KAPITEL 2

Grundlagen der Systemanalyse

Peter Ladkin

2.1 Einleitung: Über die Bedeutsamkeit des Schlussfolgerns

2.1.1 Der Vorrang des Schlussfolgerns bei Vorhersagen

Sowohl bei die Gewichtung als auch bei der Wahrung der Safety von Artefakten und Prozeduren geht es hauptsächlich um eine Vorhersage, was in der Zukunft einmal passieren könnte. Wenn man durch einen Blick in die Zukunft schon sicher wüsste, dass kein Unfall eintreten wird wüsste man auch, dass absolute Safety garantiert wäre. Da die Zukunft jedoch noch nicht passiert ist, kann man auf diese Weise keine Aussagen über Safety treffen. Man muss gegenwärtige und vergangene Situationen betrachten und daraus schlussfolgern, was in ähnlichen Situationen in der Zukunft passieren könnte. Man muss das existente Wissen so gut wie möglich verwenden, um die Zukunft so akkurat wie es geht vorherzusagen. Daher verwendet eine Abschätzung der Safety Schlussfolgerungen. Demnach verlangt eine genaue Abschätzung der Safety auch genaue Schlussfolgerungsprozesse. Die formale Logik ist die Lehre des genauen Schlussfolgerns. In gewissem Sinne muss eine strenge Einschätzung der Safety also angewandte formale Logik sein.

Einschränkungen in der Ausdruckskraft des Schlussfolgerns

Dennoch ist die formale Logik wie sie von Logikern sowie philosophischen Logikern genutzt und von Informatikern und Ingenieuren angewandt wird keine abgeschlossene Wissenschaft. Eigentlich gibt es nur relativ wenige „eingesessene“ Teilgebiete der formalen Logik: Das Aussagenkalkül, das Prädikatenkalkül, bestimmte so genannte „konstruktive“ Formulierungen dieser, bestimmte Formen der temporalen Logik, bestimmte Logik mit Modalitäten (Modallogik) und bestimmte Formen der Logik höherer Ordnung. Es gibt weitere wichtige Bestandteile des Schlussfolgerns im technischen Bereich, die entweder keine einheitliche Bezeichnung haben oder sich als unpassend für ihren einst erhofften Zweck herausgestellt haben. Diese sind: Die Arithmetik selbst, die Argumentation über Teil und Ganzes, die Argumentation über Verpflichtungen und die Argumentation über Kausalität.

Einschränkungen bei Berechnungsmöglichkeiten des Schlussfolgerns

Abgesehen von diesen Problemen stehen wir immer noch vor dem kleinen Problem, wie wir Schlussfolgerungen umgehen sollen. Schlussfolgerungen können einfach oder komplex sein. Sie können ganz leicht zu verstehen sein, oder aber auch nur für einige Auserwählte zugänglich sein. Es kann sehr schwierig sein eine Argumentationsstruktur aufzustellen, die für eine bestimmte Aussage zeigt ob diese durch andere Aussagen als kanonisch wahr, kanonisch widerlegt oder kanonisch unentschieden wird. Selbst wenn die Formulierung einer Schlussfolgerung nachweislich hinreichend ist, wird unsere Fähigkeit innerhalb einer solchen Formulierung Schlüsse ziehen zu können jedoch durch ihre kombinatorische Komplexität eingeschränkt.

Unsicherheit

Bei vielen wichtigen Dingen sind wir sehr unsicher. Wird dieses bestimmte Verbindungsstück dem konstanten Druck die nächsten zwei Jahre standhalten? Unsere Schlussfolgerungen sagen ja; die meisten haben in der Vergangenheit gehalten; speziell durchgeführte Tests sind positiv verlaufen; aber einige, sehr wenige konnten den Anforderungen nicht standhalten. Wir können also nicht absolut sicher sein, dass das Verbindungsstück halten wird. Wir können jedoch vermuten, dass es mit größter Wahrscheinlichkeit hält und würden auch eine größere Summe darauf verwetten, dass es tatsächlich so ist. Beschäftigt sich formale Logik mit unsicheren Überlegungen, wird

sie auch oft Wahrscheinlichkeitslogik genannt. Sie ist sehr schwierig, komplex und kann auch bei Entscheidungen nicht viel Unterstützung bieten. Formale schlussfolgernde Methoden für die Entscheidungsfindung sind auch kompliziert. Trotz alledem müssen wir sie dennoch verwenden.

Die Aufgabe der Ingenieure

Die Konstruktion von Sicherheitsnachweisen von Systemen oder auch die Beurteilung der Safety von Systemen sind Beispiele für überaus komplizierte Schlussfolgerungsstrukturen. Da Systeme gebaut und verwendet werden, muss man auf jegliche verfügbare Methode zurückgreifen, auch wenn sie noch so unvollständig ist. Auf der einen Seite war dieses Vorgehen in der Vergangenheit sehr erfolgreich. Komplizierte Verkehrsflugzeuge stürzen immerhin nicht jeden Tag ab. Auf der anderen Seite war man vielleicht doch nicht ganz so erfolgreich. Denn Verkehrsflugzeuge stürzen doch immer wieder aus den gleichen, sich wiederholenden und eigentlich einfach zu verhindernden Gründen ab. Der rasante Anstieg der Komplexität von Systemen verhindert den Einsatz von in der Vergangenheit erfolgreich eingesetzten Techniken zur Bestimmung der Safetyeigenschaften. Manchmal werden zuverlässige und sichere Konstruktionen durch neue ersetzt. Leider bleiben dabei aber auch manchmal Eigenschaften wie Zuverlässigkeit oder Safety (oder vielleicht sogar beide) auf der Strecke. Und dann gibt es natürlich vollständig neue Systeme, die auch neue Funktionen erfüllen. Es gibt also viel zu tun.

Praktische Anwendung

Das zu tun was man schon früher getan hat, wenn es denn funktioniert hat, kann ein guter Grundsatz sein. Das ist der Grund warum wir Standards und Checklisten befolgen. Das zu lassen was man früher getan hat kann dann ein guter Leitfaden sein, wenn es nicht funktioniert hat. Aus diesem Grund analysieren wir Unfälle. Außerdem ist es immer eine gute Idee, Fehler die man zu verhindern weiß nicht noch einmal zu machen. Darum ist es eine gute Angewohnheit Fehler im Design durch Kontrollen, Überprüfungen und generelles Verifizieren von Schlussfolgerungen zu unterbinden. Inspektionen und sachgerechte Wartungen sind eine gute Möglichkeit um die Konsequenzen einer unerwünschten Veränderung zu verhindern.

Voraussetzungen für ein effektives Schlussfolgern

Ein wichtiger Teil von Safety besteht aus Schlussfolgerungsprozessen über Systeme, und das schon in der Designphase. Um effektiv schlussfolgern zu können ist es üblich eine Sprache zu verwenden in der man Aussagen treffen kann, die dann irgendwie an die *„Welt“* und ihre Zustände gebunden werden; an die *„objektive Realität“* und an die Begriffe der *„Wahrheit“* und *„Falschheit“* von Aussagen im Bezug auf die *„aktuelle Situation“*. Mir ist kein anderer Weg bekannt. Erfolg im Ingenieurwesen basiert auf Techniken und Prozeduren, die von vielen erfahrenen Leuten – Ingenieuren – verwendet werden können. Im Bereich des Ingenieurswesens muss also auf die gleiche Art und Weise argumentiert und geschlussfolgert werden, wie schon seit hunderten von Jahren auf verschiedensten Gebieten argumentiert und geschlussfolgert wird. Logik, Ontologie, Objekte, Eigenschaften und Relationen, Aussagen über solche und deduktive Relationen sowie logische Schlussfolgerungen, Wahrscheinlichkeit und wahrscheinlichkeitstheoretische Argumentationen, die Verwendung einer formalen Sprache um sich Menschen vom Fach gegenüber unzweideutig ausdrücken zu können, Methoden um die Wahrheit und Falschheit oder auch die Wahrscheinlichkeit von Aussagen bestimmen zu können, und so weiter.

Unumgänglich: Strenge gefolgt von noch mehr Strenge

Es wird oft gesagt, dass man an alles denken muss wenn man Safety beurteilen und sicherstellen will. Dem bleibt eigentlich nur hinzuzufügen, dass man dies ganz besonders sorgfältig tun sollte. Situationen sollten gründlich überprüft werden. Das Streben nach strengen Vorgehensmethoden hat zu *Formalismen* geführt – ritualisierten Methoden des Sprechens und Argumentierens die durch Standardvorgehensweisen einfach zu reproduzieren sind und Schwächen und mögliche Probleme aufzeigen. Durch formale Sprachen können wir spezifizieren was wir überhaupt sagen können, und durch die Verwendung dieser Sprachen können wir genau beschreiben was wir sehen. So können wir ausgiebig verschiedene Möglichkeiten beschreiben und diese dann überprüfen. Wenn eine erschöpfende Spezifizierung zu mühsam ist können wir ausserdem indirekte Techniken verwenden. Im Prinzip können wir Fehler aufdecken, die uns in Äußerungen oder in der Argumentation unterlaufen. So können wir auch bemerken ob wir etwas vergessen haben und dies anschliessend korrigieren. Und all das können wir prinzipiengetreu machen, um Wiederholungen zu verhindern.

Strenge im Vorgehen

Wenn man sich mit Artefakten beschäftigt, kann Strenge auf zwei Arten angewandt werden. So kann sie bei der Argumentation über das Artefakt angewandt werden. Weiterhin kann sie bei der Handhabung des Artefakts sowie anderen menschlichen Verhaltensstrukturen in der Umgebung des Artefaktes angewendet werden. Und beides ist wichtig; beide Anwendungen gelten in der Tat als unverzichtbar in der modernen System Safety. Hier sollten wir uns jedoch auf die Anwendung des ersten Punktes beschränken: Die Argumentation über das Artefakt, sein Design, seine Eigenschaften und seine Umgebung.

2.2 Formale kausale Systemanalyse

Artifizielle Systeme werden von Menschen auf der Basis von kausalen Prinzipien gebaut. Die Bestandteile werden so entworfen, dass sie bestimmten Einfluss auf andere Bestandteile ausüben: Dieser Einfluss ist kausal. Die Analyse der Arbeitsvorgänge eines solchen Systems muss aus diesem Grund auch eine Form der kausalen Analyse sein. Die bisher verwendeten Methoden zur Analyse der Safetyeigenschaften sind zwar oft von ihrer Gestalt her kausale Analysen, jedoch fehlt ihnen meist eine genaue und strenge Auffassung von dem, was einen Grund oder einen Kausalfaktor definiert. Das bedeutet letztendlich, dass sie auf dem intuitiven Verständnis eines Grundes beruhen. Diese Intuition muss durch die Erfahrung und die Übereinstimmung in der Gemeinschaft der Ingenieure entstehen. Es bleibt jedoch die Tatsache, dass eine Methode, die auf undefinierten und nicht abgeklärten Auffassungen basiert nicht streng sein kann. Die Feuerprobe der Objektivität besteht zum einen in der Deutlichkeit der Beurteilungskriterien und zum anderen stellt sich da die Frage, ob prinzipiell auch Dritte diese anwenden können, wenn sie mit der in der Gesellschaft üblichen Intuition nicht vertraut sind und daher auch nicht auf diese zurückgreifen können um die Ergebnisse der Argumentationsprozesse zu überprüfen. So kann Intuition Prozesse beschleunigen, sollte aber nicht das Fundament bilden, gerade wenn Alternativen verfügbar sind.

Veranschaulichen wir einmal die Methoden zur Systemanalyse. Diese Methoden

- ermöglichen die kausale Analyse von Artefakten,
- basieren auf einer ausführlichen formalen Notation des Kausalfaktors, der, so

hat die Erfahrung gezeigt, in der Praxis mit minimalem Training relativ genau bestimmt werden kann,

- sind für die Safetyanalyse in der Entwicklungsphase gedacht: Causal System Analysis (CSA),
- sind auch für die kausale Analyse von Unfällen gedacht: Why-Because-Analysis (WBA).

2.3 Was ist ein System?

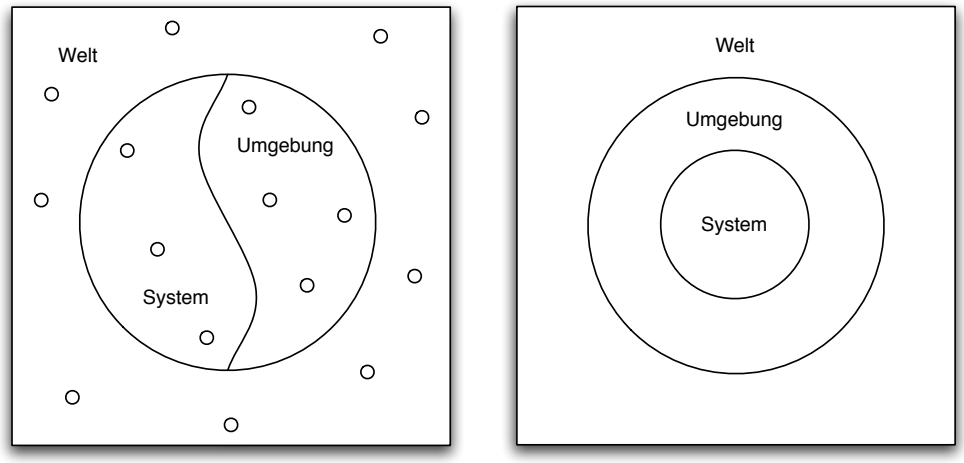
Beispiele

Wir bezeichnen verschiedene Dinge mit dem Wort System. Soziologen sprechen von sozialen Systemen [50]; Politiker von system-theoretischen Einflüssen [26]; andere Soziologen sprechen von komplexen technischen Systemen [58, 61]; Flugzeugkonstrukteure sprechen von technischen Systemen und Subsystemen; Ökologen sprechen von Räuber-Beute Systemen oder vom ökologischen Kreislauf der Substanzen; und natürlich gibt es noch die Computersysteme.

Was haben sie gemeinsam?

Ich behaupte, dass alle Systeme *Objekte* beinhalten welche an *Verhalten* teilnehmen. Dieses Verhalten, hervorgerufen durch die Objekten aus denen einem System besteht, kann beträchtlich durch das Verhalten von Objekten beeinflusst werden, die kein Bestandteil des Systems sind. Dies geschieht durch Relationen. Relationen zwischen den Objekten innerhalb eines Systems und Objekten außerhalb dieses Systems. Man sagt, die Objekte außerhalb des Systems gehören zur *Umgebung* des Systems. Dazu kommen dann noch weitere Objekte, die keinen erkennbaren Einfluss und keine wichtigen Relationen zu Systemobjekten haben und umgekehrt. Man sagt, dass diese Objekte zur *Welt* gehören.

Zum Beispiel, wenn ich mein Fahrrad mit all seinen Bestandteilen (inklusive Fahrer) als System ansehe, dann gehören die Straßen und Wege auf denen ich fahre genau so zur Umgebung wie die direkten Einflüsse auf ihre Zustände wie das Wetter oder ein über seine Ufer getretener Fluss. Da ich jedoch in Deutschland bin, gehört die Chinesische Mauer zur Welt, nicht zum System oder der Umgebung.



(a) Die Welt der Objekte

(b) Die Welt der Objekte wie sie sein sollte

Abbildung 2.1: Die Welt der Objekte

Eine Möglichkeit diese Aufteilung der Objekte darzustellen ist ein Venn-Diagramm, wie es in Abbildung 2.1a zu sehen ist. Die Objekte werden dort durch Punkte repräsentiert. So ein Diagramm kann jedoch missverständlich wirken, da die Objekte der Welt und die Objekte des Systems sich dort eine Grenze teilen. Diese Grenzteilung kann (irreführenderweise oder nicht) mit einer Relation gleichgesetzt werden. Objekte im System haben jedoch per Definition nur Relationen mit Objekten in ihrer Umgebung. Aus diesem Grund repräsentiert das Diagramm in Abbildung 2.1b besser, worauf wir hinaus wollen.

Die Systemgrenze

Obwohl das gesamte Universum als ein System (die Sammlung der Objekte = alles; Relationen und Eigenschaften = alle Relationen und Eigenschaften) betrachtet werden kann, beschäftigen sich Ingenieure meist mit anderen Systemen. Man berücksichtigt meistens kleinere Teile des Universums; und infolgedessen kann man auch eine Unterscheidung zwischen den Objekten treffen die zum System gehören und denen die dies nicht tun. Diese Unterscheidung führt zum Begriff der *Systemgrenze*, also

der Unterscheidung zwischen den Objekten und Verhalten die als Bestandteile des Systems gelten und denen, die das nicht tun. Die Systemgrenze kann in Anlehnung an etwas Reales gewählt werden, oder auch einfach nur eine Art Metapher sein. Wie die richtige Entscheidung aussieht, hängt sehr stark von der Art des Systems ab.

Teleologische und andere Systeme

Definieren wir ein *teleologisches System* als ein System mit einem bestimmten Zweck oder Ziel. Dieser Zweck kann ein bestimmtes Verhalten oder auch das Erlangen eines bestimmten Zustandes des Systems, seiner Umgebung oder beider sein (nicht jedoch der „Welt“, da die Bestandteile dieser per Definition außerhalb des gegenseitigen Einflusses zwischen System und Umgebungen liegen). Artefakte sind typische Beispiele für teleologische Systeme. Ein Auto wird mit dem Gedanken im Kopf entworfen, dass Menschen auf bestimmte Art und Weise transportiert werden sollen. Ein Computersystem wird entworfen, damit es bestimmte Arten von Berechnungen in einer bestimmten Art und Weise durchführt. Beispiele für nicht-teleologische Systeme sind Umweltsysteme wie ein industrieller Abflusskreislauf oder ein Räuber-Beute System. Das internationale politische System ist auch ein Beispiel für ein nicht-teleologisches System, obwohl die einzelnen Komponentensysteme, die Regierungen der Länder, teleologisch sind. Wir werden uns hauptsächlich mit teleologischen Systemen beschäftigen: Solche für die Finalzustände oder Zielverhalten des Systems und der Umgebung identifiziert werden können. Die Konstruktion teleologischer Systeme ist das primäre Ziel von Ingenieuren.

Die Grenzannahme bei teleologischen Systemen

Für ein teleologisches System sind die Ziele irgendwie festgelegt. Eine typische Methode wie die Grenze zwischen einem teleologischen System und seiner Umgebung festgelegt werden kann, wird durch die Überlegung geleitet auf welche Merkmale des *Universums* (die gesamte betrachtete Welt, inklusive des Systems und seiner Umgebung als möglicherweise dazugehörige Bestandteile) man mehr oder weniger Kontrolle ausüben kann, und auf welche nicht. Auf der Basis dieser Überlegung trifft man dann die Entscheidung. Man nennt die Annahme, dass mithilfe dieses Kontroll-Kriteriums die Entscheidung wie die Grenze zu setzen ist mehr oder weniger einheitlich getroffen werden kann, *die Grenzannahme*.

Beispiele für eine Grenze

Demnach kann der Zustand eines Rollfeldes in gewissem Maße kontrolliert werden: Fremdkörper können beseitigt werden, genau wie übermäßig viel Schnee oder Wasser auch. Der Flugverkehr kann umgeleitet werden bis diese Aufgabe erfüllt ist. Im Gegensatz dazu hat man auf die am Flughafen vorherrschende Wetterlage keinen oder nur wenig Einfluss. Daraus resultierend hat man auch keine Kontrolle über die variierenden Bedingungen der Luft durch die startende und landende Flugzeuge fliegen müssen. Vor diesem Hintergrund wäre es angemessen, den Zustand des Rollfeldes als Teil des Systems zu sehen, das Wetter jedoch nicht. Das (erwartete) Verhalten ändert sich dementsprechend. Obwohl man dazu verpflichtet ist zu warten bis sich schlechte Wetterkonditionen zugunsten sichererer Flugbedingungen verbessern, wäre ein unfähiger (oder hilfsbedürftiger) Flugfeldmanager am Werk, wenn dieser einfach warten würde bis der Schnee von seinem Rollfeld geschmolzen ist.

2.4 Objekte und Fluents

Was sind Objekte?

Objekte sind, grob gesagt, alles was mit einem Nomen oder einer Nominalphrase bezeichnet werden kann. Etwas allgemeiner ausgedrückt: Alles, was Wert eines Quantoren sein kann [59]. Also sind ich und du Objekte, die reellen Zahlen sind Objekte und auch das Wasser innerhalb einer festgelegten Begrenzung (z.B. durch drei gerade Linien zwischen geographisch festgelegten Punkten) ist ein Objekt. Eitelkeit hingegen scheint kein Objekt zu sein, genau so wenig wie Humor, Willenskraft oder auch der Wert einer bestimmten Speicherzelle eines Computers im Verlauf der Zeit, da sich dieser kontinuierlich verändert.

Fluents

Diese Quantitäten können als Objekte gesehen werden, wenn man bestimmte Operationen durchführt, die Logikern und Ontologen (Menschen die sich den Kopf darüber zerbrechen was überhaupt Objekte sind und welche Objekte es gibt) wohlbekannt sind. Daher wollen wir *Fluents* einführen als *Dinge, die über einen Zeitraum einen Wert annehmen*. Wenn wir den Begriff der Fluents akzeptieren, kann die Anzahl der Dinge

die wir als Objekte betrachten können um einiges steigen. Wenn wir binäre Zahlen als Objekte ansehen, dann ist der Wert einer Speicherstelle über einen Zeitraum ein Fluent, das Binärzahlen als Werte hat. Wenn wir Auswirkungen der Eitelkeit als Aktivität bestimmter menschlicher Hirnregionen sehen, können wir Freds Eitelkeit als eine Funktion über einen Zeitraum betrachten, die diese Aktivität in angemessener Art und Weise darstellt. Und natürlich können wir durch diese Fluents weitere Fluents definieren, die wiederum Fluents als Werte haben und iterativ so weiter. Kurz gesagt, mithilfe herkömmlicher Objekte und Fluents kann man bei Systemen über so viele Objekte sprechen wie man möchte.

2.5 Zustände, Ereignisse und Verhalten

Verhalten

Objekte haben Verhalten. Unter Verhalten verstehen wir die Veränderung von Eigenschaften eines Objektes und seiner Relationen zu anderen Objekten in einem bestimmten Zeitraum. Allgemeiner ausgedrückt: Verhalten ist die Veränderung einer Menge von Objekten (inklusive der Struktur der Menge selbst) über einen bestimmten Zeitraum.

Veränderung

Stellen wir uns eine vollständige Beschreibung von Objekten vor, mit all ihren Eigenschaften und Relationen untereinander. Zwar ist es aus vielen Gründen unmöglich eine solche Beschreibung zu erreichen, aber sehen wir darüber im Moment mal hinweg. Diese Beschreibung könnte zu einem bestimmten Zeitpunkt wahr sein. Einige Zeit später könnte sie genau so gut falsch sein. Das ist es, was wir als Veränderung über einen Zeitraum verstehen.

Zustand

Nennen wir eine solche vollständige Beschreibung von Objekten mit ihren Eigenschaften und Relationen untereinander zu einem bestimmten Zeitpunkt *Zustandsbeschreibung*, und die aktuelle Konfiguration von Objekten, Eigenschaften und Relationen die sie beschreibt einen *Zustand*. Abbildung 2.2 zeigt ein Beispiel für einen Systemzustand.

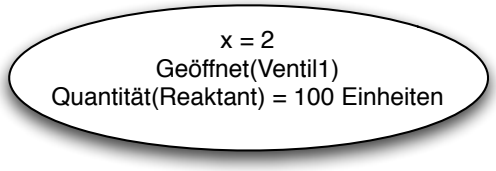


Abbildung 2.2: Ein Systemzustand

Das Fluent *x* hat den Wert 2; das Objekt *Ventil 1* hat die Eigenschaft *Geöffnet* (das verstehen wir so, dass *Ventil 1* geöffnet ist); die Menge an Reaktant (wir verstehen *Quantität*, in den nachfolgenden Grafiken auch als Q bezeichnet, als die Funktion, die uns als Wert die Menge ihres Arguments liefert; ihr Argument ist *Reaktant*) beträgt 100 Einheiten.

Eine Einschränkung: Aus technischen Gründen ist es wichtig, dass die Objekte, Eigenschaften und Relationen, die in einem Zustand angegeben sind, keinen direkten Bezug zu Zeit oder Veränderung beinhalten.

Begründung dieser Auffassung eines Zustands

Man kann sich durchaus fragen, warum wir diese anscheinend recht restriktive Auffassung eines Zustandes verwenden. Warum kann ein Zustand nicht alles Mögliche sein? Die Antwort auf diese Frage ist, dass diese Auffassung aus folgenden Gründen nicht so restriktiv ist wie sie anfangs vielleicht scheint:

- Niemand kann in die Zukunft schauen. Möchte man von einem Zustand genau festgelegte Aussagen erwarten können, darf man in einem gegenwärtigen Zustand keine Spekulationen über die Zukunft anstellen.
- Temporale Eigenschaften die die Vergangenheit betreffen können über das einfache Hilfsmittel der „Verlaufsvariablen“ eingebunden werden: Man führt ein Fluent ein, dass alle Informationen der Vergangenheit beinhaltet, die durch das System gespeichert werden müssen oder die man benötigt wenn man Aussagen über das System treffen möchte. Ein „Protokoll“ wenn man so will. In einer formalen Systembeschreibung ist die Verwendung von Verlaufsvariablen allgegenwärtig.

- Die Mengenlehre in der Prädikatenlogik erster Stufe (oder auch „Mengenlehre erster Stufe“ oder „Zermeki-Frankel Mengenlehre“ benannt nach zweier ihrer Erfinder) reicht aus, um die Komponenten-Struktur von Systemen zu beschreiben. Genau so bringt sie auch genug Mathematik mit um das diskrete oder analoge Verhalten eines Systems zu beschreiben. Einige Mathematiker und auch manche Informatiker mögen die Mengenlehre in der Prädikatenlogik aus verschiedenen Gründen nicht. Um dem Drang nach einer Vermeidung der Mengenlehre nachzukommen, kann man mithilfe von Typenlogik höherer Ordnung äquivalente Aussagen treffen. Wenn überhaupt, gibt es nur sehr wenige Eigenschaften, die sich nicht durch eine der beiden formalen Sprachen ausdrücken lassen.

2.6 Vergleichen von Zuständen

Haben wir zwei Zustände (oder zwei Zustandsbeschreibungen) gegeben, können wir sie miteinander vergleichen um die Unterschiede zwischen diesen festzustellen. Unter *Veränderung* verstehen wir daher einfach nur das was sich unterscheidet. Unter einer *Beschreibung der Veränderung* eine Spezifikation der Unterschiede (es kann ab und zu auch wichtig sein die Gleichheiten in einem Vergleich anzugeben, jedoch nicht immer). Abbildung 2.3 zeigt ein Beispiel für eine Veränderung anhand eines Vergleichs zweier Zustände. *Ventil 1* ist im ersten Zustand noch geschlossen, im zweiten jedoch geöffnet. Die anderen Prädikate haben sich nicht verändert: Das Fluent *x* hat immer noch den Wert 2, die Menge an Reaktant (ausgezeichnet durch das Prädikat Q mit dem Argument *reaktant*) beträgt immer noch 100 Einheiten.

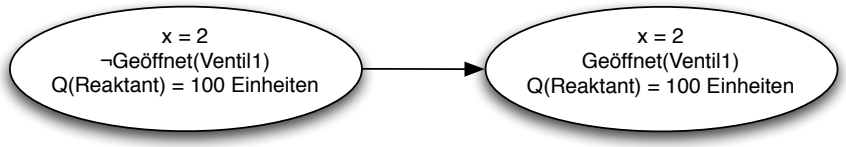


Abbildung 2.3: Eine Zustandsveränderung

Ereignisse und Ereignistypen

Die Veränderung in Abbildung 2.3 verrät über keinen der beiden Zustände besonders viel. Angenommen ich wäre ein Teil dieses gleichen Systems, zu dem auch die oben beschriebenen Objekte gehören. Dann würde nichts darüber ausgesagt welche Kleidung ich im Moment trage, wo ich mich momentan aufhalte oder wie spät es gerade ist. Möglicherweise beschreibt die Abbildung auch viele individuelle Veränderungen im System. Das wären genau die Veränderungen zu denen die beschriebenen Objekte in ihrem dargestellten Zustand passen. Nennen wir also eine bestimmte Veränderung, welche an einem bestimmten Zeitpunkt eintritt, ein *Ereignis*. Dann beschreibt die Zustandsveränderung in Abbildung 2.3 viele Ereignisse, nämliche genau die, in denen sich die Objekte wie beschrieben verändern. Wir sagen, sie beschreibt einen *Ereignistyp*. Ein Ereignis gehört genau dann zu diesem Ereignistyp, wenn es ein Ereignis ist in dem *x* 2 bleibt, *Q(reaktant)* unverändert auf 100 Einheiten steht, und *Ventil 1* sich von *nicht geöffnet* (geschlossen) auf *geöffnet* ändert.

Verhaltensbeschreibungen aus Zustandsvergleichen ableiten

Da wir ja, um zu sehen was sich verändert hat, zwei Zustände miteinander vergleichen können, können wir auch drei (einen nach dem anderen) miteinander vergleichen um eine schrittweise Beschreibung der Veränderungen zu erhalten. Oder vier, oder fünf, oder einhundert! Wir können Verhalten also als eine Sequenz solcher Zuständen betrachten, die eine Serie von Veränderungen beschreibt. Es ist wichtig zu bemerken, dass diese Sequenz *diskret* ist. Das bedeutet, jeder Zustand in dieser Abfolge hat einen bestimmten Vorgänger und einen bestimmten Nachfolger. Unter Umständen müssen wir sehr viele dieser Zustandsvergleiche aneinanderreihen, wenn wir eine hohe Detailtiefe benötigen oder sich die Sequenz über einen langen Zeitraum erstreckt. Aus diesem Grund kann eine Zustandssequenz sehr viele oder sogar unendlich viele Teilnehmerzustände haben. *Verhalten* sollten wir also als eine *unendliche, diskrete Sequenz von Zuständen* betrachten. Abbildung 2.4 zeigt eine solche diskrete Sequenz (oder wenigstens die ersten vier Zustände einer solchen).

Eine weitere Einschränkung: Aus technischen Gründen sollte eine Zustandssequenz, die ein Verhalten bildet, immer einen *Startzustand* besitzen – also einen Zustand der vor allen anderen auftritt und selbst keinen Vorgänger besitzt.

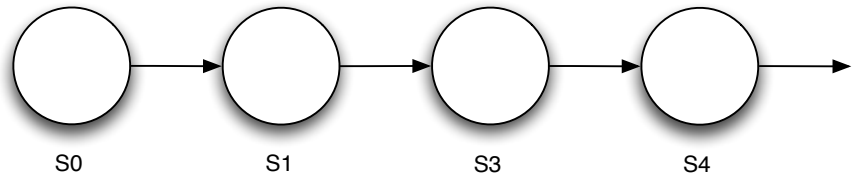


Abbildung 2.4: Ein Verhalten

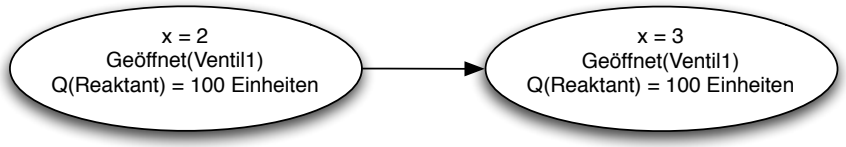


Abbildung 2.5: Eine nahe Veränderung

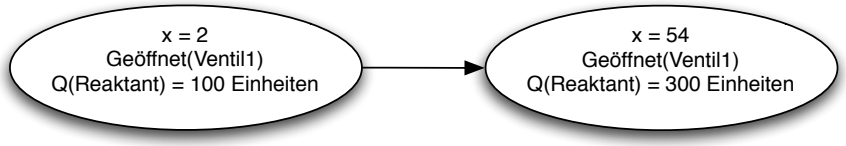


Abbildung 2.6: Eine entfernte Veränderung

Nahe und entfernte Zustandsveränderungen

Wir sollten noch geringe von umfangreichen Zustandsveränderungen unterscheiden. Geringe Änderungen werden wir *nahe Veränderungen*, und umfangreiche Veränderungen *entfernte Veränderungen* nennen. Diese Ausdrücke basieren auf ihrer intuitiven Bedeutung. Abbildung 2.5 zeigt zum Beispiel eine nahe Veränderung, in der der Wert des Fluents x sich von 2 auf 3 ändert, sonst jedoch alles gleich bleibt. Abbildung 2.6 zeigt eine entfernte Veränderung, in der sich der Wert des Fluents x beträchtlich auf 54 ändert und sich auch die Menge des Reaktanten verdreifacht.

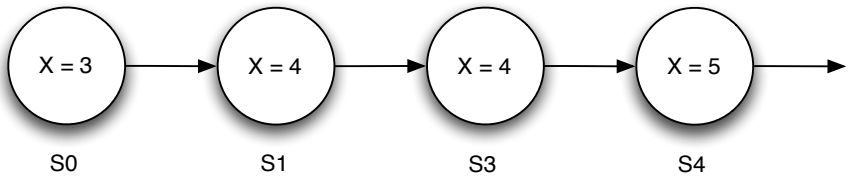


Abbildung 2.7: Eine Referenzverhalten

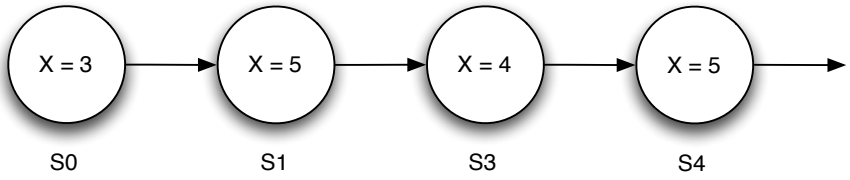


Abbildung 2.8: Ein dem Referenzverhalten nahes Verhalten

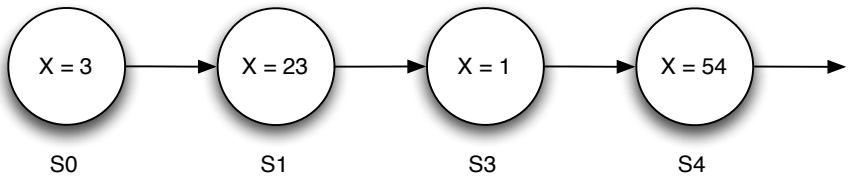


Abbildung 2.9: Ein dem Referenzverhalten weiter entferntes Verhalten

Nahe und entfernte Verhalten

Außerdem benötigen wir noch die Begriffe der nahen und entfernten Verhalten, die analog definiert sind. Dieser Begriff stellt einen Vergleich zwischen drei Verhalten an: Ein Verhalten *B* ist einem Verhalten *A näher* als ein Verhalten *C*. So ist das Verhalten in Abbildung 2.8 dem Referenzverhalten in Abbildung 2.7 zum Beispiel naher als das Verhalten in Abbildung 2.9.

Der einzige Unterschied zwischen dem Referenzverhalten in Abbildung 2.7 und seinem nahen Verhalten in Abbildung 2.8 liegt in dem Wert des Fluents x, der in

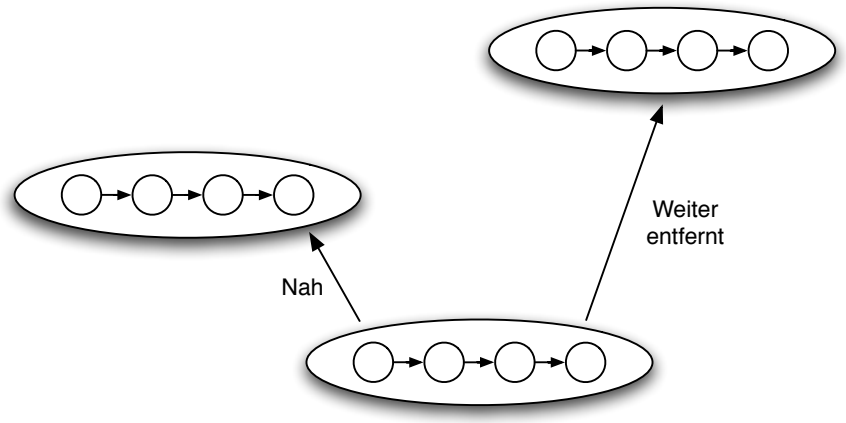


Abbildung 2.10: Nähere und entferntere Verhalten

Zustand *S1* des nahen Verhaltens 5 und nicht 4 ist.

Im Gegensatz dazu unterscheiden sich die Werte von *x* in den Zuständen *S1* und *S3* des entfernten Verhaltens sehr von den Werten von *x* in diesen Zuständen im Referenzverhalten. Auch der Wert von *x* im Zustand *S2* ist ein wenig anders.

Die „Entfernung“ zwischen Verhalten

Man stelle sich vor, wir möchten Verhalten als Punkte in einem Venn Diagramm visualisieren. Dann könnten wir die Abstände zwischen den Punkten im Diagramm dazu verwenden, die Nähe, und dementsprechend die Entferntheit, der Verhalten untereinander grafisch darzustellen. Genau das ist in Abbildung 2.10 zu sehen. Betrachten wir die „Entfernung“ aller möglichen Verhalten als Venn Diagramm, können wir damit, gesetzt den Fall wir hätten die Möglichkeit Nähe und Entferntheit auf einer *ordinalen Skala* [28] (siehe Abschnitt 8.2 für eine Beschreibung der genauen Eigenschaften) zu messen, die relative Nähe und Entferntheit aller denkbaren Verhalten zu einem gegebenen Verhalten, der *„realen Welt“*, darstellen. Dies ist in Abbildung 2.11 zu sehen.

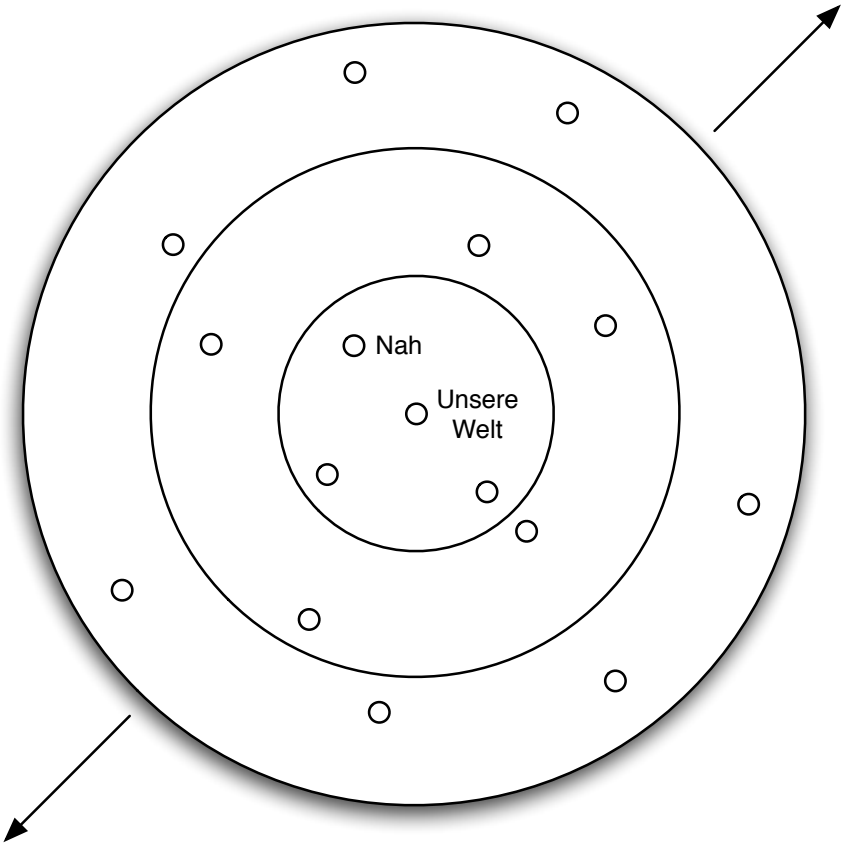


Abbildung 2.11: Alle Verhalten, angeordnet in „Entfernungskreisen“

Der Zweck dieser Definitionen

Der springende Punkt dieser Ontologie ist, dass

- es beweisbar vollständige Methoden zur formalen Schlussfolgerung über diese Strukturen gibt; und
- man jegliche Situation aus der „realen Welt“ mit diesen Strukturen angemessen beschreiben kann.

Da wir jegliche Situation die uns in den Kopf kommt beschreiben können, und wir aus dieser Beschreibung auf formale Weise präzise Schlüsse ziehen können, bietet es sich an diese Ontologie beim strengen Schlussfolgern über Systeme, wie es bei der

Safetyanalyse notwendig ist, zu verwenden.

2.7 Objekte, Teile und das Schließen auf Fehler

Objekte mit Teilen

Man stelle sich einen großen Brocken Computercode vor. Sagen wir ein Programm mit ein paar tausend Zeilen. Dieser Code *beinhaltet* Prozeduren und Befehle. Die Prozeduren sind *Teil* des Programms; die Befehle sind *Teil* der Prozeduren.

Strukturelle Teile

Die Befehle sind Teil der Prozeduren und die Prozeduren Teil des Programms. Aus einem anderen Blickwinkel ist das Programm aber nichts anderes als ein sehr langer String aus alphabetischen Symbolen. Jeder zusammenhängende Substring ist ebenfalls ein Teil des Programms, auch wenn er vielleicht in der Mitte eines Befehls beginnt und in der Mitte eines anderen Befehls aufhört. Möglicherweise ist also auch jegliche Sammlung von zusammenhängenden Substings Teil des Programms. Was wir sagen wollen ist, dass vollständige Befehle und vollständige Prozeduren die bedeutungstragenden Bestandteile eines Programms sind (wenn es um die Funktion des Programms geht). Abgesehen davon, dass die einzelnen Buchstaben und Strings Befehle und Prozeduren bilden, sind diese keine bedeutungstragenden Bestandteile eines Programms. Compiler treffen die Unterscheidung stillschweigend durch ihren Preprocessor, ein lexikalisches Analysewerkzeug, das Gruppen von einzelnen Zeichen zu *Tokens* zusammenfasst. Diese gelten als minimale bedeutungstragende Objekte wenn es um die Funktion des Programms geht. Zwischen diesen Fällen unterscheidet man dadurch, dass man die durch den Preprocessor (wenn dieser korrekt arbeitet) identifizierten Befehle, Prozedurteile und Tokens *strukturellen Teile* des Programms nennt, solange man die Funktion des Programms betrachtet. Betrachtet man den Speicherbedarf des Programms oder vergleicht man es mit der Arbeit von sechs Affen an Schreibmaschinen, können auch Buchstaben als strukturelle Teile gelten.

$$\frac{\begin{array}{l}\mathit{Versagt}(S)\\ \neg\mathit{Versagt}(\mathit{hardware}(S))\\ S = \mathit{hardware}(S) \oplus \mathit{software}(S)\end{array}}{\mathit{Versagt}(\mathit{software}(S))}$$

Abbildung 2.12: Korrektes Schliessen auf Fehler mit falscher Prämisse

Mereologie und Fusion

Es gibt ein Teilgebiet der Logik, die Mereologie, die sich damit beschäftigt welche Teile von Objekten existieren. Eine breit akzeptierte mereologische Operation ist die *Fusion*, bei der aus den Objekten X und Y das Objekt $X \oplus Y$ gebildet wird. Dieses neue Objekt nennt man die *'mereologische Summe'*, die X und Y als Bestandteile hat. Damit beinhaltet jedes Objekt, das sowohl X als auch Y als Bestandteile enthält, auch $X \oplus Y$: Mit anderen Worten, das 'kleinste' Objekt, das man aus X und Y bilden kann.

Teile und Fehler

Wenn ein System in seiner Funktion versagt, nimmt man oftmals eine Unterteilung in die Bestandteile vor um das Teil zu identifizieren, dass seine zugedachte Funktion nicht erfüllt hat. Schauen wir uns ein einfaches Beispiel an:

> „Das Rechnersystem hat versagt. Die Hardware hat nicht versagt. Aber das System besteht aus Software und Hardware. Also muss die Software versagt haben"

Die logische Oberflächenform dieses Schlusses ist in Abbildung 2.12 präsentiert. Unglücklicherweise ist die Schlussfolgerung falsch, obwohl sie durch Inferenz aus der Prämisse zu folgen scheint. Es gibt einige Beispiele in denen ein System versagt hat, doch weder die Hardware noch die Software ihre zugedachte Funktion verfehlt hat. So ein Beispiel ist der Ariane Flug 501. Dort gab es, wie es oft bei erfolgs- oder safetykritischen Systemen der Fall ist, eine Diskrepanz zwischen den Anforderungen an das Systemdesign, und der Umgebung in der das System tatsächlich eingesetzt wurde. Eine Subroutine in der Navigationshardware der Ariane 5 war von der Ariane 4 übernommen worden. Diese musste innerhalb bestimmter Variablenbereiche betrieben werden. Für den Flugbetrieb der Ariane 4 hatte sich herausgestellt, dass

$$\frac{\begin{array}{l}\mathit{Versagt}(S)\\ \neg\mathit{Versagt}(\mathit{hardware}(S))\\ S = \mathit{hardware}(S) \oplus \mathit{software}(S) \oplus \mathit{AnfSpez}(S)\end{array}}{\mathit{Entweder\ Versagt}(\mathit{software}(S))\mathit{oder\ Versagt}(\mathit{AnfSpez}(S))}$$

(*AnfSpez* steht hier für Anforderungsspezifikation)

Abbildung 2.13: Korrektes Schliessen auf Fehler

diese nicht überlaufen konnten. Warum auch immer unterschied sich die anfängliche Flugbahn der Ariane 5 aber von der der Ariane 4, und niemand hatte überprüft ob die Designannahmen für die Navigationsroutinen der Ariane 4 bei der Ariane 5 immer noch gültig waren. Sie waren es nicht. Eine Variable lief über, und löste damit eine Kettenreaktion aus, die zum Verlust der Kontrolle und Zerstörung der Raumfähre führte.

Wenn die ersten beiden Prämissen wahr sind, die Konklusion jedoch falsch, dann muss entweder die dritte Prämisse falsch sein, oder die Schlussfolgerung ist ungültig. Am Ariane Beispiel (und anderen die ich hier nicht zitiert habe) sehen wir, dass die Schlussfolgerung in Abbildung 2.13 angebrachter ist.

Die Aufgabe der Fusion beim Schliessen auf Fehler

In der Schlussfolgerung in Abbildung 2.12 ist die Rolle der Fusion der Prämisse deutlich zu erkennen:

> *Aber das System besteht aus Software und Hardware.*

Der Schluss, dass die Software in ihrer Funktion versagt haben muss weil die Hardware korrekt funktioniert hat, ist ein Trugschluss. In der ansonsten ähnlichen Prämisse der Schlussfolgerung in Abbildung 2.13 wird die Aufgabe der Prämisse deutlich:

> *Aber das System besteht aus Software, Hardware und seiner Anforderungsspezifikation.*

Im Gegensatz zur Schlussfolgerung in Abbildung 2.12 ist die Schlussfolgerung in Abbildung 2.13, dass entweder die Software oder die Anforderungsspezifikation fehlerhaft war da die Hardware fehlerfrei funktioniert hat, korrekt.

$$\frac{\begin{array}{l}\mathit{Fehlerhaft}(S)\\ S = \mathit{Teil}_1(S) \oplus \mathit{Teil}_2(S) \oplus \mathit{Teil}_3(S)\\ \neg\mathit{Fehlerhaft}(\mathit{Teil}_1(S))\\ \neg\mathit{Fehlerhaft}(\mathit{Teil}_2(S))\end{array}}{\mathit{Fehlerhaft}(\mathit{Teil}_3(S))}$$

Abbildung 2.14: Korrektes Schliessen auf Fehler, 2

Der Unterschied zwischen den beiden Fällen ist einerseits, dass die Prämisse Fusion beinhaltet. Andererseits die Inkorrektheit, und dementsprechend Korrektheit, der Schlussfolgerung. Ich folgere also: Wenn es um das korrekte Schließen auf Fehler geht, ist das korrekte Erfassen der Fusionsprämisse ein wichtiger Bestandteil des Schlussfolgerns.

Es scheint also wünschenswert zu sein so zu schlussfolgern wie es in Abbildung 2.14 dargestellt ist. Es wird schnell deutlich, dass diese Art der Schlussfolgerung in viele Fehleranalysemethoden einfließt (Softwaremenschen nennen das 'debugging'). Das *Software(S)* $\oplus$ *Hardware(S)* Beispiel von eben zeigt jedoch, dass man sehr aufmerksam sein muss um sicherzustellen, dass die Bestandteile, die man einem System zuordnet, auch tatsächlich alle Bestandteile sind aus denen das System zusammensetzt ist.

Ist die „Dokumentation" Teil des Systems?

Der einzige Unterschied zwischen der Schlussfolgerung mit dem falschen Schluss und der Schlussfolgerung mit dem korrekten Schluss liegt in den verschiedenen Bestandteilen bei der Fusion in der Prämisse. In dem Ariane Beispiel lag der Fehler eigentlich in der Spezifikation der Anforderungen und der Verifikation ob diese mit den Anforderungen tatsächlich übereinstimmt. Unsere Lösung, die Spezifikation in die Fusion einfließen zu lassen, führt jedoch zu einer etwas unintuitiven Idee. Diese ist, dass die Anforderungsspezifikation, die auch die Betriebsbedingungen des Systems beinhalten sollte, eigentlich ein Teil des Systems ist. Dieses sähe dann so aus:

$$S = \mathit{software(S)} \oplus \mathit{hardware(S)} \oplus \mathit{requirements(S)}$$

Es mag durchaus merkwürdig erscheinen, eine Spezifikation, die ja nur ein Stück Text ist, mit den physikalischen Komponenten eines Systems zusammenzufassen.

Aber so ungewöhnlich ist das nicht: Der Systemcode wird als Bestandteil des Systems angesehen, und Code ist auch nur Text. Wo liegt dann der Unterschied zwischen der Anforderungsspezifikation und dem Systemcode? Der Systemcode kann auch als Spezifikation gesehen werden; das System soll sich entsprechend diesem und jenem Befehl verhalten.

Angemessene Dekompositionen

Man könnte die Dekomposition eines Systems in seine Einzelteile als *angemessene Dekomposition* bezeichnen, wenn

a) das System die Fusion der vorgeschlagenen Einzelteile ist, und

b) beim Versagen des Systems mindestens eins der Einzelteile versagt hat.

Durch eine angemessenen Dekomposition ist man in der Lage, Fehler wie in Abbildung 2.14 zu erschließen. Das diskutierte Beispiel zeigt, wie einige andere auch, dass die meisten der üblichen technischen Dekompositionen von Systemen in ihre Bestandteile keine angemessenen Dekompositionen sind.

Systemunfälle und die DEPOSE Dekomposition

Der Unfall-Soziologe Charles Perrow hat angemerkt, dass *‚interaktiv komplexe‘* Systeme, die *‚eng miteinander gekoppelt‘* sind einen Hang zu *‚Systemunfällen‘* haben [58] . Diese seien Unfälle die zwar durch das System verursacht würden, aber nicht auf Versagen oder Fehlverhalten einer ihrer Komponenten zurückzuführen wären. Wenn diese Unfälle als Fehler des Systems angesehen werden, würde Perrow aufgrund der Behauptung oben ja sagen, dass für „interaktiv komplexe“ und „eng gekoppelte“ Systeme keine angemessene Dekomposition existieren kann. Natürlich gibt es für diese Behauptung keinen Beweis, und es ist auch schwer vorstellbar wie es dafür einen Beweis geben könnte. Stattdessen könnte Perrow aber gemeint haben, dass eine durch Menschenhand erstellte Dekomposition eines interaktiv komplexen und eng gekoppelten Systems wahrscheinlich keine angemessene Dekomposition darstellt. Er selbst hat ein Schema namens DEPOSE vorgestellt, das es ermöglichen soll komplexe Systeme in die Klassen ihrer Komponenten einzuordnen.

DEPOSE steht für:

- Design

- Equipment
- Procedures
- Operators
- Supplies and Materials
- Environment

Perrows Klassifikation betont wesentliche Eigenschaften, wie das Design der Prozeduren und das Training und Verhalten der menschlichen Systembenutzer, die traditionell nicht immer mit der gleichen Sorgfalt untersucht worden sind wie die technischen Komponenten. Er gibt jedoch keinen Beweisgrund an aus dem hervorgeht, dass:

- DEPOSE für jegliches komplexe System T eine angemessene Dekomposition liefert, nämlich

  $$T = D_T \oplus E_T \oplus P_T \oplus O_T \oplus S_T \oplus E_T$$

  oder, dass
- DEPOSE eine gründlichere Klassifizierung von Fehlerkategorien ermöglicht, als es die traditionellen Untersuchungsmethoden in den jeweiligen Industriezweigen gewährleisten können.

Dennoch hat Perrows Arbeit wichtige Impulse für die Untersuchung von menschlichen Fehlermöglichkeiten und das Design von Prozeduren bei der Konstruktion komplexer Systeme gegeben.

Übliche Dekompositionen in Komponententypen

Für bestimmte Klassen von Systemen können angemessene Dekompositionen existieren. Eine Erörterung einiger der üblichen oder sinnvollen Klassifikationen komplexer Systeme in Komponententypen kann in [31] gefunden werden.