eine maximal zulässige Zeit vorgegeben. Dies kann wie folgt in der obigen Definition berücksichtigt werden.

Definition 4-48: Zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe

Ein Quadrupel (Z, E, f, A) bestehend aus einer endlichen Menge Z, die als Zustandsmenge bezeichnet wird, einer endlichen Menge E, die als Ereignismenge bezeichnet wird, einer Menge A, die die möglichen Systemausgaben enthält, und einer zeitbehafteten Zustandsübergangsfunktion

$$\begin{array}{lccl} f: & Z \times E & \rightarrow & Z \times A \times \Re^+ \\ & (z, e) & \mapsto & (f_Z(z,e), f_A(z,e), f_{\Re^+}(z,e)) , \end{array}$$

die jedem Element der Zustandsmenge in Verbindung mit einem Element der Ereignismenge ein weiteres Element der Zustandsmenge, ein Element der möglichen Systemausgaben und eine positive reelle Zahl zuordnet, wird als zeitbehaftetes endliches Zustandsübergangsmodell bezeichnet. Der einem Zustands-/Ereignispaar zugeordnete Zustand $f_Z(z, e)$ ist als Folgezustand des ursprünglichen Zustandes z nach Auftreten des Ereignisses e zu verstehen, die zugehörige Ausgabe $f_A(z, e)$ entspricht der Systemausgabe beim Zustandsübergang, und die reelle Zahl $f_{\Re^+}(z,e)$ gibt die maximal zulässige Zeit für den Zustandsübergang vor.

Die heute in der Praxis verwendeten zustandsbasierten Modelle ermöglichen zahlreiche weitere Modellierungsaspekte und eine weitaus komplexere Semantik als das hier verwendete Grundmodell. Sie gestatten, Systemaktionen in Zustandsübergängen sowie beim Erreichen und Verlassen von Zuständen zu beschreiben, lassen abzählbar unendliche Zustandsmengen zu, erlauben eine Hierarchisierung der Zustände und verfügen über komplexe Mechanismen zur Modellierung des Zeitverhaltens. Auf diese Weise liefern sie mächtige, problemangepasste Beschreibungsformen für unterschiedliche Systemtypen. Ihre wachsende Komplexität und die verschiedenartigen semantischen Interpretationsmöglichkeiten stehen allerdings einer Vereinheitlichung der Modellierungsansätze entgegen, weswegen ein allgemein gültiger Ansatz nicht zur Verfügung steht. Zur Beschreibung der dynamischen Testverfahren für zustandsbasierte Systeme wird daher das oben beschriebene zeitbehaftete endliche Zustandsübergangsmodell verwendet, das bei Bedarf an weitere zustandsbasierte Modellierungstechniken angepasst werden muss.

Um ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) als Spezifikation über einem erweiterten Ein- und Ausgabebereich entsprechend Definition 4-39 zu interpretieren, ist der zugrunde liegende erweiterte Eingabebereich $I \times H$ und der erweiterte Ausgabebereich $O \times H$ mit der kombinierten Zustands- und Ereignismenge $Z \times E$ beziehungsweise der kombinierten Menge der Zustände und Systemausgaben $Z \times A$ in Beziehung zu setzen. Diese Interpretation eines zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe als Spezifikation über einem gegebenen erweiterten Ein- und Ausgabebereich kann wie folgt beschrieben werden.

Definition 4-49: Interpretation eines zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe als Spezifikation über einem erweiterten Ein- und Ausgabebereich

Wird zu einem zeitbehafteten endlichen Zustandsmodell mit Ausgabe (Z, E, f, A) ein Eingabebereich I, ein Ausgabebereich O und ein Speicherbereich H angegeben, und werden zusätzlich die bitotalen und linkseindeutigen Relationen

$Id\_Zustand \subseteq Z \times H$ (Zustandsidentifikation)

$Id\_Ereignis \subseteq E \times I$ (Ereignisidentifikation)

$Id\_Ausgabe \subseteq A \times O$ (Ausgabeidentifikation)

angegeben, so wird dies als Interpretation des zeitbehafteten endlichen Zustandsmodells bezeichnet.

Satz:

Durch eine Interpretation eines zeitbehafteten endlichen Zustandsmodells ist mit

$$
\begin{aligned}
(i, h_1, o, h_2, r) \in s \quad \Leftrightarrow \quad & (\exists(z_1 \in Z)\, \exists(z_2 \in Z)\, \exists(e \in E)\, \exists(a \in A) \\
& ((z_1, h_1) \in \text{Id\_Zustand}) \wedge \\
& ((z_2, h_2) \in \text{Id\_Zustand}) \wedge \\
& ((e, i) \in \text{Id\_Ereignis}) \wedge \\
& ((a, o) \in \text{Id\_Ausgabe}) \wedge \\
& (z_2 = f_z(z_1, e)) \wedge (a = f_A(z_1, e)) \wedge (r \leq f_{\Re^+}(z_1, e)))
\end{aligned}
$$

eine Spezifikation über dem erweiterten Ein- und Ausgabebereich $I \times H$ beziehungsweise $O \times H$ entsprechend Definition 4-39 beschrieben.

Beweis:

Für die Definitionen der verwendeten Begriffe Bitotalität und Linkseindeutigkeit einer Relation sei zunächst auf [ReiSoe01] verwiesen. Zu zeigen ist die Vollständigkeit der angegebenen Relation über $I \times H$: Für beliebiges (i, h_1) aus $I \times H$ können wegen der Bitotalität und Linkseindeutigkeit der Relationen Id_Zustand und Id_Ereignis eindeutig ein Zustand $z_1 \in Z$ und ein Ereignis $e \in E$ identifiziert werden, für die $(z_1, h_1) \in$ Id_Zustand und $(e, i) \in$ Id_Ereignis gelten. Durch die zeitbehafteten Zustandsübergangsfunktion wird dem so identifizierten Zustand und Ereignis (z_1, e) ein Tripel $(z_2, a, r) := f(z_1, e)$ mit $z_2 \in Z$, $a \in A$ und $r \in \Re^+$ zugewiesen. Wegen der Bitotalität der Relationen Id_Ereignis und Id_Ausgabe ist nun die Existenz mindestens eines Ausgabedatums $o \in O$ und mindestens eines Speicherzustandselements $h_2 \in H$ mit $(a, o) \in$ Id_Ausgabe beziehungsweise $(z_2, h_2) \in$ Id_Zustand gesichert, so dass $(i, h_1, o, h_2, r) \in s$. ■

Am Beweis wird deutlich, auf welche Art und Weise ein zeitbehaftetes endliches Zustandsübergangsmodell als Spezifikation interpretiert werden kann. Dabei ist zu beachten, dass Ausgabedatum und zugewiesener Speicherzustand nicht eindeutig bestimmt sind, da die Relationen Id_Ereignis und Id_Ausgabe nicht als rechtseindeutig vorausgesetzt werden können. Dies schlägt sich in einer über $O \times H \times \Re^+$ mehrdeutigen relationalen Beschreibung der Spezifikation nieder. Hieraus werden die Freiheitsgrade ersichtlich, die bei der Interpretation eines zeitbehafteten endlichen Zustandsübergangsmodells als Spezifikation gegeben sind. Diese sollten nach Möglichkeit im Rahmen der Spezifikation weiter detailliert werden, beispielsweise mit zusätzlichen Beschreibungstechniken. Die vorausgesetzte Linkseindeutigkeit der Ausgabeidentifikation wird im Beweis nicht verwendet, sie unterstützt lediglich die Interpretierbarkeit des zeitbehafteten endlichen Zustandsmodells. Es kann daher auch auf die Voraussetzung verzichtet werden.

Aus der formalen Beschreibung eines endlichen Zustandsübergangsmodells wird deutlich, dass der Prozess der Formalisierung, der auf dem Weg von der ursprünglichen Idee der Anforderungen bis zur formalen Beschreibung im Prüfling beschritten werden muss, bei der Erstellung einer zustandsbasierten Spezifikation deutlich vorangetrieben wird. Da dieser Prozess durch menschliche Kreativität bestimmt wird und naturgemäß fehlerbehaftet ist, muss die Erstellung des endlichen Zustandsübergangsmodells mit besonderer Sorgfalt betrieben werden; sie ist einer gründlichen Prüfung zu unterziehen. Dies kann durch Vergleich der Interpretation des Modells mit der ursprünglichen Spezifikation beziehungsweise durch Betrachtung der Differenzen geschehen. Setzt man Fehlerfreiheit bei der Mo-

dellbildung voraus, ergeben sich aus diesem Vergleich für den Stand der Formalisierung der ursprünglichen Idee zwei Möglichkeiten. Einerseits ist denkbar, dass die Formalisierung der Anforderungen durch das endliche Zustandsübergangsmodell vollständig abgeschlossen ist. Dies ist genau dann der Fall, wenn die aus dem Modell gemäß Definition 4-49 mit größtmöglicher Allgemeinheit abgeleitete Interpretation mit der ursprünglichen, in der relationalen Spezifikation beschriebenen Idee übereinstimmt. In diesem Fall sind alle Aspekte der intendierten Funktionalität im Modell enthalten, eine automatische Ableitung des Prüflings aus dem Modell zum Beispiel auf Basis von automatischer Codegenerierung ist denkbar. Im dynamischen Test kann das Modell zur Prüfung der Bestehensrelation verwendet werden, es stellt ein optimales Testorakel im Sinne von Definition 3-13 dar. Andererseits ist es möglich, dass der Prozess der Formalisierung mit Angabe des endlichen Zustandsübergangsmodells noch nicht abgeschlossen ist, so dass Abweichungen zwischen der Spezifikation der intendierten Funktionalität und möglichen Interpretationen des Modells bestehen. Das Modell ist dann als Abstraktion zu betrachten: Zur Erstellung des Prüflings auf Basis des Zustandsübergangsmodells und für seinen dynamischen Test ist in diesem Fall zusätzliches Wissen über die tatsächliche, gedächtnisbehaftete Spezifikation notwendig. In diesem Fall hat das Modell als Testorakel nur suboptimalen Charakter, die im dynamischen Test entsprechend Definition 4-43 durchgeführte Konformitätsprüfung der Testergebnisse muss sich auf vollständige, relationale Beschreibung der Spezifikation beziehen. Der dynamische Test ermöglicht in diesem Fall neben der Prüfung der Spezifikationskonformität des Prüflings auch Rückschlüsse auf die Spezifikationskonformität des Modells.

4.4.3 Verfahren zum dynamischen Test für zustandsbasierte Systeme

Aufgrund ihrer Akzeptanz in Softwaredesign und –implementierung wurden endliche Zustandsübergangsmodelle in der Literatur bereits frühzeitig auch im Hinblick auf den dynamischen Test diskutiert [Cho78, Bei90]. Zur systematischen Prüfung der in einem endlichen Zustandsübergangsmodell beschriebenen Funktionalität wurden unterschiedliche Strategien entwickelt, die in den folgenden Verfahren formuliert werden. Elementare Forderung ist die im folgenden Verfahren geforderte Erreichung aller Zustände eines Zustandsübergangsmodells im dynamischen Test [Cho78, Bei90, Lig02].

Verfahren 4-28: Zustandsüberdeckung eines zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe

(a) Voraussetzung:

Gegeben sei eine gedächtnisbehaftete Spezifikation $s \in \Sigma_{I, O, H}$, die durch ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) und eine Interpretation über dem Eingabebereich I, dem Ausgabebereich O und dem Speicherbereich H mit den Relationen

$Id\_Zustand \subseteq Z \times H$ (Zustandsidentifikation)
$Id\_Ereignis \subseteq E \times I$ (Ereignisidentifikation)
$Id\_Ausgabe \subseteq A \times O$ (Ausgabeidentifikation)

beschrieben ist.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Zustandsüberdeckung des zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe wird beschrieben durch

$$X^{Z}: \quad \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I \times H))$$
$$s \mapsto \{ T \in \wp^{fin}(I \times H): P_s(T) \}$$

mit

$$P_s(T) := \forall(z \in Z)\, \exists(z_1 \in Z)\, \exists(e_1 \in E)\, \exists((i, h) \in T)$$
$$((f_Z(z_1, e_1) = z) \wedge ((z_1, h) \in Id\_Zustand) \wedge ((e_1, i) \in Id\_Ereignis)).$$

Eine notwendige Bedingung für die in Definition 3-9 beschriebene Anwendbarkeit der Zustandsüberdeckung ist, dass jeder Zustand im Zustandsübergangsmodell erreicht werden kann. Durch das Prädikat P_s wird dann erreicht, dass zu jedem Zustand mindestens ein dynamischer Testfall in den dynamischen Testläufen enthalten ist, der entsprechend der Spezifikation den Prüfling in den geforderten Zustand überführen soll. Werden die dynamischen Testfälle eines solchen dynamischen Testlaufs bestanden, so ist sichergestellt, dass jeder Zustand mindestens einmal erreicht wurde. Alternativ oder zusätzlich zum Prädikat P_s kann auch das Prädikat

$$\tilde{P}_s(T) := \forall(z \in Z)\, \exists((i, h) \in T)\, ((z, h) \in Id\_Zustand)$$

verwendet werden, das sicherstellt, dass der Prüfling in jedem Zustand mindestens mit einem Testfall ausgeführt wurde. Diese erfordert nicht die Erreichung, sondern die Nutzung des Prüflings in jedem Zustand. Da jedoch in den meisten Darstellungen die Erreichung aller Zustände gefordert ist [Bei90, Lig02], wird in der obigen Verfahrensdefinition das Prädikat P_s verwendet.

Eine weiterführende Forderung ist der dynamische Test aller Zustandsübergänge im Modell [Bei90, Lig02]. Das Verfahren kann wie folgt formuliert werden.

Verfahren 4-29: Überdeckung der Zustandsübergänge in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe

(a) Voraussetzung:

Gegeben sei eine gedächtnisbehaftete Spezifikation $s \in \Sigma_{I, O, H}$, die durch ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) und eine Interpretation über dem Eingabebereich I, dem Ausgabebereich O und dem Speicherbereich H mit den Relationen

$Id\_Zustand \subseteq Z \times H$ (Zustandsidentifikation)
$Id\_Ereignis \subseteq E \times I$ (Ereignisidentifikation)
$Id\_Ausgabe \subseteq A \times O$ (Ausgabeidentifikation)

beschrieben ist.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung der Zustandsübergänge in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe wird beschrieben durch

$$X^{Z\ddot{U}}: \quad \Sigma_{I,O} \rightarrow \wp(\wp^{fin}(I \times H))$$
$$s \mapsto \{ T \in \wp^{fin}(I \times H): P_s(T) \}$$

mit

$$
\begin{aligned}
P_s(T) \quad := \quad & \forall(z_1 \in Z)\ \forall(z_2 \in Z) \\
& ((\exists(e_1 \in E)\ (f_z(z_1, e_1) = z_2)) \Rightarrow \\
& (\exists((i, h) \in T)\ \exists(e_2 \in E) \\
& ((f_z(z_1, e_2) = z_2) \wedge ((e_2, i) \in \text{Id\_Ereignis}) \wedge ((z_1, h) \in \text{Id\_Zustand})))).
\end{aligned}
$$

Durch das Prädikat P_s wird sichergestellt, dass zu jedem im Zustandsübergangsmodell spezifizierten Zustandswechsel mindestens ein Testfall existiert, der im Falle seines Bestehens diesen Zustandswechsel im Prüfling bewirkt. Kann ein Zustandswechsel durch mehrere Ereignisse ausgelöst werden, so genügt der dynamische Test eines beliebigen Ereignisses.

Soll darüber hinaus sichergestellt werden, dass der jeweilige Zustandswechsel durch jedes für ihn spezifizierte Ereignis im dynamischen Test ausgelöst wird, so kann das folgende, beispielsweise in [Lig02] beschriebene Verfahren verwendet werden.

Verfahren 4-30: Überdeckung aller Ereignisse zu jedem Zustandsübergang in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe

(a) Voraussetzung:

Gegeben sei eine gedächtnisbehaftete Spezifikation $s \in \Sigma_{I, O, H}$, die durch ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) und eine Interpretation über dem Eingabebereich I, dem Ausgabebereich O und dem Speicherbereich H mit den Relationen

$\text{Id\_Zustand} \subseteq Z \times H$ (Zustandsidentifikation)
$\text{Id\_Ereignis} \subseteq E \times I$ (Ereignisidentifikation)
$\text{Id\_Ausgabe} \subseteq A \times O$ (Ausgabeidentifikation)

beschrieben ist.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller Ereignisse zu jedem Zustandsübergang in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe wird beschrieben durch

$$
\begin{aligned}
X^{ZEÜ}: \quad \Sigma_{I, O} \quad & \rightarrow \quad \wp(\wp^{fin}(I \times H)) \\
s \quad & \mapsto \quad \{ T \in \wp^{fin}(I \times H) : P_s(T) \}
\end{aligned}
$$

mit

$$
\begin{aligned}
P_s(T) \quad := \quad & \forall(z_1 \in Z)\ \forall(z_2 \in Z)\ \forall(e \in E) \\
& ((f_z(z_1, e) = z_2) \Rightarrow \\
& (\exists((i, h) \in T)\ (((z_1, h) \in \text{Id\_Zustand}) \wedge ((e, i) \in \text{Id\_Ereignis})))).
\end{aligned}
$$

Verfahren 4-29 und Verfahren 4-30 zielen auf die im Zustandsübergangsmodell beschriebenen Zustandsübergänge und die sie auslösenden Ereignisse ab. Ereignisse, die laut Spezifikation in einem Zustand keinen Zustandswechsel auslösen, sondern vom Prüfling zu ignorieren sind, werden nicht berücksichtigt. Solche Ereignisse stellen jedoch eine Funktionalität dar, die ebenfalls in Implementierung und Test zu berücksichtigen ist. Für eine umfassende Prüfung kann daher ein dynamischer Test aller Ereignisse in jedem Zustand verlangt werden. Dies fordert das folgende Verfahren.

Verfahren 4-31: Überdeckung aller Ereignisse in allen Zuständen in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe

(a) Voraussetzung:

Gegeben sei eine gedächtnisbehaftete Spezifikation $s \in \Sigma_{I, O, H}$, die durch ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) und eine Interpretation über dem Eingabebereich I, dem Ausgabebereich O und dem Speicherbereich H mit den Relationen

$Id\_Zustand \subseteq Z \times H$ (Zustandsidentifikation)
$Id\_Ereignis \subseteq E \times I$ (Ereignisidentifikation)
$Id\_Ausgabe \subseteq A \times O$ (Ausgabeidentifikation)

beschrieben ist.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller Ereignisse in allen Zuständen in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe wird beschrieben durch

$$X^{ZE}: \quad \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I \times H))$$
$$s \mapsto \{ T \in \wp^{fin}(I \times H): P_s(T) \}$$

mit

$$P_s(T) := \forall(z \in Z)\, \forall(e \in E)$$
$$\exists((i, h) \in T)\, (((z, h) \in Id\_Zustand) \wedge ((e, i) \in Id\_Ereignis))).$$

Die obigen Verfahren stellen eine Bandbreite von Kriterien bereit, die eine Auswahl von Daten für den dynamischen Test orientiert an den Strukturelementen eines endlichen Zustandsübergangsmodells und eine Prüfung ihrer Vollständigkeit ermöglichen. Je nach Zahl der Zustände, Ereignisse und Zustandsübergänge im Zustandsübergangsmodell wird der Umfang der dynamischen Testläufe zur Durchführung der Verfahren unterschiedlich ausfallen. Während die Überdeckung aller Zustände im Allgemeinen als nicht hinreichendes Kriterium für den dynamischen Test anzusehen ist, kann die Prüfung aller Zustandsübergänge ist in einem umfangreichen Zustandsübergangsmodell zeitaufwändig und schwierig sein. Insbesondere die Prüfung aller Ereignisse in allen Zuständen kann problematisch sein, da es aus technischen Gründen schwierig sein kann, alle Ereignisse in allen Systemzuständen auszulösen. Diese Prüfung ist jedoch für eine effektive Qualitätssicherung sehr wichtig, da hier mögliche Unvollständigkeiten bei der Gestaltung des Zustandsübergangsmodells oder bei der Implementierung des Prüflings entdeckt werden können.

Um solche Unvollständigkeiten schon bei der Gestaltung des Zustandsübergansmodells zu verhindern, schlägt Liggesmeyer die Prüfung der Vollständigkeit mit Hilfe einer tabellarischen Darstellung vor, die eine Beschreibung der Systemreaktion für alle Ereignisse zu allen Zuständen fordert [Lig02]. Weiterhin schlägt er die Einführung eines weiteren Zustands vor, der ein Fehlverhalten kennzeichnet und beim Auftreten von in einem Zustand unvorhergesehenen Ereignissen eingenommen wird. Bei einem solchen Zustandsübergang in den fehlerbehafteten Zustand können Fehlerbehandlungen spezifiziert werden, der Zustand selbst sollte als sicher gestaltet sein. Die Spezifikation der Zustandsübergänge für unvorhergesehene Ereignisse in einen Fehlerzustand ist Grundlage für die Fehlertoleranz eines Systems und entscheidend für die Systemsicherheit und –zuverlässigkeit; sie ist daher wichtiger Bestandteil einer defensiven Programmierung. Im dynamischen Test stellen die Zustands-

übergänge in Fehlerzustände durch unvorgesehene Ereignisse ein Problem dar, da die entsprechenden dynamischen Testfälle aus technischen Gründen oft nicht oder nur schwer durchführbar sind. Wird eine Klassifikation der Zustände nach Gültigkeit oder Fehlerbehaftung im Modell mit angegeben, so kann dieser Sachverhalt im dynamischen Test berücksichtigt werden.

Definition 4-50: Kennzeichnung fehlerbehafteter Zustände in einem zeitbehafteten endliches Zustandsübergangsmodell mit Ausgabe

Fehlerbehaftete Zustände werden in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) durch Angabe der Teilmenge der fehlerhaften Zustände

$Z\_fehlerhaft \subseteq Z$

gekennzeichnet.

In der Praxis kann es zusätzlich sinnvoll sein, auf eine geringe Anzahl fehlerbehafteter Zustände zu achten. So kann beispielsweise ein fehlerbehafteter Zustand für ein Zustandsübergangsmodell ausreichend sein.

Liegt eine Kennzeichnung der fehlerbehafteten Zustände zu einem Zustandsübergangsmodell vor, so können reduzierte, praktikable dynamische Testverfahren angegeben werden. So kann der in [Lig02] vorgeschlagene dynamische Test aller Zustandsübergänge, die nicht in einen fehlerbehafteten Zustand führen, wie folgt notiert werden.

Verfahren 4-32: Überdeckung der Zustandsübergänge in nicht fehlerbehaftete Zustände in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe

(a) Voraussetzung:

Gegeben sei eine gedächtnisbehaftete Spezifikation $s \in \Sigma_{I, O, H}$, die durch ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) und eine Interpretation über dem Eingabebereich I, dem Ausgabebereich O und dem Speicherbereich H mit den Relationen

$Id\_Zustand \subseteq Z \times H$ (Zustandsidentifikation)
$Id\_Ereignis \subseteq E \times I$ (Ereignisidentifikation)
$Id\_Ausgabe \subseteq A \times O$ (Ausgabeidentifikation)

beschrieben ist. Zusätzlich seien fehlerbehaftete Zustände $Z\_fehlerhaft \subseteq Z$ im Zustandsübergangsmodell gekennzeichnet.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung der Zustandsübergänge in nicht fehlerhafte Zustände in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe wird beschrieben durch

$$X^{ZÜnf}: \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I \times H))$$
$$s \mapsto \{ T \in \wp^{fin}(I \times H): P_s(T) \}$$

mit

$$P_s(T) := \forall(z_1 \in Z)\ \forall(z_2 \in Z \setminus Z\_fehlerhaft)$$
$$((\exists(e_1 \in E)\ (f_z(z_1, e_1) = z_2)) \Rightarrow$$
$$(\exists((i, h) \in T)\ \exists(e_2 \in E)$$
$$((f_z(z_1, e_2) = z_2) \wedge ((e_2, i) \in Id\_Ereignis) \wedge ((z_1, h) \in Id\_Zustand)))).$$

Entsprechend können auch die Verfahren 4-30 und Verfahren 4-31 auf Zustandsübergänge in nicht fehlerbehaftete Zustände beschränkt werden. Die resultierenden dynamischen Testverfahren stellen praktikable Varianten der ursprünglichen Verfahren dar und können als Minimalkriterien verwendet werden, wenn die ursprünglichen Verfahren aus technischen Gründen nicht erfüllbar sind.

In [Bei90] wird darauf hingewiesen, dass ein mögliches Fehlverhalten eines gedächtnisbehafteten Prüflings oft erst nach mehreren Zustandswechseln zu beobachten ist. Dies legt die Forderung nahe, zum dynamischen Test der in einem endlichen Zustandsübergangsmodell spezifizierten Funktionalität die Prüfung aller möglichen Pfade einer vorgegebenen Länge durch den Zustandsgraphen zu fordern. Entsprechende Testverfahren werden bereits in [Cho78] diskutiert. Chow formuliert diese Testverfahren für endliche Zustandsübergangsmodelle in Anlehnung an die in [How76] und [PimRau76] angegebenen Strategien zur Überdeckung von Kontrollflussgraphen und bezeichnet sie als „n-switch coverage“. Sie können mit Hilfe der in Definition 4-46 eingeführten dynamischen Testsequenzen wie folgt beschrieben werden.

Verfahren 4-33: Überdeckung aller Pfade der Länge n ($n \in \aleph \setminus \{ 0 \}$) in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe

(a) Voraussetzung:

Gegeben sei eine gedächtnisbehaftete Spezifikation $s \in \Sigma_{I, O, H}$, die durch ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) und eine Interpretation über dem Eingabebereich I, dem Ausgabebereich O und dem Speicherbereich H mit den Relationen

$Id\_Zustand \subseteq Z \times H$ (Zustandsidentifikation)
$Id\_Ereignis \subseteq E \times I$ (Ereignisidentifikation)
$Id\_Ausgabe \subseteq A \times O$ (Ausgabeidentifikation)

beschrieben ist.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Überdeckung aller Pfade der Länge n ($n \in \aleph \setminus \{ 0 \}$) in einem zeitbehafteten endlichen Zustandsübergangsmodell mit Ausgabe wird beschrieben durch

$$X^{n\text{-}1\text{-}switch}: \quad \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I^{n-1} \times H))$$
$$s \mapsto \{ T \in \wp^{fin}(I^{n-1} \times H): P_s(T) \}$$

mit

$$
\begin{aligned}
P_s(T) \quad := \quad & \forall((z_1, z_2, \ldots, z_n) \in Z^n) \\
& ((\forall(k \in \{ 1, \ldots, n\text{-}1 \}) \exists(e \in E) (f_z(z_k, e) = z_{k+1})) \Rightarrow \\
& (\exists((i_1, i_2, \ldots, i_{n-1}, h) \in T) \\
& (((z_1, h) \in \text{Id\_Zustand}) \wedge \\
& (\exists(e_1 \in E) (((e_1, i_1) \in \text{Id\_Ereignis}) \wedge (f_z(z_1, e_1)=z_2))) \wedge \\
& (\exists(e_2 \in E) (((e_2, i_2) \in \text{Id\_Ereignis}) \wedge (f_z(z_2, e_2)=z_3))) \wedge \\
& \ldots \wedge \\
& (\exists(e_{n-1} \in E) (((e_{n-1}, i_{n-1}) \in \text{Id\_Ereignis}) \wedge (f_z(z_{n-1}, e_{n-1})=z_n))))))).
\end{aligned}
$$

Die in der graphischen Beschreibung eines Zustandsübergangsmodells intuitiv nachvollziehbare Forderung des Verfahrens, dass zu jeder im Modell möglichen Zustandsfolge der vorgegebenen Länge eine dynamische Testsequenz existiert, die diese Zustandsfolge prüft, findet hier Eingang in das komplexe logische Prädikat P_s. In diesem wird für alle Permutationen von Zuständen geprüft, ob es Ereignisse gibt, die von einem Zustand in den Folgezustand führen. Ist dies der Fall, so wird durch die Zustandsfolge ein möglicher Pfad durch das Zustandsübergangsmodell beschreiben, der im Rahmen des dynamischen Tests durch eine Testsequenz geprüft werden muss.

Chow bezieht sich in [Cho78] auf den Test von Prüflingen, die durch ein endliches Zustandsübergangsmodell mit Ausgabe und Kennzeichnung des Anfangszustands vollständig beschrieben sind. Weiterhin setzt er für das Zustandsübergangsmodell die Erreichbarkeit aller Zustände und die Minimalität des beschriebenen Automaten [Sch01] voraus. Mit diesen Voraussetzungen weist er nach, dass die Überdeckung eines endlichen Zustandsübergangsmodells mit n Zuständen gemäß „n-1-switch coverage“ ausreicht, um fehlerhafte Ausgaben, fehlerhafte Zustandsübergänge und zusätzliche Zustände im Prüfling gegenüber einer Spezifikation nachzuweisen. Chow schlägt auf Basis von automatentheoretischen Überlegungen ein weiteres Verfahren vor, das die „n-1-switch coverage“ subsumiert und das unter seinen Annahmen zusätzlich in der Lage ist, fehlende Zustände zu erkennen. Es erfordert für alle Zustandsübergänge die Generierung von dynamischen Testsequenzen, die, im Startzustand beginnend, den Zustandsübergang auslösen. An diese Sequenzen werden weitere Sequenzen angefügt, die aus einer Ereigniskombination und einer abschließenden Prüfsequenz bestehen. Die Ereigniskombinationen müssen alle möglichen Kombinationen von Ereignissen umfassen, deren Länge höchstens der maximalen Anzahl der möglicherweise fehlenden Zustände entspricht. Die Prüfsequenzen entstammen einer „charakteristischen Sequenzmenge“, durch deren Anwendung alle Zustände eindeutig voneinander unterschieden werden können. Die Existenz dieser charakteristischen Sequenzmenge ist unter den Annahmen von Chow, insbesondere der Minimalität des Automaten, sichergestellt.

Chows Verfahren zeigt eine sehr umfassende Strategie, mit der fehlerhafte Ausgaben, fehlerhafte Zustandsübergänge, zusätzliche Zustände und fehlende Zustände im Prüfling nachgewiesen werden können. Die grundsätzliche Strategie des Verfahrens kann auf den funktionsorientierten Test eines durch ein Zustandsübergangsmodell spezifizierten Prüflings übertragen werden: Hier kann entsprechend Chows Ideen der dynamische Test aller Ereigniskombinationen einer vorgegebenen Länge in jedem Zustand beziehungsweise im Anschluss an jeden Zustandsübergang gefordert werden. Grundvoraussetzung zur Beschreibung dieser Forderung als Verfahren ist die durch Definition 4-47 sichergestellte Endlichkeit der Ereignismenge. Da hier kein fester Startzustand vorausgesetzt wird, werden für die Erreichung des Zustands, an den sich die Ereigniskombinationen anschließen, keine weiteren Forderungen aufgestellt. Im Anschluss an die Ausführung der Ereigniskombinationen ist der erreichte

Zustand eindeutig festzustellen. Dies wird bei Chow durch die charakteristische Sequenzmenge erreicht, deren Existenz hier jedoch nicht vorausgesetzt werden kann. Der erreichte Zustand wird hier durch die Prüfung auf Konformität mit der Spezifikation geprüft.

Verfahren 4-34: Prüfung aller Ereignisfolgen mit Länge n ($n \in \aleph \setminus \{ 0 \}$) in jedem Zustand eines zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe

Im Folgenden bezeichnet E^k ($k \in \aleph \setminus \{ 0 \}$) das k-fache kartesische Produkt der Ereignismenge E.

(a) Voraussetzung:

Gegeben sei eine gedächtnisbehaftete Spezifikation $s \in \Sigma_{I, O, H}$, die durch ein zeitbehaftetes endliches Zustandsübergangsmodell mit Ausgabe (Z, E, f, A) und eine Interpretation über dem Eingabebereich I, dem Ausgabebereich O und dem Speicherbereich H mit den Relationen

$Id\_Zustand \subseteq Z \times H$ (Zustandsidentifikation)
$Id\_Ereignis \subseteq E \times I$ (Ereignisidentifikation)
$Id\_Ausgabe \subseteq A \times O$ (Ausgabeidentifikation)

beschrieben ist.

(b) Verfahrensbeschreibung:

Der dynamische Test zur Prüfung aller Ereignisfolgen mit Länge n ($n \in \aleph \setminus \{ 0 \}$) in jedem Zustand eines zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe wird beschrieben durch

$$X^{n\text{-e-kombi}}: \quad \Sigma_{I, O} \rightarrow \wp(\wp^{fin}(I^n \times H))$$
$$s \mapsto \{ T \in \wp^{fin}(I^n \times H): P_s(T) \}$$

mit

$$P_s(T) := \forall(z \in Z)\ \forall((e_1, e_2, \ldots, e_n) \in E^n)\ (\exists((i_1, i_2, \ldots, i_n, h) \in T)\ (((z, h) \in Id\_Zustand) \wedge (\forall(l \in \{ 1, \ldots, n \})\ ((e_l, i_l) \in Id\_Ereignis)))).$$

Das resultierende Verfahren konfrontiert den Prüfling in jedem Zustand mit allen Ereignisfolgen der vorgegebenen Länge. Es ist ein sehr umfangreiches Verfahren, das die Reaktionen des Prüflings auf mögliche Ereignisfolgen gründlich überprüft. Eine zusätzliche Prüfung von kürzeren Ereignisfolgen wie in [Cho78] ist nicht notwendig, da alle kürzeren Ereignisfolgen als Bestandteile der längeren in den dynamischen Testsequenzen enthalten sind. Durch die Prüfung des Bestehens der dynamischen Testsequenzen nach Definition 4-46 (c) ist sichergestellt, dass auch für Teilsequenzen ein spezifikationskonformes Verhalten des Prüflings sichergestellt wurde. Wird eine weniger strenge Prüfung des Bestehens der dynamischen Testsequenzen verwendet, kann eine zusätzliche Aufnahme der kürzeren Ereignisfolgen und damit eine Erweiterung des obigen Verfahrens sinnvoll sein. Dies ist beispielsweise der Fall in [Cho78], wo die während der dynamischen Testsequenz erreichten Zustände als nicht beobachtbar angenommen werden und der finale Zustand durch Prüfung der charakteristischen Sequenzmenge ermittelt wird.

Auch wenn das hier beschriebene Verfahren eine sehr umfangreiche Prüfung garantiert, können die theoretischen Ergebnisse von Chow bezüglich der garantierten Fehlerfindung nicht übertragen wer-

den. Dies ist darin begründet, dass die von Chow getroffenen Annahmen für allgemeine Prüflinge und Spezifikationen über einem erweiterten Ein- und Ausgabebereich nicht vorausgesetzt werden können.

Die in diesem Abschnitt vorgestellten Verfahren zielen darauf ab, die Auswahl der dynamischen Testfälle und die Bewertung ihrer Vollständigkeit im Hinblick auf die Elemente der formalen Beschreibung einer Spezifikation zu überprüfen, die durch ein erweitertes endliches Zustandsübergangsmodell mit Ausgabe beschrieben ist. Typischerweise wird das erweiterte endliche Zustandsübergangsmodell mit Ausgabe bereits in der Spezifikation angegeben, so dass der dynamische Test direkt auf dieser weitgehend formalen Darstellungsform aufsetzen kann. In diesem Fall ist die kreative Freiheit des Testers bei der Anwendung der Verfahren auf die Auswahl der dynamischen Testfälle zur Überdeckung der Strukturelemente beschränkt. Diese Auswahl muss der in Definition 4-49 beschriebenen Interpretation Rechnung tragen, so dass die Wahlmöglichkeiten durch den Abstraktionsgrad des Zustandsübergangsmodells bestimmt werden. Bei der Wahl der Testdaten können neben dem zustandsorientierten Vorgehen auch weitere funktionsorientierte Ansätze verwendet werden, wie im folgenden Absatz 4.4.4 angeregt werden wird. Eine Auswertung des dynamischen Tests vor dem Hintergrund der durch die Spezifikation beschriebenen Relation ermöglicht dann neben einer Qualitätssicherung des Prüflings auch eine Prüfung des Modells.

Es ist jedoch auch denkbar, dass in der Spezifikation kein Zustandsübergansmodell angegeben wurde. Falls sie sich dennoch für die Beschreibung in einem endlichen Zustandsübergansmodell eignet, kann eine zustandsbasierte Beschreibung im Design oder nachträglich im Rahmen des Testfallentwurfs erstellt werden. Dies ist dann der Fall, wenn funktional äquivalente Partitionen des Gedächtnisses und der Eingabedaten eine Bildung von Zuständen und Ereignissen erlauben, und die Systemreaktion zu den Zustands-/Ereignispaaren allgemein beschrieben werden kann. Die Erstellung des Zustandsübergangsmodells auf Basis des Spezifikationstextes muss in diesem Fall vollständig durch den Tester geleistet werden. Hierbei ist zunächst eine funktionale Partitionierung des Speicherbereichs als Grundlage der Zustandsbeschreibung entsprechend Definition 4-40 zu erstellen, auf der aufbauend das Zustandsübergangsmodell nach Definition 4-47 oder Definition 4-48 beschrieben werden muss. Auch eine funktionale Partitionierung des Eingabebereiches kann im Rahmen der Ereignisbeschreibung hilfreich sein. Implizit wird dadurch die Interpretation nach Definition 4-49 festgelegt. Das so erstellte Zustandsübergangsmodell kann dann als Grundlage des dynamischen Tests verwendet werden. Umfang und Gründlichkeit des dynamischen Tests hängen in diesem Fall stark von der intellektuellen Leistung des Testers bei der Modellierung ab. Bei einer solchen nachträglichen Modellierung muss im dynamischen Test insbesondere beachtet werden, dass die Realisierung im Prüfling als nicht-zustandsbasiert angenommen werden muss. Die zustandsbasierte Beschreibung ist in diesem Fall lediglich als Unterstützung für die Generierung und die Vollständigkeitsbewertung der Testdaten anzusehen, als Basis für die Auswertung der dynamischen Tests ist die relationale Beschreibung in der Spezifikation zu verwenden.

Können das endliche Zustandsübergangsmodell und seine Interpretation als gegeben vorausgesetzt werden, sind die hier diskutierten Verfahren vollständig beschrieben. Liegen beide Beschreibungen in maschineninterpretierbarer Form vor, so ist eine weitgehende Werkzeugunterstützung für die Überdeckungsmessung und Testdatengenerierung möglich. Das spezifizierte Zustandsübergangsmodell kann in diesen Fällen auch als Testorakel verwendet werden. Hierbei ist jedoch zu beachten, dass die Gestaltung der Relationen in Definition 4-49 Interpretationsspielräume lässt, so dass das Orakel als suboptimal zu betrachten ist und nicht die Prüfung mit der Spezifikation entsprechend Definition 4-39 ersetzen kann.

4.4.4 Kombination zustandsbasierter und nicht zustandsbasierter dynamischer Testverfahren

Durch die in Abschnitt 4.4.3 eingeführten Verfahren werden die funktionalen Aspekte einer zustandsbasierten Spezifikation systematisch geprüft. Es ist darüber hinaus möglich, weitere funktionale Aspekte der Spezifikation mit anderen funktionsorientierten Testverfahren zu prüfen. Diese können direkt im dynamischen Test des Prüflings eingesetzt werden, zur Prüfung eines gegebenen endlichen Zustandsübergangsmodells verwendet werden oder die Ableitung eines endlichen Zustandsübergangsmodells auf Basis einer informal beschrieben Spezifikation unterstützen.

So ist es beispielsweise denkbar, dass in einem zustandsbasierten Modell nur das übergeordnete Systemverhalten im Hinblick auf Zustandsübergänge formuliert ist, die tatsächlichen Systemreaktionen aber durch zusätzliche funktionale Aspekte bestimmt sind. Beispielsweise wird für ein Eingabedatum $(i, h) \in I \times H$ im Allgemeinen eine exakte Zuordnung einer bestimmten Ausgabe und eines bestimmten Speicherelements des im Zustandsübergangsmodell beschriebenen Folgezustands gefordert sein. In diesem Fall kann im dynamischen Test zusätzlich zur Prüfung auf Basis des endlichen Zustandsübergangsmodells für einzelne Zustände $H_0 \subseteq H$ eine Teilspezifikation betrachtet werden, die sich aus der Einschränkung der Spezifikation auf die Menge $I \times H_0 \times O \times H \times \Re^+$ ergibt. Diese kann mit weiteren, geeignet zu wählenden funktionsorientierten Techniken geprüft werden.

Bei der gemeinsamen Anwendung verschiedener funktionsorientierter Verfahren ergeben sich oft sinnvolle Kombinationsmöglichkeiten, die an der hier verwendeten einfachen, relationalen Notation deutlich werden. Für eine Kombination mit den zustandsbasierten Techniken bieten sich insbesondere die Verfahren der funktionalen Partitionierung oder Äquivalenzklassenbildung des Abschnitts 4.1 oder auch die Verfahren zur systematischen Analyse von Ursachen und ihren Wirkungen aus Abschnitt 4.2 an. Dies wird in den folgenden Abschnitten 4.4.4.1 und 4.4.4.2 angeregt. Eine kurze Überlegung zur Kombination der funktionsorientierten, zustandsbasierten Verfahren mit einer strukturellen Betrachtung des Prüflings wird in Abschnitt 04.4.4.3 dargestellt.

4.4.4.1 Kombination mit Verfahren der funktionalen Partitionierung

Zur Ableitung oder Prüfung eines zustandsbasierten Systems auf Basis einer informal beschriebenen Spezifikation bieten sich die Verfahren der funktionalen Partitionierung an. So wird zur Ableitung oder Prüfung der im Modell verwendeten Zustandsmenge eine Zustandsbildung über dem Speicherbereich entsprechend Definition 4-40 vorgenommen, die einer funktionalen Partitionierung nach Definition 4-4 entspricht und zusätzlich disjunkt ist.

Falls der Speicherbereich durch mehrere Parameterbereiche aufgespannt ist, müssen die für die Zustandsbildung relevanten Parameter betrachtet werden. Hier sollte geprüft werden, ob eine funktionale Äquivalenzklassenbildung sinnvoll ist: Zustände können dann durch Kombination der funktionalen Äquivalenzklassen untereinander beschrieben werden. Bei der Bestimmung der Gültigkeit oder Fehlerbehaftung einzelner Zustände ist die Gültigkeit einzelner Parameterklassen ausschlaggebend, wobei ein ungültiger Parameter die Ungültigkeit des Zustands bedingt. Funktional äquivalente Zustände, die aus einer solchen Kombination entstehen, können zusammengefasst werden. Dies können beispielsweise Zustände sein, die einen oder mehrere ungültige Parameter enthalten und zu einem einzigen ungültigen Zustand zusammengefasst werden.

Die so abgeleiteten Zustände über dem Speicherbereich können nun in ein Zustandsübergangsmodell übernommen oder mit den Zuständen eines bestehenden Modells verglichen werden. Ein entspre-

chendes Vorgehen zur Zustandsbestimmung wird auch in [Bei90] angeregt. Die Anwendung des funktionalen Partitionentests nach Verfahren 4-1 im dynamischen Test impliziert nun eine Zustandsüberdeckung entsprechend Verfahren 4-28.

Ebenso wie auf den Speicherbereich können die Verfahren der funktionalen Partitionierung auch auf den Eingabebereich angewendet werden. So kann insbesondere die Ereignisbeschreibung unterstützt oder überprüft werden. Im dynamischen Test sind dann Verfahren 4-30, Verfahren 4-31 oder Verfahren 4-32 mit kombinierter Zustands-/Ereignisüberdeckung anzuwenden.

Eine Anwendung der funktionalen Äquivalenzklassenanalyse im dynamischen Test eines zustandsbehafteten Prüflings kann darüber hinaus auch dann sinnvoll sein, wenn Ereignisse parametriert sind. Dies ist beispielsweise der Fall ist für Klassen der objektorientierten Programmierung, deren interner Zustand durch Attribute festgelegt wird und deren Methoden über komplexe Schnittstellen verfügen. Hier kann neben einer übergeordneten, zustandsbasierten Verhaltensmodellierung eine Spezifikation gegeben sein, die auf Basis von Parameterkombinationen die zu erzeugenden Ausgabe- oder Speicherdaten zuordnet. Im Bezug auf die Parameter der Schnittstellen kann eine funktionale Äquivalenzklassenanalyse notwendig sein, die gegebenenfalls für verschiedene Systemzustände separat durchzuführen ist.

Die Betrachtung der zustandsbasierten Testtechniken aus dem Blickwinkel der funktionalen Partitionierung legt eine ergänzende Grenzwertanalyse nahe. Diese bringt neue Aspekte für den Test zustandsbasierter Systeme: So kann auch im zustandsbasierten Test eine erhöhte Fehlererwartung insbesondere in Grenzbereichen der Zuordnung von Eingabedaten zu Ereignissen, der Zuordnung von Speicherinhalten zu Zuständen sowie beim Zustandsübergang und bei der Systemausgabe gerechtfertigt sein. Notwendig für die Grenzwertanalyse ist die Einführung eines Grenzwertbegriffs beispielsweise durch Wahl einer Topologie über dem Raum der Eingabe-, Ausgabe und Speicherparameter. Mit ihrer Hilfe können diejenigen Repräsentanten eines Zustandes identifiziert werden, die im Grenzbereich liegen und somit zu anderen Zuständen benachbart sind. Ebenso werden Eingabedaten in einem möglichen Grenzbereich des Ereignisses und Ausgabedaten im Grenzbereich einer spezifizierten Systemausgabe bestimmt. Der dynamische Test nach den Prinzipien der Grenzwertanalyse fordert dann die Verwendung beziehungsweise Erreichung dieser Grenzwerte. Ein solches Vorgehen kann im Test technischer Systeme mit analogen, beispielsweise mechanischen Komponenten sinnvoll sein, da hiermit die saubere Erkennung und Erreichung der Systemzustände sowie die Umsetzung der geforderten Toleranzen geprüft wird. Aber auch für reine Softwaresysteme kann es sinnvoll sein, sich über den Zustandsraum, seine Topologie, mögliche Nachbarschaften, Verwechselungen und Mehrdeutigkeiten Gedanken zu machen und daraus zusätzliche Testfälle abzuleiten.

4.4.4.2 Kombination mit Verfahren der Ursache-Wirkungs-Analyse

In [Lig02] wird bei der Einführung der Verfahren zur Ursache-Wirkungs-Analyse darauf hingewiesen, dass eine Wirkung auch eine Systemtransformation bedeuten kann. Dies zeigt, dass die Beschreibung von Ursachen und ihren Wirkungen auch bei der Spezifikation zustandsbasierter Verfahren verwendet werden. Eine solche Darstellung ergibt sich beispielsweise dann, wenn das Verhalten eines zustandsbasierten Systems als Fließtext beschrieben ist, so dass Zustandsübergänge und Systemausgaben immer als Wirkungen bestimmter Ursachenkombinationen dargestellt werden.

Bei Vorliegen einer solchen Beschreibung kann eine systematische Analyse der beschriebenen Ursache-Wirkungs-Zusammenhänge bei der Erstellung eines endlichen Zustandsübergangsmodells hilfreich sein. So beschreiben Ursachen Eigenschaften des Speicherbereichs und der Eingabedaten,

Wirkungen entsprechen den resultierenden Zustandsübergängen, eventuell gepaart mit Systemausgaben. Ausgehend von den Wirkungen können Ursachenkombinationen nun zur Beschreibung von Zuständen und Ereignissen verwendet werden, deren Aufeinandertreffen den der Wirkung entsprechenden Zustandswechsel auslöst. Hieraus resultiert eine prädikative Beschreibung von Zuständen und Ereignissen als funktionale Partitionen über dem Speicher- beziehungsweise Eingabebereich sowie von den zugehörigen Zustandsübergängen. Ist bereits ein Zustandsübergangsmodell angegeben, unterstützt die prädikative Beschreibung aus der Ursache-Wirkungs-Analyse des Fließtextes eine Vollständigkeitsprüfung der funktionalen Partitionierung über dem Speicher- und Eingabebereich und ermöglicht so die Identifikation von nicht erfassten Zuständen und Ereignissen. Sind Zustände und Ereignisse vollständig beschrieben, so können die Zustandsübergänge auf Vollständigkeit und Richtigkeit geprüft werden. Hierzu ist zu prüfen, ob in allen Zuständen für jedes Ereignis die intendierte Systemreaktion spezifiziert wurde.

Auf diese Weise kann eine anhand von Ursachen und Wirkungen textuell beschriebene, zustandsbehaftete Spezifikation in einem endlichen Zustandsübergangsmodell dargestellt und systematisch ergänzt werden. Anschließend sollte das Modell mit den Verfahren zum Test zustandsbasierter Systeme geprüft werden. Darüber hinaus besteht die Möglichkeit, zusätzliche dynamische Testfälle mit Hilfe der Verfahren des Abschnitts 4.2.2 der Ursache-Wirkungs-Analyse festzulegen.

4.4.4.3 Kombination mit strukturellen Betrachtungen des Prüflings

Eine Kombination der funktionsorientierten, zustandsbasierten Verfahren mit strukturellen Betrachtungen des Prüflings ist insbesondere dann sinnvoll, wenn die Realisierung des Prüflings in Form eines Zustandsautomaten sichergestellt ist. Für eine solche Realisierung bieten sich tabellarische Implementierungen an, wie sie beispielsweise in [Bei90] angeregt werden. Entsprechende Implementierungen können auch aus Werkzeugen zur Spezifikation von Zustandsübergangsmodellen erzeugt werden, soweit diese eine automatische Codegenerierung unterstützen.

Kann eine entsprechende Realisierung der Zustandsübergänge im Prüfling mit bekannten Strukturelementen vorausgesetzt werden, so kann es hilfreich sein, diese Strukturelemente zu instrumentieren und im dynamischen Test ihre Überdeckung zu fordern. So kann beispielsweise durch vollständige Überdeckung der Zustände des Prüflings sichergestellt werden, dass die Implementierung im Prüfling keine zusätzlichen Zustände enthält. Entsprechendes gilt für Zustandsübergänge. Darüber hinaus besteht durch eine Instrumentierung die Möglichkeit, den erreichten Zustand des Prüflings im dynamischen Test zu protokollieren, so dass dieser nicht anhand der Systemausgabe oder mit Hilfe der beobachtbaren Äquivalenz [DooFra94] ermittelt werden muss.

So können aus dem kombiniert funktions- und strukturorientierten dynamischen Test eines zustandsbasierten Prüflings sehr viel schärfere Aussagen als bei einem rein funktionsorientierten Vorgehen abgeleitet werden. Bei allen Formen der Instrumentierung ist jedoch zu beachten, dass diese einen Eingriff in die Realisierung des Prüflings darstellen und somit Auswirkungen auf sein Verhalten haben können, die das Testergebnis möglicherweise verfälschen.

4.5 Erweiterte Ansätze zum funktionsorientierten Test

Die in den vorangegangenen Abschnitten dieses Kapitels beschriebenen Verfahren zum funktionsorientierten Test sind auf eine systematische Prüfung der in einer Spezifikation beschriebenen funktiona-

len Anforderungen an einen Prüfling ausgerichtet. Eine Erweiterung dieses Verständnisses des funktionsorientierten Testens wird in den Veröffentlichungen [How80] und [How86] angeregt.

So wird in [How80] am Test eines Programmpakets mit Implementierungen mathematischer Funktionen eine Strategie für den dynamischen Test entwickelt, die sich speziell am mathematischen Funktionsbegriff orientiert. Die zu realisierende Funktion wird als Grundlage einer Teststrategie verwendet, die im Wesentlichen einem dynamischen Test funktionaler Äquivalenzklassen und ihrer Kombinationen entspricht. Betrachtet werden nicht nur numerische Parameter, sondern auch erweiterte Strukturen wie Felder und Matrizen mit ihren Dimensionen und mathematischen Eigenschaften. Auch für mathematische Funktionen, die als Argumente über geeignete Schnittstellen übergeben werden, werden Anforderungen an den dynamischen Test formuliert, die beispielsweise die Lage der Nullstellen berücksichtigen. Howden beschreibt sein Vorgehen zur Parameterwahl lediglich exemplarisch, weshalb keine allgemeine Verfahrensbeschreibung abgeleitet werden kann. Dennoch verdeutlicht sein Vorgehen anschaulich die notwendige Kreativität bei der Anwendung des Verfahrens der funktionalen Äquivalenzklassenanalyse im speziellen Kontext, der hier einen mathematisch-algorithmischen Charakter hat.

Howden beschränkt sich bei seiner Strategie jedoch nicht auf die Betrachtung von Prüfling und Spezifikation als monolithische Funktion, sondern regt darüber hinaus eine systematische funktionale Dekomposition an. Als Grundlage hierfür wird die interne Struktur des mathematischen Algorithmus verwendet. Dieser wird als Komposition von Teilfunktionen aufgefasst, die durch Kontrollstrukturen, von Howden als Designfunktionen oder Formen bezeichnet, miteinander verbunden sind. Als Grundstrukturen werden in [How86] algebraische, konditionale und iterative Verknüpfungen sowie die sequenzielle Ausführung identifiziert. Howden fordert für alle Teilfunktionen einen dynamischen Test nach funktionalen Aspekten und gibt zusätzliche Teststrategien für die Prüfung der Verknüpfungen vor. Zur technischen Realisierung regt er die Verwendung von Testrahmen an. Die Identifikation der funktionalen Struktur und der enthaltenen Teilfunktionen kann nach Howden auf unterschiedliche Weise geschehen. So empfiehlt er die Verwendung der funktionalen Dekomposition aus dem Design, soweit diese in Dokumenten beschrieben ist. Auch eine Analyse der Realisierung im Prüfling wird zur Bestimmung der relevanten Teilfunktionen und Verknüpfungen angeraten. Hierbei sind Kontrollstrukturen, Datenflussinformationen und interne Schnittstellen zu berücksichtigen. Der Bezug zur strukturellen Überdeckung und zum Code-Review wird diskutiert.

Aus diesem Vorgehen wird deutlich, dass Howden eine gleichzeitige Betrachtung funktionaler und struktureller Aspekte anstrebt. Er beschränkt sich dabei nicht auf die gleichzeitige Anwendung verschiedener Verfahren entsprechend Definition 3-12, sondern analysiert strukturelle Aspekte, um die funktionale Betrachtung systematisch zu verfeinern. Dabei nutzt er sowohl die Spezifikation und das Design als auch den Prüfling als Informationsquelle, so dass das resultierende Verfahren nicht als funktionsorientiert nach Definition 4-1 betrachtet werden kann. Da er hierbei über die Aufrufstruktur der Unterprogramme des Prüflings hinaus auch die Elemente der strukturierten Programmierung betrachtet, ergibt sich eine sehr detaillierte Betrachtung der geforderten und der implementierten funktionalen Zusammenhänge, die auch durch einen funktionsorientierten Test von Einzelkomponenten mit sukzessiver Integration nicht erreicht werden kann. Die Effektivität dieses Vorgehens für den Test mathematischer Software zeigt Howden anhand seiner Studie in [How80]; sie steht dem hohen analytischen Aufwand zur Durchführung des Verfahrens gegenüber.

5 Fazit und Ausblick

Ausgangspunkt für die vorliegende Arbeit waren die Überlegungen der Kapitel 1 und 2, die eine verstärkte Funktionsorientierung in der analytischen Softwarequalitätssicherung motivierten: Sie beschrieben die Softwareentwicklung als einen Prozess der sukzessiven Formalisierung von informal beschriebenen Anforderungen an die Funktionsweise von Software bis hin zu einer formalen, maschineninterpretierbare Darstellung, dem eigentlichen Softwareprodukt. Dieser Prozess der Formalisierung wurde als Kernprozess der Softwareentwicklung verstanden, der auch in der analytischen Qualitätssicherung besondere Berücksichtigung finden muss. Unterstützt wurde diese Überlegung durch die Forderung des Einsatzes funktionsorientierter Verfahern in der analytischen Qualitätssicherung durch relevante Normen sowie durch die Beobachtung, dass sich die Orientierung an Kundenbedürfnissen und –anforderungen als zunehmend kritischer Erfolgsfaktor in Softwareprojekten erweist. Diesen Betrachtungen gegenübergestellt wurde die Beobachtung, dass der funktionsorientierte Test in Wissenschaft und Praxis oftmals nur unzureichend betrachtet und wenig systematisch unterstützt beziehungsweise umgesetzt wird.

Aus dieser Motivation ergab sich das Ziel, eine systematische Untersuchung des funktionsorientierten Tests durchzuführen. Konkret sollte eine formale, vereinheitlichende Beschreibung für das Vorgehen in funktionsorientierten Test erreicht und als Basis einer eindeutigen Darstellung und systematischen Untersuchung der funktionsorientierten Testverfahren verwendet werden.

Die folgenden Abschnitte geben einen Überblick über die Ergebnisse dieser Arbeit und zeigen auf, wie durch ihre Verwendung in der Praxis eine Hinwendung zur Funktionsorientierung und damit eine besser an den funktionalen Anforderungen orientierte Gestaltung des Entwicklungsprozesses erreicht werden kann. Abschnitt 5.1 fasst dazu die Resultate der allgemeinen Modellbildung für den dynamischen Test zusammen. Diese liefern für die hier durchgeführte Analyse eine geeignete formale Basis; sie können darüber hinaus auch in weiterführenden Untersuchungen des dynamischen Tests genutzt werden. Abschnitt 5.2 zeigt die Ergebnisse der Untersuchung des funktionsorientierten Tests und spezieller praxisrelevanter Verfahren. Für diese konnte eine Systematisierung und Ergänzung der in der Literatur häufig sehr informalen Verfahrensbeschreibung erreicht werden. So wurde eine Grundlage für ihre Standardisierung und Weiterentwicklung erarbeitet, die bereits in der vorliegenden Untersuchung zu zahlreichen algorithmischen Ergänzungen sowie zum Aufzeigen von effizienten Verfahrenserweiterungen und –kombinationen geführt hat. Die Untersuchung der Verfahren verdeutlichte darüber hinaus formalisierbare und nicht formalisierbare Aspekte, deren Problematik, aber auch Nutzen in Abschnitt 5.3 diskutiert wird. Einen Ausblick auf eine verstärkte Funktionsorientierung im Entwicklungsprozess sowie eine Skizze zur Realisierung eines solchen Ansatzes in Form einer integrierten Entwicklungsumgebung gibt Abschnitt 5.4.

5.1 Ergebnisse zur grundlegenden Modellbildung für den dynamischen Test

Um eine formale Beschreibung der Grundlagen des funktionsorientierten Tests zu erreichen, wurden in Kapitel 3 zunächst Annahmen getroffen, die es erlaubten, das dynamische Testen als einen wohldefinierten und endlichen Prozess aufzufassen. Aufbauend darauf wurde mit Hilfe der allgemeinen Mengenlehre und Prädikatenlogik ein Modell zur Beschreibung des Vorgehens im dynamischen Test erarbeitet, das eine Definition der zentralen Begriffe im Einklang mit den in der Literatur verwendeten

Darstellungen ermöglichte. Hierbei wurden insbesondere relationale und funktionale Strukturen genutzt, um funktionale Aspekte wie beispielsweise die Begriffe der Spezifikation, des Prüflings, der Korrektheit eines Prüflings bezüglich einer Spezifikation, der Durchführung und Auswertung von dynamischen Testfällen, Testläufen und Testverfahren zu formulieren. So konnte eine formale Basis zur Definition der zentralen Begriffe des dynamischen Tests und zur Untersuchung seiner Verfahren geschaffen werden. Auch weitere, in der Literatur gebräuchliche Begriffe wie die Kombination und Subsumption von dynamischen Testverfahren, ihre Anwendbarkeit („applicability“ im Sinne von [FraWey88]) sowie die Auswertung des dynamischen Tests mit Hilfe von Testorakeln oder einem erweiterten Akzeptanzbegriff konnten einfach in dieses Modell integriert werden. Abschließend wurden Erweiterungen des Modells angeregt, die seine Verwendung auch in anderen Kontexten ermöglichen. Tabelle 5-1 stellt den Nutzen und die positiven Aspekte des entwickelten Modells seinen Einschränkungen und Erweiterungsmöglichkeiten gegenüber.

Nutzen und positive Aspekte des Modells zum dynamischen Test	**Einschränkungen und Erweiterungsmöglichkeiten des Modells zum dynamischen Test**
Klare Modellbildung auf Basis der Annahmen: Wenige, klare Annahmen genügen als Grundlage der Modellbildung zum dynamischen Test und ermöglichen eine saubere, mathematisch analysierbare Definition als wohldefinierter endlicher Prozess. Flexible Gestaltung der Datenbereiche: Bei der Gestaltung der Datenbereiche für Eingabe, Ausgabe und Gedächtnis wurden keinerlei Einschränkungen vorgegeben. So wurde beispielsweise auch einen Berücksichtigung von Systemen ermöglicht, die neben Software- auch Hardwarekomponenten sowie beliebig gestaltete Sensoren und Aktoren enthalten. Ein Formalismus zur Beschreibung mehrdimensionaler Parameterbereiche wurde angegeben. Flexible Gestaltung von Prüfling und Spezifikation: Die einfache Modellierung der Spezifikation als Relation und des Prüflings als partielle Funktion ohne zusätzliche Annahmen bezüglich Form und Gestaltung ermöglicht den Einsatz des Modells für unterschiedliche Typen von Spezifikationen und Prüflingen. Begriffsdefinitionen in Einklang mit Literatur und Vorgehen der Praxis: Die einfache relationale Beschreibung der Korrektheit eines Prüflings bezüglich einer Spezifikation, die Definition der Durchführung und des Bestehens von dynamischen Testfällen, Testläufen und Testverfahren sowie die Formulierung zentraler Begriffe, wie der der Subsumption, der Anwendbarkeit und der Kombination von dynamischen Testverfahren, stehen im Einklang mit der gebräuchlichen Verwendung der Begriffe in der Literatur. Die Formulierung von zentralen Aspekten des dynamischen Tests, so zum Beispiel der Begriffe der Subsumption und der Kombination von dynamischen Testverfahren stehen in Einklang mit dem gängigen Vorgehen in der Praxis. Das Modell bietet zudem mit den Definitionen des suboptimalen Testorakels und der erweiterten Akzeptanzrelation die Möglichkeit zur formalisierten Beschreibung und Analyse von in der Praxis auftretenden Ungenauigkeiten bei der Auswertung des dynamischen Tests. Eignung als Grundlage der Untersuchung des funktionsorientierten Test Das Modell enthält alle wichtigen, für die Beschreibung des Vorgehens im dynamischen Test notwendigen Definitionen und hat sich in Kapitel 4 als geeignete Grundlage zur Beschreibung und Untersuchung der Verfahren des funktionsorientierten Tests erwiesen. Erweiterbarkeit: Die einfache Erweiterbarkeit des Modells zeigte sich zum Beispiel bei der Beschreibung des Tests zustandsbasierter Systeme.	Restriktion der Anwendbarkeit des Modells aufgrund der zugrundegelegten Annahmen: Die vorausgesetzte Ein-Schritt-Verarbeitungssemantik ist ungeeignet für die Beschreibung des dynamischen Tests von reaktiven Systemen, z. B. Regelsystemen. Die im Modell vorausgesetzte, für eine eindeutige Auswertung des dynamischen Tests notwendige Eindeutigkeit und Vollständigkeit der Spezifikation ist in der Praxis im Allgemeinen nicht gegeben. Die Endlichkeit der Menge der Testdaten wird im Modell vorausgesetzt. Sie ist in realen Testprozessen im Allgemeinen garantiert, muss jedoch bei der Definition von Testverfahren zur Erreichung eines anwendbaren Verfahrens explizit sichergestellt werden. Die Forderung eines Performanzkriteriums für jedes Eingabedatum ist in realen Spezifikationen selten erfüllt, so dass eine implizite Annahme der akzeptablen Performanz bei der Testauswertung zugrunde gelegt werden muss. Ein deterministisches Verhalten des Prüflings kann in realen Prozessen nicht immer sichergestellt werden (z. B. Beeinflussung durch menschliche Interaktion, Zugriff auf verteilte, gemeinsam genutzte Ressourcen). Zur Betrachtung eines stochastischen Verhaltens des Prüflings sind umfassende Erweiterungen im Modell notwendig. Keine Betrachtung wirtschaftlicher Aspekte: Das Modell sieht keine Betrachtung wirtschaftlicher Aspekte des dynamischen Tests vor, beispielsweise zu den Kosten dynamischer Testläufe oder zu vermiedenen Fehlleistungskosten. Für entsprechende Betrachtungen kann das Modell wie in Abschnitt 3.3.2 skizziert erweitert werden. Keine Betrachtung von nicht-funktionalen Anforderungen: Die Darstellung nicht-funktionaler Anforderungen wie Sicherheits-, Zuverlässigkeits- und Verfügbarkeitsanforderungen im Modell nicht vorgesehen. Lediglich die Performanz zu einzelnen Eingabedaten wird im Rahmen der Spezifikation berücksichtigt. Zur Betrachtung von Sicherheits-, Zuverlässigkeits- und Verfügbarkeitsanforderungen muss das Modell wie in Abschnitt 3.3.3 dargestellt erweitert werden. Keine Betrachtung des internen Datenzustand des Prüfling Eine Betrachtung oder Beeinflussung des internen Datenzustands des Prüflings im dynamischen Test ist im Modell nicht vorgesehen und kann nur mit zusätzlichem Eingriff in den Prüfling wie in Abschnitt 3.3.5 dargestellt realisiert werden.

Tabelle 5-1: Gegenüberstellung des Nutzens und der Einschränkungen des in dieser Arbeit entwickelten und zugrunde gelegten Modells zur Beschreibung des Vorgehens im dynamischen Test

5.2 Ergebnisse zu den Verfahrensuntersuchungen des funktionsorientierten Tests

In Kapitel 4 wurde speziell der Begriff des funktionsorientierten Tests im Rahmen des Modells formuliert. Dabei wurde auch die für das funktionsorientierte Testen typische Zerlegung der Spezifikation in handhabbare Teilspezifikationen beschrieben, deren kombinierte Prüfung als Prüfung der Gesamt-

spezifikation akzeptiert wird. Somit stand eine systematische Basis für die Formalisierung und Diskussion der bekannten und praxisrelevanten Verfahren des funktionsorientierten Tests bereit.

Die Analyse der Verfahren der funktionsorientierten Partitionierung sowie der funktionalen Äquivalenzklassen- und Grenzwertanalyse haben gezeigt, dass der Begriff der funktionalen Äquivalenz in allgemeiner Form nur informal zu fassen ist, so dass die Ableitung der funktionalen Partitionen beziehungsweise Äquivalenzklassen immer einer kreativen Leistung des Testers bedarf. Auch für den Begriff des Grenzwertes wurde gezeigt, dass eine weniger formale Definition orientiert an den Fehlererwartungen des Testers zielführender im Hinblick auf eine effiziente Testdurchführung sein kann als die formale Definition auf Basis einer Topologie. Bei der formalen Beschreibung der Verfahren wurde weiterhin deutlich, dass wichtige Aspekte in den klassischen Darstellungen oft nicht angesprochen werden. So zeigte sich, dass die Abdeckung ungültiger funktionaler Ausgabeäquivalenzklassen im Vollständigkeitskriterium nicht aufgenommen werden kann, da sie auf Basis von Fehlererwartungen beruht und nicht spezifikationsbasiert formuliert werden kann. Auch bei der Betrachtung verwandter Verfahren, wie des kombinatorischen Tests auf Basis einer funktionalen Äquivalenzklassenanalyse, half die formale Verfahrensbeschreibung bei der Unterscheidung funktionsorientierter und nicht funktionsorientierter Verfahrensvarianten. Bei der Untersuchung von Kombinationsmöglichkeiten mit anderen Verfahren des dynamischen Tests zeigte sich, dass sich die Verfahren der funktionsorientierten Partitionierung insbesondere zur Kombination mit den strukturorientierten Verfahren des Pfadbereichstests [WhiCoh80] eignen. Hierdurch ergeben sich aussagekräftige dynamische Testverfahren, die in ihrer Aussagefähigkeit unter geeigneten Voraussetzungen einem Korrektheitsnachweis gleichkommen können [WeyOst80, RicCla81, RicCla85]. Auf Basis der hier formulierten Verfahrensbeschreibung konnte weiterhin ein Ansatz der funktionalen Partitionierung auch für die Prüfung eines spezifizierten Performanzverhaltens angeregt werden. Die Ergebnisse der Verfahrensdefinitionen und -untersuchungen zu den Verfahren der funktionalen Partitionierung sind in Tabelle 5-2 am Ende dieses Abschnitts zusammengestellt.

Bei der Betrachtung der Ursache-Wirkungs-Analyse und des verwandten Tests auf Basis von Entscheidungstabellen wurde deutlich, dass die hierzu notwendige Formalisierung der funktionalen Anforderungen einer Modellbildung entspricht. Diese unterstützt die Vollständigkeit der Beschreibung und die Systematisierung der Prüfung. Es wurde dargestellt, wie das Modell semantisch über den Ein- und Ausgabedatenbereichen interpretiert wird, und wie auf dieser Basis die Auswahl und Bewertung von Testdaten stattfindet. Die ursprünglich heuristische Darstellung der Auswahl der dynamischen Testfälle [Mye79] wurde algorithmisch als Teil der Modellbildung verankert. Dies ermöglichte eine einfache prädikatenlogische Verfahrensdarstellung. In einer beispielhaften Untersuchung wurde deutlich, wie viel Interpretationsspielraum die informalen Darstellungen der Verfahren der Ursache-Wirkungs-Analyse trotz umfangreicher sprachlicher Formulierung lassen, und welche Unvollständigkeiten dabei im dynamischen Test entstehen können. Auf dieser Basis wurde gezeigt, welche zusätzlichen Angaben für eine eindeutige Darstellung notwendig sind, und wie darüber hinaus weitere aussagekräftige Testfälle bestimmt werden können. Zusätzlich wurde deutlich, dass gewisse Minimalkriterien für die Verfahren einzuhalten sind, damit auch beim Vorliegen von Randbedingungen eine ausreichende Prüfung sichergestellt ist. Hierbei wurde die Parallelität zu den Verfahren der strukturorientierten Bedingungsüberdeckung deutlich. Dies regte eine Verwendung der strukturorientierten Minimalkriterien in der funktionsorientierten Betrachtung an und legte die kombinierte Anwendung der Verfahren im dynamischen Test nahe. Die Verwandtschaft der Verfahren der Ursache-Wirkungs-Analyse zu den Verfahren der funktionalen Partitionierung, die sich aus der semantischen Interpretation des Ursache-Wirkungs-Modells über den Datenbereichen ergibt, regte darüber hinaus eine An-

wendung zusätzlicher Kriterien wie beispielsweise der Ansätze der Grenzwertanalyse an. Weitere Möglichkeiten zur Erweiterung der rein kausalen Betrachtungen der Ursache-Wirkungs-Analyse auf kombiniert kausal-temporale Zusammenhänge wurden durch die prädikatenlogische Darstellung der Zusammenhänge nahe gelegt. Eine Ergebnisübersicht zu den Verfahren zum Test von Ursachen und ihren Wirkungen wird am Ende des Abschnitts in Tabelle 5-3 gegeben.

Die anschließende Betrachtung der funktionsorientierten Testverfahren zur Prüfung graphisch spezifizierter Funktionalität setzte zunächst eine semantische Interpretation von Strukturelementen der Spezifikation über dem Ein- und Ausgabedatenbereich voraus. Diese ermöglichte die Beschreibung der Testdatenselektion und Testdatenadäquatheit anhand des graphischen Modells. So wurden Strukturelemente mit Teilmengen der Ein- und Ausgabedatenbereiche assoziiert, sodass das graphische Modell als relationale Beschreibung zwischen Ein- und Ausgabedaten verstanden werden konnte. Dies ermöglichte die Definition der Überdeckung von Strukturelementen durch dynamische Testfälle. Auf dieser Basis konnten zunächst generische Verfahren zur Überdeckung der Strukturelemente beschrieben werden, die anschließend für gebräuchliche Spezifikationsformen wie Syntaxgraphen, Aktivitätsdiagramme, Sequenz- und Kollaborationsdiagramme angepasst wurden. Auch für Petrinetze wurde beispielhaft aufgezeigt, wie sie in einer funktionsorientierten Beschreibung und Prüfung genutzt werden können. Im Bezug auf die Spezifikationsformen wurde reflektiert, wie eine möglichst vollständige und unmissverständliche semantische Interpretation der graphischen Beschreibungsformen erreicht werden kann. So wurde zur Beschreibung der Strukturelemente eine intensive Nutzung von Prädikaten angeregt, die die assoziierten Daten kenntlich machen. Eine Übersicht der Ergebnisse zeigt Tabelle 5-4.

Lediglich für zustandsbasierte Beschreibungsformen hat sich die so beschriebene Auffassung der graphischen Beschreibung als Modell und die Interpretation über dem Ein- und Ausgabedatenbereich als unzureichend erwiesen. Hier musste für eine sinnvolle Betrachtung der spezifizierten und implementierten Funktionalität neben den Ein- und Ausgabedaten auch der interne Zustand des Prüflings betrachtet werden, was nur durch eine Erweiterung des zugrundegelegten Modells des dynamischen Tests erreicht werden konnte. So wurde der Ein- und Ausgabedatenbereich jeweils um einen internen Speicherbereich erweitert, was die Definition einer gedächtnisbehafteten Spezifikation und eines gedächtnisbehafteten Prüflings ermöglichte. Dies führte zu einer Erweiterung des Korrektheitsbegriffs eines Prüflings bezüglich seiner Spezifikation im Hinblick auf das Gedächtnis, zu einer Berücksichtigung des Gedächtnisses bei dynamischen Testfällen, Testläufen und Testverfahren und zu einer Definition von dynamischen Testsequenzen, die bei der Prüfung eines gedächtnisbehafteten Prüflings von zentraler Bedeutung sind. Bei der Betrachtung von zustandsbasierten Spezifikationen wurde deutlich, dass sie als Modelle des Verhaltens eines gedächtnisbehafteten Prüflings aufzufassen sind und über dem Ein- und Ausgabedatenbereich sowie dem internen Speicherbereich semantisch interpretiert werden müssen. Erst mit Hilfe dieser Interpretation konnten die bekannten dynamischen Testverfahren auf zustandsbasierten Spezifikationen prädikativ beschrieben werden, wie beispielsweise die Zustandsüberdeckung oder auch die Überdeckung von allen Ereignissen in allen Zuständen. Die Verwendung von Testsequenzen ermöglichte die Erfüllung von Kriterien, die für Pfade durch den Zustandsraum [Lig02] oder auch für Ereignisfolgen [Cho78] in der Literatur beschrieben sind. Bei der Diskussion der Kombination zustandsbasierter und nicht-zustandsbasierter Ansätze im dynamischen Test wurde deutlich, wie Verfahren der funktionalen Partitionierung und der Ursache-Wirkungs-Analyse bei der Modellbildung der zustandsbasierten Spezifikation unterstützen können. Darüber hinaus ergaben sich aus der kombinierten Betrachtung Hinweise für einen erweiterten Test gedächtnisbehafteter Prüflinge: So kann die funktionale Äquivalenzklassenanalyse beispielsweise ergänzend zu

anderen Verfahren für Klassen der objektorientierten Programmierung mit umfangreichen Schnittstellen verwendet werden. Grenzwertbetrachtungen legen zusätzliche dynamische Testfälle bei der Betrachtung des Gedächtnisses eines Prüflings nahe. Ursache-Wirkungs-Zusammenhänge, die Zustandswechsel beschreiben, wurden ebenfalls zur Ableitung zusätzlicher Testfälle empfohlen. Als Spezialfall wurde die in [Bei90] angeregte kombinierte funktions- und strukturorientierte Betrachtung eines gedächtnisbehafteten Prüflings analysiert, dessen zustandsbasierte Realisierung sichergestellt und instrumentiert werden kann: Hier kann, wie bereits in [Cho78] ausgeführt, aus bestandenen Testfällen ein Nachweis über die Abwesenheit bestimmter Fehlerklassen erbracht werden. Tabelle 5-5 stellt die Ergebnisse zum dynamischen Test auf Basis einer zustandsbasierten Spezifikation zusammen.

Abschließend wurde das Ineinandergreifen funktionsorientierter und strukturorientierter Betrachtungsweisen vorgestellt, das in [How80] und [How86] zu einer Gesamtstrategie eines funktionsorientierten Vorgehens im dynamischen Test kombiniert wird. Dieser Ansatz regt durch die Idee der funktionalen Dekomposition insbesondere die Einbettung funktionsorientierter Sichtweisen in den Entwicklungsprozess an.

Die folgende tabellarische Zusammenstellung liefert eine Übersicht zu den Ergebnissen der Verfahrensuntersuchungen des Kapitels 4. Hier werden die formalisierbaren und nicht formalisierbaren Aspekte der Verfahren des funktionsorientierten Tests zusammengestellt, so dass deutlich wird, in welchem Rahmen Möglichkeiten für eine automatisierte Unterstützung in der Durchführung der Verfahren bestehen und wo die Kreativität des Testers eingebracht werden muss. Darüber hinaus werden die in dieser Arbeit entwickelten Ansätze zur Ergänzung und Erweiterung der Verfahren aufgeführt.

Verfahren der funktionalen Partitionierung (Abschnitt 4.1)		
Verfahrenstypen mit relevanten Unterverfahren		
Funktionale Partitionierung des Eingabebereichs (Abschnitt 4.1.1) Unterverfahren: Neben funktionsorientierten sind auch strukturorientierte Verfahrensausprägungen und Ausprägungen auf Basis von Fehlererwartung möglich	Funktionale Äquivalenzklassenanalyse (Abschnitt 4.1.2) Unterverfahren: Kombinatorischer Test der funktionalen Äquivalenzklassen nach Schröder und Korel (Abschnitt 4.1.3)	Grenzwertanalyse (Abschnitt 4.1.4) Unterverfahren: Grenzwertanalyse auf Basis einer funktionalen Äquivalenzklassenbildung, Grenzwertanalyse für eine funktionale Partitionierung des Eingabebereichs
nicht formalisierbarer Anteil		
Ableitung der funktionalen Partitionierung aus der Spezifikation Testauswertung bezüglich Konformität mit der Spezifikation	Ableitung der funktionalen Äquivalenzklassen aus der Spezifikation Testauswertung bezüglich Konformität mit der Spezifikation	Ableitung der funktionalen Äquivalenzklassen bzw. der funktionalen Partitionierung aus der Spezifikation, Wahl einer Topologie beziehungsweise eines geeigneten Grenzwertbegriffs Testauswertung bezüglich Konformität mit der Spezifikation
formalisierbarer Anteil, Möglichkeiten zur automatisierten Verfahrensunterstützung		
Automatisierte Unterstützung der Beschreibung und Verwaltung der funktionalen Partitionen in der Spezifikation und über den Ein- und Ausgabedatenbereichen Identifikation und Aufzeigen von Datenmengen in den Ein- und Ausgabedatenbereichen, die nicht in der spezifizierten Partitionierung berücksichtigt wurden Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen im Hinblick auf die funktionale Partitionierung Stochastische Auswahl von Testdaten zu dynamischen Testläufen auf einer gegebenen funktionalen Partitionierung, gegebenenfalls auch nach vorgegebenem Nutzungsprofil	Automatisierte Unterstützung der Beschreibung und Verwaltung der funktionalen Äquivalenzklassen in der Spezifikation und über den Ein- und Ausgabeparameterbereichen Identifikation und Aufzeigen von Parametermengen, die nicht in der Spezifikation berücksichtigt wurden Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen im Hinblick auf die funktionale Äquivalenzklassenbildung Stochastische Auswahl von Testdaten zu dynamischen Testläufen auf einer gegebenen funktionalen Äquivalenzklassenbeschreibung, gegebenenfalls auch mit vorgegebenem Nutzungsprofil	Automatisierte Unterstützung der Wahl einer Topologie oder eines geeigneten Grenzwertbegriffs für Parameterbereiche eines speziellen Datentyps; hierbei können gegebenenfalls Schätzungen der Anzahl der notwendigen dynamischen Testfälle zur Erreichung des Testdatenadäquatheit in Abhängigkeit von der Gestaltung der Partitionen angegeben werden Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen im Hinblick auf die funktionale Äquivalenzklassenbeschreibung mit Grenzwertbildung Auswahl von Testdaten zu dynamischen Testläufen auf einer gegebenen funktionalen Äquivalenzklassenbeschreibung mit Grenzwertbegriff
Erweiterungsmöglichkeiten (Abschnitt 4.1.5)		
Erweiterung auf funktionale Partitionen des Ausgabebereichs Kombination mit strukturorientierten Verfahren des Partitionstests	Betrachtung von funktionalen Äquivalenzklassen über einem mehrdimensionalen Parameterbereich Betrachtung der Abhängigkeit zwischen Äquivalenzklassen auf Basis der Spezifikation als Grundlage einer reduzierten kombinatorischen Prüfung Äquivalenzklassenanalyse des Performanzbereichs (hohe, mittlere, geringe Performanzanforderungen)	Grenzwertbegriff für verschiedene Datentypen (numerisch eindimensional, numerisch mehrdimensional, textuell, Listen, Strukturen, Aufzählungstypen) Grenzwertanalyse des Verhaltens des Prüflings bzgl. des Performanzbereichs (hohe, mittlere, geringe Performanzanforderungen)

Tabelle 5-2: Ergebnisse zu den Verfahren der funktionalen Partitionierung

Verfahren zum Test von Ursachen und ihren Wirkungen (Abschnitt 4.2)	
Verfahrenstypen mit relevanten Unterverfahren	
Dynamischer Test auf Basis von Entscheidungstabellen (Abschnitt 4.2.1) Unterverfahren: Test auf Basis von Entscheidungsbäumen	Ursache-Wirkungs-Analyse (Abschnitt 4.2.2) Unterverfahren: Klassische Ursache-Wirkungs-Analyse Erweiterte Ursache-Wirkungs-Analyse (Abschnitte 4.2.2.2.1, 4.2.2.2.2)
nicht formalisierbarer Anteil	
Beschreibung der Ursachen und Wirkungen, möglichst mit prädikativer semantischer Interpretation über dem Ein- und Ausgabedatenbereich Analyse der Ursachenkombinationen und ihrer Wirkungen am Spezifikationstext, dabei Aufstellung der relevanten Ursache-Wirkungs-Kombinationen mit Vermerk in Entscheidungstabelle bzw. Entscheidungsbaum Reduktion der Ursache-Wirkungs-Tabelle als Kompromiss zwischen Testvollständigkeit und beherrschbarem Testumfang Erstellung der dynamischen Testfälle (gegebenenfalls ist Unterstützung auf Basis der prädikativen semantischen Interpretation möglich) Testauswertung bezüglich Konformität mit Spezifikation	Ableitung des Ursache-Wirkungs-Modells aus dem Spezifikationstext: Beschreibung der Ursachen und Wirkungen, möglichst mit prädikativer semantischer Interpretation über dem Ein- und Ausgabedatenbereich Erfassung der Ursache-Wirkungs-Zusammenhänge aus der Spezifikation Erfassung der Randbedingungen auf Basis der Spezifikation und der semantischen Interpretation Erstellung der dynamischen Testfälle (gegebenenfalls ist Unterstützung auf Basis der prädikativen semantischen Interpretation möglich) Testauswertung bezüglich Konformität mit Spezifikation
formalisierbarer Anteil, Möglichkeiten zur automatisierten Verfahrensunterstützung	
Automatisierte Unterstützung der Beschreibung und Verwaltung von Ursachen und Wirkungen, auch im Hinblick auf die semantische Interpretation über dem Ein- und Ausgabedatenbereich Identifikation und Aufzeigen von Datenmengen in den Ein- und Ausgabedatenbereichen, die nicht in den Ursachen beziehungsweise Wirkungen berücksichtigt wurden Halbautomatische Vervollständigung der Ursache-Wirkungs-Tabelle durch Ergänzung unberücksichtigter logischer Ausdrücke, gegebenenfalls mit Prüfung der semantischen Sinnhaftigkeit, soweit dies auf Basis der prädikativen Ursachenbeschreibung möglich ist Automatisierte Unterstützung bei der Reduktion der vollständigen Ursache-Wirkungs-Tabelle durch Suchfunktion von Ursachenkombinationen mit gleicher Wirkung und Unterschied in nur einer Ursachenausprägung Automatisierte Auswahl von adäquaten dynamischen Testläufen, soweit eine Partitionierung des Eingabenbereichs aus der prädikativen Ursachenbeschreibung ableitbar ist Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen Unterstützung bei der Testauswertung durch Anwendung der Ursache-Wirkungs-Tabelle als suboptimales Testorakel, soweit die semantische Interpretation der Wirkungen eine Analyse der Ausgabedaten unterstützt	Automatisierte Unterstützung der Beschreibung und Verwaltung von Ursachen und Wirkungen, auch im Hinblick auf die semantische Interpretation über dem Ein- und Ausgabedatenbereich Identifikation und Aufzeigen von Datenmengen in den Ein- und Ausgabedatenbereichen, die nicht in den Ursachen beziehungsweise Wirkungen berücksichtigt wurden Algorithmische Ableitung der Ursachenkombinationsforderungen aus dem Ursache-Wirkungs-Modell, gegebenenfalls Prüfung gegen zusätzliche Minimalkriterien (siehe Erweiterungsmöglichkeiten) Automatisierte Auswahl von adäquaten dynamischen Testläufen, soweit eine Partitionierung des Eingabebereichs aus der prädikativen Ursachenbeschreibung ableitbar ist Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen Unterstützung bei der Testauswertung durch Anwendung der Ursache-Wirkungs-Tabelle als suboptimales Testorakel, soweit die semantische Interpretation der Wirkungen eine Analyse der Ausgabedaten unterstützt
Erweiterungsmöglichkeiten	
Anwendung von Entscheidungstabellen oder Entscheidungsbäumen bereits als Beschreibungsform in der Spezifikation	Erweiterungen im Hinblick auf Effizienz Nur dann Forderung von mehr als einer Wahrheitswertbelegung für eine komplexe Teilbedingung, wenn der Wahrheitswert der Teilbedingung unter den vorgegebenen Wahrheitswertbelegungen der anderen Bedingungen die Wirkung beeinflussen kann Verzicht auf mehrfache Prüfung der Realisierung einer komplexen Teilbedingung im Prüfling Erweiterungen im Hinblick auf die Vollständigkeit Einbindung zusätzlicher Prüfkriterien angelehnt an die Kriterien der Bedingungsüberdeckung (einfach, minimal mehrfach, modifizierte Bedingungs- und Entscheidungsüberdeckung) zur Sicherstellung einer umfassenden Prüfung auch dann, wenn Randbedingungen zur Elimination einzelner Ursachenkombinationsforderungen geführt haben Kombination mit der strukturorientierten Bedingungsüberdeckungstest mit dem Ziel, sowohl die spezifizierten als auch die realisierten kausalen Zusammenhänge zu berücksichtigen Kombination mit der Grenzwertanalyse Betrachtung von Ursache-Wirkungs-Zusammenhängen in der Spezifikation zustandsbasierter Systeme Betrachtung von Ursache-Wirkungs-Zusammenhängen im Hinblick auf die Performanz Betrachtung kausal-temporaler Ursache-Wirkungs-Zusammenhänge

Tabelle 5-3: Ergebnisse zu den Verfahren zum Test von Ursachen und ihren Wirkungen

Verfahren zum Test von graphisch spezifizierter Funktionalität (Abschnitt 4.3)			
Verfahrenstypen mit relevanten Unterverfahren			
Generische Verfahren zum Test von graphisch spezifizierter Funktionalität: Knoten- , Kanten- und Pfadüberdeckung in pfadzusammenhängenden Graphen, Überdeckung der Pfade mit höchstens n - maligem Durchlauf der enthaltenen Kreise eines pfadzusammenhängenden Graphen, Überdeckung der bis zu n - fachen Ausführung aller Kreise eines pfadzusammenhängenden Graphen			
Dynamischer Test auf Basis von Syntaxgraphen (Abschnitt 4.3.3.1)	Dynamischer Test auf Basis von Aktivitätsdiagrammen (Abschnitt 4.3.3.2)	Dynamischer Test auf Basis von Sequenz- und Kollaborationsdiagrammen (Abschnitt 4.3.3.3)	Dynamischer Test auf Basis von Petri-Netzen (Abschnitt 4.3.3.5)
nicht formalisierbarer Anteil			
Spezifikation des Syntaxgraphen Erstellung der dynamischen Testfälle, die nicht vom Syntaxgraphen akzeptiert werden Testauswertung bezüglich Konformität mit Spezifikation (gesamt)	Spezifikation des Aktivitätsdiagramms mit Angabe der Transitionsbedingungen, möglichst prädikativ über dem Eingabebereich, Beschreibung der Assoziation der erweiterten Pfade Erstellung der dynamischen Testfälle Testauswertung bezüglich Konformität mit Spezifikation (gesamt)	Spezifikation des Sequenz- oder Kollaborationsdiagramms mit prädikativer Angabe der Transitionsbedingungen, Beschreibung der assoziierten Eingabedaten Erstellung des dynamischen Testfalls, bei Bedarf auch mehrerer dynamischer Testfälle Testauswertung bezüglich Konformität mit Spezifikation (gesamt)	Spezifikation Petri-Netzes mit Angabe der Startmarkierungsassoziation Erstellung von dynamischen Testfällen Testauswertung bezüglich Konformität mit Spezifikation (gesamt) Bei Bedarf Anpassung der semantischen Interpretation von Petri-Netzen als Spezifikation über dem ein- und Ausgabebereich
formalisierbarer Anteil, Möglichkeiten zur automatisierten Verfahrensunterstützung			
Automatisierte Unterstützung der Beschreibung und Verwaltung von Syntaxgraphen Unterstützung bei der Generierung von dynamischen Testfällen, die durch den Syntaxgraphen akzeptiert werden (Achtung: Eine Prüfung der aus dem Modell abgeleiteten Testfälle ist für eine ebenfalls aus dem Modell abgeleitete Realisierung des Prüflings im Allgemeinen nicht sinnvoll) Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen Unterstützung bei der Testauswertung auf Basis des Syntaxgraphen	Automatisierte Unterstützung der Beschreibung und Verwaltung von Aktivitätsdiagrammen Unterstützung bei der Generierung von dynamischen Testfällen auf Basis der prädikativen Transitionsbedingungen und der Assoziation der erweiterten Pfade, soweit angegeben und automatisiert auswertbar Identifikation von Teilmengen des Eingabebereichs, die nicht zu erweiterten Pfaden assoziiert sind oder den prädikativen Transitionsbedingungen entsprechen Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen auf Basis der Assoziation der erweiterten Pfade Unterstützung bei der Testauswertung durch Vergleich der spezifizierten und der ausgeführten Aktivitäten, soweit automatisiert beobachtbar	Automatisierte Unterstützung der Beschreibung und Verwaltung von Aktivitätsdiagrammen Unterstützung bei der Generierung des dynamischen Testfalls auf Basis der prädikativen Transitionsbedingungen Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testfällen Unterstützung bei der Testauswertung durch Vergleich des spezifizierten und des beobachteten Transaktions- oder Kommunikationsflusses, soweit automatisiert beobachtbar	Automatisierte Unterstützung der Beschreibung, Verwaltung und Analyse (Erreichbarkeitsgraph, Zielmarkierungen, Transitionsverhalten) von Petri-Netzen Unterstützung bei der Generierung von dynamischen Testfällen (beispielsweise durch Analyse des Erreichbarkeitsgraphen im Hinblick auf bestimmte Markierungsbedingungen, deren Erreichung durch das Überdeckungskriterium gefordert ist) Unterstützung bei der Testauswertung durch Vergleich des spezifizierten und des beobachteten Systemverhaltens, soweit automatisiert beobachtbar
Erweiterungsmöglichkeiten			
Nutzung der formalen Beschreibung der generischen Verfahren zum Test von graphisch spezifizierter Funktionalität auch für die Beschreibung des dynamischen Tests auf Graphen, die aus der Struktur des Prüflings abgeleitet sind (Anweisungs-, Zeig- und Pfadabdeckung, strukturierter Pfadtest).			

Tabelle 5-4: Ergebnisse zu den Verfahren zum Test von graphisch spezifizierter Funktionalität

Verfahren zum Test zustandsbasierter Systeme (Abschnitt 4.4)
Verfahrenstypen
Zustandsüberdeckung Überdeckung aller Zustandsübergänge Überdeckung aller Ereignisse in jedem Zustandsübergang Überdeckung aller Ereignisse in allen Zuständen Überdeckung der Zustandsübergänge in nicht fehlerbehaftete Zustände Überdeckung aller Pfade einer vorgegebenen Länge Überdeckung aller Ereignisfolgen mit vorgegebener Länge in jedem Zustand
nicht formalisierbarer Anteil
Analyse des intendierten Systemverhaltens mit dem Ziel der Formulierung einer zustandsbasierten Spezifikation in Form eines zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe (Modellbildung) Festlegung von Eingabe-, Ausgabe- und Gedächtnisbereich (erweiterter Ein- / Ausgabebereich) Beschreibung der semantischen Interpretation des zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe über dem erweiterten Ein- / Ausgabebereich, beispielsweise durch Angabe der funktionalen Partitionierungen des Speicherbereichs, des Eingabebereichs und, falls gewünscht, des Ausgabebereichs Kennzeichnung fehlerbehafteter Zustände
formalisierbarer Anteil, Möglichkeiten zur automatisierten Verfahrensunterstützung
Automatisierte Unterstützung der Beschreibung und Verwaltung von zeitbehafteten endlichen Zustandsübergangsmodellen mit Ausgabe Identifikation von Teilmengen des Eingabebereichs und des Gedächtnisbereichs, die nicht in der semantische Interpretation berücksichtigt sind (Unvollständigkeit der Zustands- und der Ereignisidentifikation) Unterstützung bei der vollständigen Definition der Zustandsübergangsfunktion durch Abfrage der intendierten Systemreaktion für alle Ereignisse in allen Zuständen. Unterstützung bei der Generierung von dynamischen Testfällen zu den Überdeckungskriterien auf Basis der semantischen Interpretation des zeitbehafteten endlichen Zustandsübergangsmodells, hierbei Verwendung der semantischen Interpretation mit den Relationen der Zustands-, Ereignis- und Ausgabeidentifikation Prüfung der Testdatenadäquatheit zu beliebigen dynamischen Testläufen auf Basis des zeitbehafteten endlichen Zustandsübergangsmodells mit Ausgabe mit Hilfe der Zustands- und Ereignisidentifikation der semantischen Interpretation Unterstützung bei der Testauswertung durch Verwendung des zeitbehafteten endlichen Zustandsübergangsmodells als suboptimales Testorakel, beispielsweise mit Hilfe von Zustands- und Ausgabeidentifikation der semantischen Interpretation.
Erweiterungsmöglichkeiten
Anwendung von Verfahren der funktionalen Partitionierung bei der Beschreibung von Zuständen und Ereignissen, insbesondere Anwendung der funktionalen Äquivalenzklassenanalyse in mehrdimensionalen Parameterbeeichen. Anwendung von Verfahren der funktionalen Partitionierung, insbesondere der funktionalen Äquivalenzklassenanalyse, bei der Prüfung von parametrisierten Zustandsübergängen, beispielsweise in der objektorientierten Programmierung bei umfangreichen Methodenschnittstellen Anwendung der Ursache-Wirkungs-Analyse bei der Ableitung des endlichen Zustandsübergangsmodells aus einer textuell beschriebenen Spezifikation zur Unterstützung der vollständigen Beschreibung von Zustandsraum und Zustandsübergängen Anwendung der Ursache-Wirkungs-Analyse bei der Ableitung des endlichen Zustandsübergangsmodells aus einer textuell beschriebenen Spezifikation als Grundlage für die Auswahl zusätzlicher dynamischer Testdaten und als zusätzliches Kriterium für die Bewertung der Testdatenadäquatheit Erweiterungsmöglichkeiten durch Kombination von funktionsorientiertem und strukturorientiertem Testen für einen Prüfling, der mit Hilfe einer zustandsbasierten Spezifikation spezifiziert und als Zustandsübergangstabelle realisiert und instrumentiert wurde: automatisierte Unterstützung der Testauswertung durch Verwendung des zeitbehafteten endlichen Zustandsübergangsmodells als suboptimales Testorakel Prüfung, ob Prüfling nicht spezifizierte Zustände oder Zustandsübergänge enthält

Tabelle 5-5: Ergebnisse zu den Verfahren zum Test zustandsbasierter Systemen

5.3 Betrachtungen zur Formalisierbarkeit des funktionsorientierten Tests

Bei der Untersuchung der bekannten funktionsorientierten Testverfahren ist deutlich geworden, dass ihre vollständige formale Beschreibung im Allgemeinen nicht möglich ist. So benötigen die funktionsorientierten Testverfahren Informationen, die zunächst aus den meist informal beschriebenen funktionalen Anforderungen an den Prüfling herauszuarbeiten sind. Welche Information benötigt werden, wurde in den Voraussetzungen zu den Verfahrensbeschreibungen des Kapitels 4 herausgearbeitet: So waren Informationen über funktionale Partitionen oder Äquivalenzklassen anzugeben, die im Allgemeinen im Fließtext einer Spezifikation beschrieben sind. Ursache-Wirkungs-Zusammenhänge, die meist kausal in einer textuellen Beschreibung angegeben sind, wurden in einem Modell zusammengefasst. Auch für zustandsbasierte oder graphisch beschriebene Verhaltensbeschreibungen wurde eine formalisierte Beschreibung verlangt, die es ermöglichte, einen Bezug zwischen Strukturelementen der graphischen Beschreibungsform und dynamischen Testfällen herzustellen. Erst auf Basis dieser Informationen konnte eine formale Verfahrensbeschreibung erreicht werden. Hieraus wurde deutlich, in welchem Rahmen die Kreativität und Intuition des Testers im funktionsorientierten Test notwendig sind.

Die Formalisierbarkeit der Verfahren stellt somit das größte Problem bei der Beschreibung und Umsetzung des funktionsorientierten Tests dar, weist aber gleichzeitig auch auf seine eigentliche Stärke hin. So ist der funktionsorientierte Test mehr als alle anderen Testverfahren an der Spezifikation des Prüflings orientiert und damit an der ursprünglichen Idee, die der Entwicklung des Prüflings zugrunde liegt: Hier wird die Spezifikation nicht nur zur Prüfung des Testergebnisses herangezogen, sondern auch zur Ableitung der dynamischen Testfälle und zur Bewertung ihrer Vollständigkeit. Der funktionsorientierte Test ist damit mehr als alle anderen Testverfahren geeignet, die Umsetzung der ursprünglich informalen, der Entwicklung zugrunde liegenden Ideen im Prüfling zu begutachten.

Für die Erarbeitung, Bereitstellung und Nutzung dieser Informationen bestehen verschiedene Möglichkeiten im Softwareentwicklungsprozess: So ist es sinnvoll, bereits während der Beschreibung der Anforderung auf eine möglichst vollständige und gleichzeitig systematische Gestaltung der Spezifikation zu achten. Die Orientierung an den funktionsorientierten Verfahren unterstützt hierbei in vielfältiger Weise: Verfahren der funktionalen Partitionierung, der funktionalen Äquivalenzklassenanalyse oder der Grenzwertanalyse legen beispielsweise eine systematische Beschreibung der Spezifikation in Form von Wertetabellen über dem Eingabebereich mit Beschreibung des geforderten funktionalen Verhaltens und der Angabe der Grenz- und Toleranzbereiche nah. Die systematische Ursache-Wirkungs-Analyse fördert die vollständige Betrachtung der kausalen Zusammenhänge und regt durch die prädikative Interpretation der Ursachen und Wirkungen eine eindeutige Beschreibung der betrachteten Daten an. Die Überlegungen zu den graphischen Spezifikationsformen zeigen auf, dass für ihre eindeutige und unmissverständliche Interpretation die Assoziation konkreter Ein- und Ausgabedaten mit den Strukturelementen des Graphen möglich sein muss. Diese Überlegung regt beispielsweise eine möglichst vollständige Beschreibung der prädikativen Bedingungen für den Durchlauf von Kanten eines Aktivitätsdiagramms über dem Eingabebereich an. Die Vollständigkeitsbetrachtungen zu den zustandsbasierten Verfahren fördern die Beschreibung aller denkbaren Zustände und Ereignisse und ihre Berücksichtigung in der Spezifikation.

Wurden diese Betrachtungen bei der Erstellung der Spezifikation nicht angestellt, so dass diese das geforderte funktionale Verhalten nur unvollständig wiedergibt, kann eine Ergänzung der Angaben im

Rahmen der Entwicklung nachgezogen werden. Hierbei helfen die Verfahren des funktionsorientierten Tests bei Analyse und Systematisierung einer informal beschriebenen und unvollständigen Spezifikation. Funktionale Partitionen und Äquivalenzklassen können dann eine Grundlage der Implementierung bilden, der Grenzwertbegriff klärt Zugehörigkeiten zu funktionalen Äquivalenzklassen und mögliche Toleranzen, Ursachen und Wirkungen werden vollständig interpretiert und im Prüfling geeignet verknüpft, graphische Beschreibungsformen werden systematisch umgesetzt und zustandsbasierte Aspekte können systematisch ergänzt und als Grundlage einer gut strukturierten Implementierung verwendet werden.

Spät im Entwicklungsprozess unterstützen die Ansätze des funktionsorientierten Tests – entsprechend ihrer ursprünglichen Intention – bei der systematischen Qualitätssicherung. So leiten sie die gezielte Analyse der Anforderungen bei der Erstellung der Testspezifikation und unterstützen bei der Auswahl relevanter Testfälle. Sind die Angaben funktionaler Anforderungen in der Spezifikation unvollständig oder informal, können diese mit Hilfe der Verfahren analysiert und vervollständigt werden. Angewendet auf eine so vervollständigte Anforderungsdokumentation ermöglichen die Verfahren eine Bewertung der Testdatenadäquatheit orientiert an den funktionalen Anforderungen an den Prüfling.

5.4 Ausblick auf eine verstärkte Funktionsorientierung im Entwicklungsprozess

Die Überlegungen des vorangegangenen Abschnitts zeigen, dass eine funktionsorientierte Betrachtungsweise nicht nur in einem abschließenden Test gegen Anforderungen, sondern auch in anderen Phasen der Entwicklung sinnvoll unterstützen kann. Dies legt eine phasenübergreifende Unterstützung des Entwicklungsprozesses mit funktionsorientierten Ansätzen nahe.

Realisiert werden kann eine solche Hinwendung zur Funktionsorientierung in einer integrierten Entwicklungsumgebung, die auf Basis der Ergebnisse dieser Arbeit gestaltet werden kann. In dieser Entwicklungsumgebung können alle formalisierbaren Anteile der funktionsorientierten Testverfahren aus Kapitel 4 automatisiert unterstützt und in das in Kapitel 3 modellierte Vorgehen integriert werden. Die kreative Arbeit der Entwickler und Tester kann dann auf die nicht formalisierbaren Anteile konzentriert werden, so zum Beispiel auf die Modellbildung, die semantische Interpretation und die sukzessive Verfeinerung der Entwicklungsergebnisse. Konkret kann die Software- und Systementwicklung in einer solchen Entwicklungsumgebung wie folgt konzipiert werden:

- Zunächst sind die funktionalen Anforderungen in Fließtext aufzunehmen und anschließend mit geeigneten Beschreibungsformen zu modellieren und systematisch zu ergänzen. Dazu bieten sich die Beschreibungsformen des Kapitels 4 an, weitere Beschreibungs- und Modellierungsformen können nach Bedarf und Verfügbarkeit ergänzt werden.
- In einem nächsten Schritt sind auf Basis der funktionalen Anforderungen die Datenbereiche für die Ein- und Ausgabe zur ermitteln. Über diesen Datenbereichen ist eine eindeutige semantische Interpretation der Elemente der formalisierten Anforderungsbeschreibungen zu leisten. Hierzu ist das in Kapitel 4 zum jeweiligen Verfahren ausgearbeitete Vorgehen zu verwenden.
- Sobald die Parameterbereiche der Realisierung festgelegt und die Modelle und Beschreibungsformen semantisch interpretiert sind, können automatisiert im Rahmen der Entwicklungsumgebung Vollständigkeitsprüfungen der Anforderungen im Hinblick auf die Datenbereiche durchgeführt werden. So können Datenmengen identifiziert werden, die in den funktionalen Anforderungen noch nicht berücksichtigt wurden.

- Darüber hinaus kann die Entwicklungsumgebung, wie es bereits heute in gängigen anforderungsorientierten Entwicklungsumgebungen geleistet wird, bei der Verwaltung von Verknüpfungen zwischen textuellen und formalisierten Darstellungsformen der Anforderungen helfen. So wird die Verfeinerungen von Anforderungen beispielsweise im Design unterstützt.
- Ausgehend von den Anforderungen und ihren Verfeinerungen kann eine systematische analytische Qualitätssicherung nach Kriterien des funktionsorientierten Tests betrieben werden. Die Auswahl der Verfahren ist dabei durch den Tester zu treffen. Die Adäquatheit der gewählten Testdaten kann in der integrierten Entwicklungsumgebung mit Hilfe der semantischen Interpretation auf Basis der Modelle automatisiert ermittelt und bereits bei Definition der Testfälle bestimmt werden. Sie kann daher losgelöst von der eigentlichen Entwicklung oder der Durchführung der dynamischen Testfälle geschehen. Dabei werden nicht nur die abstrakten Verknüpfungen zwischen Anforderungen und Testfällen betrachtet, sondern konkret die Kriterien der in Kapitel 4 beschriebenen Verfahren.
- Für die Testdurchführung und –auswertung ergibt sich in einer solchen funktionsorientierten Entwicklungsumgebung ein weiterer zentraler Vorteil: So stehen formalisierte Informationen und Modelle der funktionsorientierten Beschreibungen bereit, die bei hinreichender Interpretierbarkeit über den Parameterbereichen als suboptimale Testorakel verwendet werden können. Die Auswertung des dynamischen Tests mit Hilfe von suboptimalen Testorakeln wurde in Abschnitt 3.2.3 dargestellt.
- Zusätzlich zu den funktionsorientierten Testverfahren können in eine solche Entwicklungsumgebung auch weitere Ansätze der analytischen Qualitätssicherung integriert werden. So wird in der Entwicklungsumgebung der Prüfling erstellt, der in der analytischen Qualitätssicherung einer kombinierten Betrachtung durch funktionsorientierte und strukturorientierte Verfahren unterzogen werden kann. Solche Verfahrenskombinationen haben eine sehr hohe Aussagekraft, wie in den Abschnitten 4.1.5.2, 4.2.2.2.3 und 4.4.4.3 dargelegt, und können unter geeigneten Umständen einem Korrektheitsbeweis gleichkommen.
- Darüber hinaus erleichtert die funktionsorientierte Beschreibung der Anforderungen mit Angabe einer semantischen Interpretation in der integrierten Entwicklungsumgebung eine Beschreibung stochastischer Testprofile über dem Eingabedatenbereich. Hierfür sind über den funktionalen Partitionen Häufigkeiten der Benutzung anzugeben. Die stochastischen Testprofile können dann im Rahmen der analytischen Qualitätssicherung einfach zur stochastischen Testdatengenerierung eingebunden werden. Zur automatisierten Auswertung der Testergebnisse, wie sie bei einer großen Zahl automatisiert erzeugter Testdaten unbedingt notwendig ist [MilEA92, How98], stehen die formalisierten Informationen und Modelle als suboptimale Testorakel bereit.
- Wird ein solcher automatisierter, an einem operationalen Nutzungsprofil orientierter Test im Rahmen der Entwicklung und sukzessiven Verbesserung des Prüflings begleitend angewendet, können mit Hilfe von Zuverlässigkeitswachstumsmodellen Zuverlässigkeitsaussagen abgeleitet werden [Mus80, MusIanOku90, Lyu95, ElbMäc02, ElbMäc04]. Diese können beispielsweise als Abnahmekriterien für Prüflinge mit moderaten Zuverlässigkeitsanforderungen eingesetzt oder zur Prognose von Wartungsaufwendungen verwendet werden.
- Wird hingegen die analytische Qualitätssicherung eines Prüflings mit hohen Sicherheitsanforderungen beispielsweise im Rahmen einer Zertifizierung durchgeführt, unterstützt der hohe Automatisierungsgrad in der integrierten Entwicklungsumgebung bei Generierung und Analyse der benötigten großen Zahl von Testfällen. Verwendet werden können hier sowohl operationale Nutzungs-

profile zur Generierung von Zuverlässigkeitsaussagen als auch inverse Profile, die ungewöhnliche Betriebssituationen simulieren und so die funktionale Sicherheit des Prüflings testen [Voa99]. Auch hier unterstützen suboptimale Testorakel die Auswertung, insbesondere wenn sie auf formalisierten Beschreibungen der Anforderungen an die funktionale Sicherheit beruhen.

Eine solche Entwicklungsumgebung verwendet eine funktionsorientierte Sicht als Ausgangspunkt der Konstruktion und des dynamischen Tests und ermöglicht darüber hinaus sinnvolle Kombinationen mit weiteren Techniken der analytischen Qualitätssicherung. Sie unterstützt damit sowohl den Formalisierungsprozess der Softwareentwicklung als auch die systematische Absicherung seiner Ergebnisse. Ganz konkret stellt sie eine sehr umfassende Möglichkeit zur konstruktiven Umsetzung der Ergebnisse dieser Arbeit dar. So wird abschließend die Hoffnung ausgesprochen, dass die hier vorgelegte Beschreibung des dynamischen Tests und die Erweiterungen zu den funktionsorientierten Testverfahren zu einer Anwendung in Wissenschaft und Praxis führen und dass so neue Impulse für eine Verbesserung der Softwarequalität gesetzt werden.

Verzeichnis der Definitionen

Verzeichnis der Verfahren

Verzeichnis der Abbildungen

Verzeichnis der Tabellen

Verzeichnis der verwendeten Symbole

Mathematische Logik

$\neg$	nicht
$\wedge$	und
$\vee$	oder
$\Rightarrow$	wenn – dann
$\Leftrightarrow$	genau dann – wenn
$\forall$	für alle
$\exists$	es gibt
$:=$	nach Definition gleich
$:\Leftrightarrow$	nach Definition äquivalent
$\perp$	ungültig, undefiniert

Mengenlehre

$\{ m_1, m_2, \dots \}$	Menge der Elemente m_1, m_2, …
$\{ m \in M: \dots \}$	Menge der Elemente aus M, für die … gilt
$\varnothing$	leere Menge
$\in$	Element von
$\subseteq$	enthalten in
$\subset$	echt enthalten in
$\supseteq$	umfasst
$M_1 \setminus M_2$	M_1 ohne M_2
$M_1 \cup M_2$	M_1 vereinigt mit M_2
$M_1 \cap M_2$	M_1 geschnitten mit M_2
$\bigcup_{k \in \{1,\dots,n\}} M_k$	Vereinigung aller M_k
$\bigcap_{k \in \{1,\dots,n\}} M_k$	Schnitt aller M_k
$\wp(M)$	Menge aller Teilmengen von M, Potenzmenge von M
$\wp^{fin}(M)$	Menge aller endlichen Teilmengen von M

Zahlensysteme

$\aleph$	Menge der natürlichen Zahlen
$\Re$	Menge der reellen Zahlen
$\Re^+$	Menge der positiven reellen Zahlen

Relationen und Strukturen

(m_1, m_2)	geordnetes Paar
$(m_1, m_2, \ldots, m_n)$	n-Tupel
$M_1 \times M_2$	kartesisches Produkt der Mengen M_1, M_2, Menge der geordneten Paare (m_1, m_2) mit $m_1 \in M_1$ und $m_2 \in M_2$
M^n	n-faches kartesisches Produkt der Menge M

Grammatiken

S*	Menge der Wörter, die durch Konkatenation von Symbolen der Menge S erzeugt werden können

Für weitere mathematische Definitionen sei auf die sehr systematische Darstellung in [ReiSoe01] verwiesen.

Verzeichnis der verwendeten Literatur

[AjmEA95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, John Wiley & Sons, 1995.

[Bac59] J. W. Backus, *The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference*, Proceedings of the International Conference on Information Processing, 1959.

[Bal98] H. Balzert, *Lehrbuch der Software-Technik, Band 2: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*, Spektrum Akademischer Verlag, 1998.

[Bal00] H. Balzert, *Lehrbuch der Software-Technik, Band 1: Software-Entwicklung*, Spektrum Akademischer Verlag, 2000.

[BasSel87] V. R. Basili, R. W. Selby, *Comparing the Effectiveness of Software Testing Strategies*, IEEE Transactions on Software Engineering, Vol. 13(12), pp. 1278—1296, December 1987.

[Bau96] B. Baumgarten, *Petri-Netze – Grundlagen und Anwendungen*, Spektrum Akademischer Verlag, 1996.

[Bei90] B. Beizer, *Software Testing Techniques*, Van Nostrand Reinhold, 1990.

[BerDia91] B. Berthomieu, M. Diaz, *Modelling and Verification of Time Dependent Systems Using Time Petri Nets*, IEEE Transactions on Software Engineering, Vol. 17(3), pp. 259 – 273, March 1991.

[BerGauMar91] G. Bernot, M.-C. Gaudel, B. Marre, *Software Testing based on Formal Specifications, a Theory and a Tool*, Software Engineering Journal, Vol. 6(6), pp. 387—405, November 1991.

[BerStr96] A. Bertolino, L. Strigini, *On the Use of Testability Measures for Dependability Assessment*, IEEE Transactions on Software Engineering, Vol. 22(2), pp. 97—108, February 1996.

[BerYuh93] L. Bernstein, C. Yuhas, *Testing Network Management Software*, Journal of Network and System Management, Vol. 1(1), pp. 5—15, January 1993.

[Bis90] P. G. Bishop, Ed., *Dependability for Critical Computer Systems 3: Techniques directory*, Elsevier Applied Science, 1990.

[Bit01] F. Bitsch, *Safety Patterns - the Key to Formal Specification of Safety Requirements*, Proceedings of 20th International Conference on Computer Safety, Reliability and Security (SAFECOMP), pp. 176—190, 2001.

[BooRumJac98] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.

[BroProPha92] R. Brownlie, J. Prouse, M. Phadke, *Robust Testing of AT&T PMX/StarMail Using OATS*, AT&T Technical Journal, Vol. 71(3), pp. 41—47, March 1992.

[ChiMil94] J. J. Chilenski, S. P. Miller, *Applicability of Modified Condition/Decision Coverage to Software Testing*, Software Engineering Journal, pp. 193—200, September 1994.

[Cho78] T. S. Chow, *Testing Software Design Modelled by Finite State Machines*, IEEE Transactions on Software Engineering, Vol. 4(3), pp. 178—187, May 1978.

[CiaGerLin94] G. Ciardo, R. German, C. Lindemann, *A Characterization of the Stochastic Process Underlying a Stochastic Petri Net*, IEEE Transactions on Software Engineering, Vol. 20(7), pp. 506—515, July 1994.

[ClaEA85] L. A. Clarke, A. Podgurski, D. J. Richardson, S. J. Zeil, *A Comparison of Data Flow Path Selection Criteria*, Proceedings of 8th IEEE International Conference on Software Engineering, pp. 244—251, 1985.

[ClaGruPel00] E. M. Clarke, O. Grumberg, D. A. Pelet, *Model Checking*, MIT-Press, 2000.

[ClaHasRic82] L. A. Clarke, J. Hassell, D. J. Richardson, *A Close Look at Domain Testing*, IEEE Transactions on Software Engineering, Vol. 8(4), pp. 380—390, July 1982.

[ClaWin96] E. M. Clarke, J. M. Wing, *Formal Methods: State of the Art and Future Directions*, ACM Computing Surveys, Vol. 28(4), pp. 626—643, December 1996.

[CobClaOst00] J. M. Cobleigh, L. A. Clarke, L. J. Osterweil*, Verifying Properties of Process Definitions*, Proceedings of the International Symposium on Software Testing and Analysis, pp. 96—101, 2000.

[CouCou79] P. Cousot, R. Cousot, *Systematic Design of Program Analysis Frameworks,* Proceedings of the ACM Symposium on Principles of Programming Languages, 1979.

[DauGauMar93] P. Dauchy, M.-C. Gaudel, B. Marre, *Using Algebraic Specifications in Software Testing: a Case Study on the Software of an Automatic Subway*, Journal of Systems and Software, Vol. 21(3), pp. 229—244, June 1993.

[DeM85] T. DeMarco, *Structured Analysis and System Specification*, Prentice Hall, 1985.

[DeMLipSay78] R. A. DeMillo, R. J. Lipton, F. G. Sayward, *Hints on Test Data Selection: Help for the Practicing Programmer*, IEEE Computer, Vol. 11(4), pp. 34—41, April 1978.

[DesEA97] J. Desel, A. Oberweis, A., T. Zimmer, G. Zimmermann, *Validation of Information System Models: Petri Nets and Test Case Generation*, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 3401—3406, 1997.

[Deu04] A. Deutsch, *Static Verification Of Dynamic Properties*, Internet Publication: http://www.polyspace.com/white_papers.htm, 2004.

[EN 50128 01] Europäisches Komitee für Elektrotechnische Normung, *Bahnanwendungen Telekomunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Software für Eisenbahnsteuerungs- und Überwachungssysteme*, EN 50128, 2001.

[DooFra94] R.-K. Doong, P. G. Frankl, *The ASTOOT Approach to Testing Object-Oriented Programs*, ACM Transactions on Software Engineering and Methodology, Vol. 3(2), pp.101—130, April 1994.

[DurNta84] J. W. Duran, S. C. Ntafos, *An Evaluation of Random Testing*, IEEE Transactions on Software Engineering, Vol. 10(4), pp. 438—444, 1984.

[Eme90] E. A. Emerson, *Temporal and Modal Logic*, Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, pp. 997—1072, Elsevier, 1990.

[ElbMäc02] B. Elbel, O.Mäckel, *Softwarezuverlässigkeit bewerten - Software Reliability Engineering als Bestandteil des Managements von Produktlebenszyklen*, QZ - Qualität und Zuverlässigkeit, pp. 915—918, September 2002.

[ElbMäc04] B. Elbel, O. Mäckel, *Efficient reliability growth modelling for industrial software failure data*, Proceedings of the 7th International Conference on Probabilistic Safety Assessment and Management, June 2004.

[Flo67] R. W. Floyd, *Assigning Meanings to Programs*, Proceedings of the American Mathematical Society, Symposium in Applied Mathematics, Vol. 19, pp. 19—32, 1976.

[FosOst76] L. D. Fosdick, L. J. Osterweil, *Data Flow Analysis in Software Reliability*, ACM Computing Surveys, Vol. 8(3), pp. 305—330, September 1976.

[FraEA98] P. G. Frankl, R. G. Hamlet, B. Littlewood, L. Strigini, *Evaluating Testing Methods by Delivered Reliability*, IEEE Transactions on Software Engineering, Vol. 24(8), pp. 586—601, August 1998.

[FraSie91] H. Franzen, G. Siegel, *Die Methode der strukturierten Analyse mit Petri-Netzen (SA/PN) als Echtzeiterweiterung*, Informatik-Fachberichte, Vol. 273, pp. 178—190, Springer-Verlag, 1991.

[FraWei93] P. G. Frankl, S. N. Weiss, *An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing*, IEEE Transactions on Software Engineering, Vol. 19(8), pp. 774—787, August 1993.

[FraWey88] P. G. Frankl, E. J. Weyuker, *An Applicable Family of Data Flow Testing Criteria*, IEEE Transactions on Software Engineering, Vol. 14(10), pp. 1483—1498, October 1988.

[FraWey93] P. G. Frankl, E. J. Weyuker, *A Formal Analysis of the Fault-Detecting Ability of Testing Methods*, IEEE Transactions on Software Engineering, Vol. 19(3), pp. 202—213, March 1993.

[GanMcMHam81] J. D. Gannon, P. R. McMullin, R. G. Hamlet, *Data-Abstraction, Implementation, Specification and Testing*, ACM Transactions on Programming Lan-

guages and Systems, Vol. 3(3), pp. 211—223, July 1981.

[GauJam99] M.-C. Gaudel, P. R. James, *Testing Algebraic Data Types und Processes: A Unifying Theory*, Formal Aspects of Computing, Vol. 10(5-6), pp. 436—451, 1999.

[GooGer75] J. B. Goodenough, S. L. Gerhart, *Toward a theory of test data selection*, IEEE Transactions of Software Engineering, Vol. 1(2), pp. 156—173, June 1975.

[Gou83] J. S. Gourlay, *A Mathematical Framework for the Investigation of Testing*, IEEE Transactions on Software Engineering, Vol. 9(6), pp. 686—709, November 1983.

[GroEA99] N. Gross, M. Stepaneck, O. Port, J. Carey, *Software Hell*, Business Week, pp. 104 – 118, December 6, 1999, Internet Publication: http://www.businessweek.com/datedtoc/1999/9949.htm.

[GroGriWeg93] M. Grochtmann, K. Grimm, J. Wegener, *Tool-Supported Test Case Design for Black-BoxTesting by Means of the Classification-Tree Editor,* Proceedings of EuroSTAR, 1993.

[Gut95] W. J. Gutjahr, *Optimal test distributions for software failure cost estimation*, IEEE Transactions on Software Engineering, Vol. 21(3), pp. 219—228, March 1995.

[Ham96] R. G. Hamlet, *Predicting dependability by testing,* ACM SIGSOFT Software Engineering Notes, Proceedings of the International Symposium on Software Testing and Analysis, pp. 84—91, 1996.

[HamTay90] R. G. Hamlet and R. Taylor, *Partition Testing does not Inspire Confidence*, IEEE Transactions on Software Engineering, Vol. 16(12), pp. 1402—1411, December 1990.

[HamVoa93] R. G. Hamlet and J. Voas, *Faults on its Sleeve: Amplifying Software Reliability*, ACM SIGSOFT Software Engineering Notes, Proceedings of the International Symposium on Software Testing and Analysis, pp. 89—98, 1993.

[Ham00] R. G. Hamlet, *On Subdomains: Testing, Profiles, and Components*, ACM SIGSOFT Software Engineering Notes, Proceedings of the International Symposium on Software Testing and Analysis, Vol. 25(5), pp. 71—76, 2000.

[Ham02] R. G. Hamlet*, Continuity in Software Systems*, ACM SIGSOFT Software Engineering Notes, Proceedings of the International Symposium on Software Testing and Analysis, Vol. 27(4), pp. 196—200, 2002.

[Har87] R. G. Harel, *Statecharts, A Visual Formalism For Complex Systems*, Science of Computer Programming, Vol. 8(3), pp. 231—274, June 1987.

[HarEA90] R. G. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, M. Trachtenbrot, *Statemate: a working Environment for the Development of*

Complex Reactive Systems, IEEE Transactions on Software Engineering, Vol. 16(4), pp. 403—414, April 1990.

[HasSne03] M. Hasitschka, H. M. Sneed, *Reestablishing Links between Test Cases and Concepts via Static and Dynamic Analysis*, Proceedings of GI-TAV Meeting, 2003.

[How75] W. E. Howden, *Methodology for the generation of program test data*, IEEE Transactions on Computers, Vol. C-24(5), pp. 554—560, May 1975.

[How76] W. E. Howden, *Reliability of the Path Testing Strategy*, IEEE Transactions on Software Engineering, Vol. 2(3), pp. 208—215, September 1976.

[How77] W. E. Howden, *Symbolic Testing and the DISSECT Symbolic Evaluation System*, IEEE Transactions on Software Engineering, Vol. 3(4), pp. 266—278, July 1977.

[How78] W. E. Howden, *A survey on dynamic analysis methods*, Software Testing and Validation Techniques (second edition), pp. 184-206, IEEE Computer Society Press, 1978.

[How80] W. E. Howden, *Functional program testing*, IEEE Transactions on Software Engineering, Vol. 6(2), pp. 162—169, March 1980.

[How82] W. E. Howden, *Weak mutation testing and completeness of test sets*, IEEE Transactions on Software Engineering, Vol. 8(4), pp. 371—379, July 1982.

[How86] W. E. Howden, *A functional approach to program testing and analysis*, IEEE Transactions on Software Engineering, 1986 Vol. 12(10), pp. 997—1005, October 1986.

[How98] W. E. Howden, *Good enough versus high assurance software testing and analysis methods*, Proceedings of the third IEEE International Symposium on High-Assurance Systems Engineering, pp. 166—175, 1998.

[HowHua95] W. E. Howden, Y. Huang, *Software Trustability Analysis*, ACM Transaction on Software Engineering and Methodology, Vol. 4(1), pp. 36—64, January 1995.

[IEC 61508 99] International Electrotechnical Commission, *Functional safety of electrical / electronical / programmable electronic safety-related systems*, CEI/IEC 61508, 1999.

[IEEE 830 98] Institute of Electrical and Electronics Engineers, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE Std 830-1998, IEEE Computer Society Press, 1998.

[ISO IEC 2382 93] International Organization for Standardisation / International Electrotechnical Commission, *Information Technology – Vocabulary*, 1993.

[ISO IEC 9126 01] International Organization for Standardisation / International Electrotechnical Commission, *Software Engineering – Product Quality*, ISO/IEC 9126:2001, 2001.

[JilDesMil01] L. J. Jilani, J. Desharnais, A. Mili, *Defining and Applying Measures of Dis-*

tance between Specifications, IEEE Transactions on Software Engineering, Vol. 27(8), pp. 673—703, August 2001.

[JonLanLie77] N. D. Jones, L. H. Landweber, Y. E. Lien, *Complexity of some Problems in Petri Nets*, Theoretical Computer Science, Vol. 4, pp. 277—299, 1977.

[Kor90] B. Korel, *Automated Software Test Data Generation*, IEEE Transactions on Software Engineering, Vol. 16(8), pp. 870—879, August 1990.

[Kos82] S. R. Kosaraju, *Decidability of Reachability in Vector Addition Systems*, Proceedings of 14th Annual ACM Symposium on Theory of Computing, pp. 267—281, 1982.

[Kra92] W. Kranz, *Die griechische Philosophie – Zugleich eine Einführung in die Philosophie überhaupt*, Sammlung Dieterich, 1992.

[LasKor83] J. W. Laski, B. Korel, *A Data Flow Oriented Program Testing Strategy*, IEEE Transactions on Software Engineering, Vol. 9(3), pp. 347—354, May 1983.

[LefWid99] D. Leffingwell, D. Widrig, *Managing Software Requirements: A Unified Approach*, Addison-Wesley, 1999.

[LehWeg00] E. Lehmann, J. Wegener, *Test Case Design by Means of the CTE XL*, Proceedings of EuroSTAR, 2000.

[LevSch06] V. Levering, O. Schömann, *Entwicklung von Hard-/Software-Komponenten mit neuem BMW-Anforderungsmanagement-Prozess - Roll-out bei der BMW Group und Lieferanten, Proceedings of REConf, 2006.*

[LevSto87] N. G. Leveson, J. L. Stolzy, *Safety Analysis using Petri Nets*, IEEE Transactions on Software Engineering Vol. 13(3), pp. 386–397, March 1987.

[Lig90] P. Liggesmeyer, *Modultest und Modulverifikation – State of the art*, BI Wissenschaftsverlag, 1990.

[Lig02] P. Liggesmeyer, *Software Qualität – Testen, Analysieren und Verifizieren von Software*, Spektrum Akademischer Verlag, 2002.

[Lig05] P. Liggesmeyer, in: *Deutschland schlampt bei der Softwarequalität*, Computer Zeitung, 35. Jahrgang, Nr. 21, p. 1, Mai 2005.

[LitEA00] B. Littlewood, P. T. Popov, L. Stringini, N. Shryane, *Modeling the Effects of Combining Diverse Software Fault Detection Techniques*, IEEE Transactions on Software Engineering, Vol. 26(12), pp. 1157—1167, December 2000.

[Lyu95] M. R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, 1995.

[ManKou01] L. I. Manolache, D. G. Kourie, *Software Testing using Model Programs*, Software – Practice and Experience, Vol. 31, pp. 1211—1236, 2001.

[ManPnu95] Z. Manna, A. Pnueli, *Temporal Verification of Reactive Systems: Safety*, Springer-Verlag, 1995.

[ManWal78] Z. Manna, R. Waldinger, *The logic of computer programming*, IEEE Transactions on Software Engineering, Vol. 4(3), pp. 199—229, May 1978.

[MeyGrüSpa03] D. Meyerhoff, H. Grün, R. von Spangenberg, *Von den Anforderungen zu den Testfällen - Wie optimiert man Kosten und Nutzen des Tests*, Proceedings of GI-TAV Meeting, 2003.

[MilEA92] K. W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nicol, B. W. Murrill, J. M. Voas, *Estimating the probability of failure when testing reveals no failures*, IEEE Transactions on Software Engineering, Vol. 18(1), pp. 33—43, January 1992.

[Mor90] L. J. Morell, *A theory of fault-based testing*, IEEE Transactions on Software Engineering, Vol. 16(8), pp. 844—857, August 1990.

[MorMur96] L. J. Morell, B. W. Murrill, *Using perturbation analysis to measure variation in the information content of test sets*, ACM SIGSOFT Software Engineering Notes, Proceedings of the 1996 International Symposium on Software Testing and Analysis, Vol. 21(3), pp. 92—97, 1996.

[Mus80] D. R. Musser, *Abstract Data Type specification un the AFFIRM System*, IEEE Transactions on Software Engineering, Vol. 6(1), pp. 24—32, January 1980.

[MusIanOku90] J. D. Musa, A. Iannino, K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, Professional Edition, McGraw-Hill, 1990.

[Mye76] G. J. Myers, *Software Reliability: Principles and Practices*, Wiley, 1976.

[Mye79] G. J. Myers, *The Art of Software Testing*, Wiley, 1979.

[Nel73] E. C. Nelson, *A statistical basis for software reliability assessment*, TRW-SS-73-03, March 1973.

[Nta01] S. C. Ntafos, *On Comparisons of Random, Partition, Proportional Partition Testing*, IEEE Transactions on Software Engineering, Vol. 27(10), pp. 949—960, October 2001.

[Oes98] B. Oestereich, *Objektorientierte Softwareentwicklung*, 4. Auflage, R. Oldenbourg Verlag, 1998.

[OffHay96] A. J. Offutt, J. H. Hayes, *A semantic model of program faults*, ACM SIGSOFT Software Engineering Notes, Proceedings of the 1996 International Symposium on Software Testing and Analysis, Vol. 21(3), pp. 195—200, 1996.

[OMG03] Object Management Group, *OMG Unified Modeling Language Specification*, Version 1.5, March 2003.

[Ost04] N. Oster, *Automatische Generierung optimaler datenflussorientierter Testdaten mittels evolutionärer Verfahren*, Proceedings of GI-TAV Meeting, 2004.

[OstEA96] L. J. Osterweil et al., *Strategic directions in software quality*, ACM Comput-

ing Surveys, Vol. 28(4), pp. 738—750, December 1996.

[Pet62] C. A. Petri, *Kommunikation mit Automaten*, Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM, 1962.

[Pet81] J. L. Peterson, *Petri Net Theory and the modelling of systems*, Prentice-Hall, 1981.

[PimRau76] S. Pimont, J. C. Rault, *A Software Reliability Assessment based on Structural Behaviour Analysis of Programs*, Proceedings of the second International Conference on Software Engineering, pp. 486—491, 1976.

[Pnu77] A. Pnueli, *The temporal logic of programs*, Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46—67, 1977.

[Ram00] S. Ramaswamy, *A Petri Net based Approach for Establishing necessary Software Design and Test Requirements*, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 4, pp. 3087—3092, 2000.

[RapWey85] S. Rapps, E. J. Weyuker, *Selecting Software Test Data Using Data Flow Information*, IEEE Transactions on Software Engineering, Vol. 11(4), pp. 367—375, April 1985.

[ReiSoe01] F. Reinhardt, H. Soeder, *dtv-Atlas Mathematik – Band 1 – Grundlagen, Algebra und Geometrie*, 12. Auflage, Deutscher Taschenbuch Verlag, 2001.

[RicCla81] D. J. Richardson, L. A. Clarke, *A Partition Analysis Method to Increase Program Reliability*, Proceedings of the fifth International Conference on Software Engineering, pp. 244—245, 1981.

[RicCla85] D. J. Richardson, L. A. Clarke, *Partition Analysis: A Method Combining Testing and Verification*, IEEE Transactions on Software Engineering, Vol. 11(2), pp.1477 – 1490, December 1985.

[RicTho88] D. Richardson, M. Thomas, *The RELAY Model of Error Detection and its Application*, in Proceedings of ACM SIGSOFT / IEEE second Workshop on Program Testing, Analysis and Verification, pp. 223—230, 1988.

[Rie97] E. H. Riedemann, *Testmethoden für sequentielle und nebenläufige Softwaresysteme*, Teubner, 1997.

[RotEA01] G. Rothermel, R. H. Untch, C. Chu, M. J. Harrold, *Prioritizing Test Cases For Regression Testing*, IEEE Transactions on Software Engineering, Vol. 27(10), pp. 929—948, October 2001.

[Rup02] C. Rupp, *Requirements-Engineering und -Management - Professionelle, iterative Anforderungsanalyse für die Praxis,* 2. Auflage, Hanser, 2002.

[Saa05] A. Saad, *Autos brauchen fehlertolerante IT-Systeme*, Interview: C. Schulzki-Haddouti, Computer Zeitung, 35. Jahrgang, Nr. 21, p. 22, Mai 2005.

[Sch01] U. Schöning, *Theoretische Informatik - kurzgefasst*, Spektrum Akademischer Verlag, 2001.

[Sch06] A. Schavan, *Rede der Bundesministerin für Bildung und Forschung anlässlich der Eröffnung des Wissenschaftsjahres 2006 – Informatikjahr –*, Internet Publication: http://www.informatikjahr.de/fileadmin/content/documents/Reden/Ministerin_Schavan_Rede_zum_Auftakt_Informatikjahr_060117.pdf, Berlin, 2006.

[SchKor00] Patrick J. Schröder, Bogdan Korel, *Black Box Test reduction using Input-Output Analysis*, ACM SIGSOFT Software Engineering Notes, Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, Vol. 25(5), pp. 173—177, 2000.

[Sne03] H. M. Sneed, *Automatic Extracting of Test Cases from a semiformal Specification*, Proceedings of EuroSTAR, 2003.

[Tai80] K. C. Tai, *Program Testing Complexity and Test Criteria*, IEEE Trancactions on Software Testing, Vol. 6(6), pp. 531—538, November 1980.

[TayLevKel92] R. N. Taylor, D. L. Levine, C. D. Kelly, *Structural Testing of Concurrent Programs*, IEEE Transactions on Software Engineering, Vol. 18(3), pp. 206 - 215, March 1992.

[ThaLipNel78] T. A. Thayer, M. Lipow, E. C. Nelson, *Software Reliability*, North-Holland, 1978.

[TurRob93] C. D. Turner, D. J. Robson, *The State-based Testing of Object-Oriented Programs*, Proceedings of the Conference on Software Maintenance, pp. 302—310, September 1993.

[VM97] Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik, *Entwicklungsstandard für IT-Systeme des Bundes, Vorgehensmodell*, Allgemeiner Umdruck Nr. 250/1, Bundesdruckerei Bonn, 1997.

[Voa92] J. M. Voas, *Pie: A Dynamic Failure Based Technique*, IEEE Transactions on Software Engineering, Vol. 18(8), pp. 717—727, August 1992.

[Voa99] J. M. Voas, *Certifying Software for High-Assurance Environments*, IEEE Software, Vol. 16(4), pp. 48—54, July/August 1999.

[VoaMil95] J. M. Voas, K. W. Miller, *Software Testability: The New Verfication. IEEE Software*, vol. 12, pp. 17—28, May 1995.

[WegBarSth01] J. Wegener, A. Baresel, H. Sthamer, *Evolutionary Test Environment for Automatic Structural Testing*, Information and Software Technology, Special Issue devoted to the Application of Metaheuristic Algorithms to Problems in Software Engineering, Vol. 43, pp. 841—854, 2001.

[WeiWey88] S. N. Weiss, E, J. Weyuker, *An extended domain-based model of software reliability*, IEEE Transactions on Software Engineering, Vol. 14(10), pp. 1512—1524, October 1988.

[Wey83] E. J. Weyuker, *Assessing Test Data Adequacy through Program Inference*, ACM Transactions on Programming, Languages and Systems, Vol. 5(4), pp. 641—655, 1983.

[Wey86] E. J. Weyuker, *Axiomatizing Software Test Data Adequacy*, IEEE Transactions on Software Engineering, Vol. 12(12), pp. 1128—1138, December 1986.

[WeyJen91] E. J. Weyuker, B. Jeng, *Analyzing Partition Testing Strategies*, IEEE Transactions on Software Engineering, Vol. 17(7), pp. 703—711, July 1991.

[WeyOst80] E. J. Weyuker, T. J. Ostrand, *Theories of Program Testing and the Application of Revealing Subdomains*, IEEE Transactions on Software Engineering, Vol. 6(3), pp. 236—246, May 1980.

[WhiCoh80] L. J. White, E. I. Cohen, *A Domain Strategy for Computer Program Testing*, IEEE Transactions on Software Engineering, Vol. 6(3), pp. 247—257, May 1980.

[WooHedHen80] M. R. Woodward, D. Hedley, M. A. Hennell, *Experience with Path Analysis and Testing of Programs,* IEEE Transactions on Software Engineering, Vol. 6(3), pp. 278—286, 1980.

Lebenslauf

Benedikte Antonie Elbel

geboren 24.06.1971 in Marburg (Hessen)

verheiratet, ein Kind

Berufliche Tätigkeit

03/2004 – dato	Methoden- und Prozessentwicklung, BMW Group
10/1999 – 02/2004	Softwarezuverlässigkeitsbewertung und Testen technischer Systeme, Corporate Technology, Siemens AG
09/1998 – 09/1999	Sensorik und Algorithmen zur passiven Sicherheit, BMW Group

Wissenschaftlicher Werdegang

03/1993 – 12/1998	Diplomstudiengang Mathematik, Nebenfächer Informatik und BWL, Technische Universität Darmstadt
09/1996 – 03/1997	Auslandsstudium an der École Normale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble
10/1995	Aufnahme in die Studienstiftung des deutschen Volkes

Berufsausbildung

09/1990 – 02/1993	Ausbildung zur mathematisch-technischen Assistentin, Deutsche Bundesbank, Frankfurt
02/1993 – 03/1993	Tätigkeit als mathematisch-technische Assistentin, Deutsche Bundesbank, Frankfurt

Mitgliedschaft in Berufsverbänden

Gesellschaft für Informatik	Aktive Mitarbeit in der Fachgruppe Test, Analyse und Verifikation, Arbeitskreis Testen von eingebetteten Systemen

Veröffentlichungen

- V. Alyokhin, B. Elbel, M. Rothfelder, A. Pretschner, *Coverage metrics for Continuous Function Charts*, Proceedings of the 15th International Symposium on Software Reliability Engineering, pp. 257—268, November 2004.
- B. Elbel, O. Mäckel, *Efficient reliability growth modelling for industrial software failure data*, Proceedings of the 7th International Conference on Probabilistic Safety Assessment and Management, June 2004.
- B. Elbel, O. Mäckel, *Softwarezuverlässigkeit effizient bewerten – Ein Erfahrungsbericht aus der Praxis*, Tagungsband zum 15. Workshop zu Testmethoden und Zuverlässigkeit von Schaltungen und Systemen, pp. 8—12, März 2003.
- B. Elbel, O.Mäckel, *Softwarezuverlässigkeit bewerten - Software Reliability Engineering als Bestandteil des Managements von Produktlebenszyklen*, QZ - Qualität und Zuverlässigkeit, pp. 915—918, September 2002.
- B. Elbel, G. Steidl, *Fast Fourier transforms for nonequispaced data*, Approximation Theory IX, C.K. Chui and L.L. Schumaker (Eds.), Vanderbuilt University Press, pp. 39—46, 1998.