Ein methodischer Übergang von asynchron zu synchron kommunizierenden Systemen

Bernhard Schätz

Institut für Informatik
der Technischen Universität München

Ein methodischer Übergang von asynchron zu synchron kommunizierenden Systemen

Bernhard Schätz

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Manfred Paul
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Manfred Broy
2.	Univ.-Prof. Dr. Dr. h.c. Wilfred Brauer

Die Dissertation wurde am 24. Juni 1998 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 31. August 1998 angenommen.

Zusammenfassung

Bei der Modellierung reaktiver, verteilter, kommunizierender Systeme werden hinsichtlich der Art der Kommunikation zwei wesentliche Paradigmen unterschieden: die asynchron oder unbeschränkt gepuffert kommunizierenden Systeme sowie die synchron oder ungepuffert kommunizierenden Systeme. Für beide Sichtweisen wurden unterschiedliche formale Modellierungen entwickelt; beide fanden ihren Niederschlag in unterschiedlichen Sprachen zur Programmierung verteilter Systeme; beide wurden Grundlage unterschiedlicher Vorgehensweisen zur Entwicklung verteilter Systeme. Dabei wurde wiederholt festgestellt, daß sich die asynchrone Sichtweise eher an abstrakteren Systemmodellen orientiert, während die synchrone Sichtweise stärker zur Implementierungsnähe neigt.

Es liegt somit nahe, die asynchrone Sichtweise als Ausgangspunkt einer Entwicklungsmethode zu wählen, und erst im Laufe der Entwicklung eines Systems dieses auf die synchrone Sichtweise hin zu verfeinern. Ziel dieser Arbeit ist es, zu untersuchen, wie beide Sichtweisen vereinheitlicht werden können, um einen methodischen Rahmen für die schrittweise Entwicklung synchron kommunizierender verteilter Systeme zur Verfügung zu stellen. Es werden zwei geeignete Semantiken vorgestellt, ein Übergang zwischen diesen beiden Formalismen eingeführt und ein methodisches Verfahren zur Behandlung der Synchronisierung entwickelt. Der Schwerpunkt der Arbeit liegt auf der Durchgängigkeit und der Methode einer solchen Vorgehensweise zur Entwicklung verteilter Systeme.

Danksagung

Die vorliegende Arbeit entstand durch vielfältige Unterstützung. Darum ist es mir eine Freude, den folgenden Personen zu danken:

Für den Anstoß zum wissenschaftlichen Arbeiten und die Unterstützung beim Abschluß der Arbeit bedanke ich mich bei Wilfried Brauer.

Für die Ermutigung zur Beschäftigung mit dem Themengebiet der Arbeit, die Hilfe bei der Themenfindung und die Unterstützung bei der Durchführung bedanke ich mich bei Manfred Broy. Besonderer Dank gilt ihm jedoch für seine Geduld sowie für seine Freizügigkeit, mich den von mir gewählten Ansatz verfolgen zu lassen.

Für die Schaffung der Rahmenbedingungen und die Ermöglichung des Promotionsstipendiums der Siemens AG bedanke ich mich bei Peter Graubmann und Carsten Hammer.

Für zahlreiche Diskussionen bedanke ich mich bei Katharina Spies, Franz Regensburger und Oscar Slotosch. Besonderer Dank gilt Katharina Spies und Ursula Hinkel sowie Oscar Slotosch für die sorgfältige Durchsicht und die hilfreichen Kommentare zu dieser Arbeit.

Nicht minder wichtig für das Entstehen dieser Arbeit war die geduldige Nachsicht und Aufmunterung meiner Familie, Freunde und Kollegen. Auch ihnen gilt daher mein besonderer Dank für die Unterstützung bei der Anfertigung dieser Arbeit.

Though this be madness, yet there is method in't.
[211]
Hamlet, William Shakespeare 1564-1616

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Synchrone und asynchron Systeme 2

1.2 Motivation 4

1.2.1 Schrittweise Systementwicklung 4

1.2.2 Schrittweise Entwicklung nachrichtenorientierter Systeme 4

1.2.3 Ziele der Arbeit 5

1.2.4 Umfeld der Arbeit 6

1.2.5 Verwandte Arbeiten 8

1.3 Übersicht 8

2 Modelle verteilter Systeme **11**

2.1 Modellierung verteilter Systeme 12

2.1.1 Auswahl der Modelle 12

2.1.2 Kompositionsoperatoren 13

2.2 Spursemantik 14

2.2.1 Grundbegriffe des Spurmodells 15

2.2.2 Spurbeschreibung asynchroner Systeme 17

2.2.3 Kompositionsoperatoren 18

2.2.4 Beispiel 19

2.2.5 Spezielle Formen der Spurdarstellung 20

2.3 Failuresemantik 32

2.3.1 Kompositionsoperatoren 33

2.3.2 Eigenschaften der Failuresemantik 34

2.3.3 Beispiel 35

2.4 sat-Kalkül 36
2.4.1 sat-Relation 36
2.4.2 Spurregeln 37
2.4.3 Failureregeln 39
2.5 Vergleich der Modelle 40
2.5.1 Informelle Beschreibung 40
2.5.2 Formale Beschreibung 41
2.6 Systementwurf auf der Spurebene 42
2.6.1 Partielle und vollständige Eigenschaften 42
2.6.2 Parallelität auf der Spurebene 43
2.6.3 Beschreibung verteilter Systeme 47
2.7 Zusammenfassung 49

3 Modellwechsel **51**
3.1 Unendliche Failuresemantik 52
3.1.1 Unverträglichkeit der Failuresemantik 53
3.1.2 Modell 54
3.1.3 Intuitive Bedeutung 54
3.1.4 Anpassung 55
3.2 Transformation 57
3.2.1 Verträglichkeit der partiellen Eigenschaften 58
3.2.2 Implizierte Failureeigenschaften 60
3.2.3 Verträglichkeit der Transformation 61
3.3 Kombinierte Systeme 66
3.3.1 Aktionsweise Parallelität der Eingabe 66
3.3.2 Weitere Form der Abschwächung 68
3.4 Transformation sequentieller Prozesse 69
3.4.1 Failuredarstellung monotoner Prozesse 70
3.4.2 Failuredarstellung unterbrechbarer Prozesse 74
3.5 Methodischer Einsatz der Transformation 76
3.5.1 Entwurf auf der Spurebene 77

3.5.2 Wechsel zur Failureebene 78
3.6 Abstraktion und Verfeinerung 80
3.6.1 Verfeinerung und Unterspezifikation 80
3.6.2 Asynchronität als Abstraktion 81
3.6.3 Asynchronität als methodischer Aspekt 83
3.7 Abschwächung der Systemanforderung 86
3.7.1 Abschwächung 87
3.7.2 Beispiel 89
3.8 Verschärfung der Anforderungen 89
3.8.1 Nichtblockierendes Senden 90
3.8.2 Allgemeine Verfeinerungsschritte 92
3.9 Zusammenfassung 92

4 Effiziente Realisierung **93**
4.1 Residuen 94
4.2 Residuen verteilter Systeme 97
4.2.1 Residuen im Entwicklungsprozeß 97
4.2.2 Residuen und Kompositionsoperatoren 99
4.2.3 Notwendigkeit des unendlichen Modells 102
4.3 Vorgehensweise 104
4.3.1 Aufbauorientierte Verfahrensweise 105
4.3.2 Residuum für Prozeßteile 109
4.3.3 Beispiel 110
4.4 Direkte Residuendarstellung 113
4.4.1 Abstraktion 113
4.4.2 Parallelkomposition 114
4.5 Kanalorientierte Realisierung 117
4.5.1 Modellierung von Kanälen 118
4.5.2 Kanalzuordnung 120
4.5.3 Prozeßrealisierung 122
4.6 Beispiel 124
4.7 Zusammenfassung 133

5 Zusammenfassung und Ausblick **135**

5.1 Zusammenfassung 136

5.2 Ausblick 137

5.2.1 Dynamische Netze 137

5.2.2 Parallelitätsgrad 138

5.2.3 Maschinelle Unterstützung 138

A Unendliche Failuresemantik **141**

A.1 Einleitung 142

A.2 Divergenz 143

A.2.1 Endliche Failuresemantik 143

A.2.2 Failure-Divergence-Semantik 147

A.2.3 Unzulänglichkeit der Divergenz 147

A.3 Unendliche Failuresemantik 149

A.3.1 Bereich 150

A.3.2 Semantik 151

A.4 Beispiele 152

A.4.1 Sequentielle Prozesse 153

A.4.2 Parallele Prozesse 155

A.4.3 Deutung der Ergebnisse 156

A.5 Unterbrechung 158

A.5.1 Prozeßoperator 159

A.5.2 Beispiel 160

A.6 Zusammenfassung und Ausblick 161

B Beweise **163**

B.1 Systemstrukturen 164

B.2 Erweitertes semantische Modell 167

B.2.1 Komplexes Failuremodell 168

B.2.2 Komplexe Failuresemantik 170

B.2.3 Nachweise der Abschlußeigenschaften 171

B.2.4 Abstraktion zur unendlichen Failuresemantik 219

B.2.5 Komplexes und unendliches Failuremodell 222
B.3 Terminierbarkeit 223
B.4 Eigenschaften der unendlichen Failuresemantik 228
B.4.1 Eigenschaften von $P\backslash H$ 228
B.4.2 Weitere Eigenschaften 231
B.5 Lemmata 240
B.5.1 Algebraische Aussagen über Prozesse 240
B.5.2 Hilfsaussagen über endliche Mengen 244
B.5.3 Hilfsaussagen über Ketten 245
B.5.4 Verflechten von Spuren 251
B.5.5 Punktweise und elementweise Erweiterung 252

Literaturverzeichnis **253**

Kapitel 1

Einleitung

As refinement proceeds from high- to low-level descriptions, from expressive to restrictive subsets of a programming language, more and more decisions are made by the developer on the basis of the target architecture. For the distributed systems, that is particularly interesting. Distributed systems differ widely in the way their components communicate, from shared memory to asynchronous message passing to synchronous rendezvous. (...) It is especially important therefore that such details do not intrude too early in the development of a distributed algorithm from its specification (...). Thus in the development of distributed programs are the benefits of refinement especially evident. The restricted sub-language towards which the refinement steps are directed is one that closely matches the target architecture in that particular case. In another case, subsequently on a different architecture, the refinement step need be repeated only from the point at which the difference became apparent.

R.J.R. Back, [BS91]

Die Entwicklung informationsverarbeitender Systeme, die formal nachweisbar bestimmte vorgegebene Eigenschaften erfüllen, ist seit langem ein ambitioniertes Anliegen der Informatik. In der letzten Zeit ist dieses Interesse durch den vermehrten Einsatz verteilter Systeme verstärkt worden. Gerade die Komplexität verteilter Systeme und sich daraus ergebenden Probleme bei deren Entwicklung verleihen dieser Aufgabenstellung zunehmende Bedeutung.

Zu Beginn lag der Schwerpunkt der Untersuchungen zur formalen Entwicklung verteil-

ter Systeme auf der Modellierung der Eigenschaften verteilter Systeme. Dazu wurde eine Vielzahl unterschiedlicher *Beschreibungsmittel* für verteilte Systeme entwickelt, wie zum Beispiel prozeßalgebraische (z.B. CCS [Mil83], (T)CSP [Hoa85]), temporallogische (z.B. [Krö87]), operational orientierte (z.B. Petrinetze ([Rei85], I/O-Automaten [LT89]) oder denotationelle (z.B. stromverarbeitende Funktionen [Bro87a], Failuremengen [Hoa85]). Das Interesse bei der formalen Entwicklung korrekter verteilter Systeme hat sich jedoch mehr und mehr auf die methodischen Aspekte der Aufgabenstellung konzentriert (z.B. FOCUS [BDD+92], ProCoS [ORSS92]). Das Ziel dieser Ansätze liegt hierbei weniger darin, Beschreibungsformalismen für verteilte Systeme zu entwickeln und semantisch zu fundieren, als vielmehr diese in einen methodischen Rahmen einzubetten und dem versierten Benutzer der Formalismen geeignete Leitlinien zur Systementwicklung in die Hand zu geben. Diese sollen es ihm ermöglichen, eine geordnete Vorgehensweise einzusetzen anstatt sich in der Vielfalt von vorhandenen Möglichkeiten zu verlieren.

Als geeignetes Mittel zur Strukturierung des Entwicklungsprozesses wird hierzu im allgemeinen dessen Zergliederung in kleinere Teilschritte eingesetzt. Diese Modularisierung soll den Entwickler dabei unterstützen, sein Augenmerk jeweils nur auf bestimmte Teilprobleme der Gesamtentwicklung zu richten. Die Modularisierung hat jedoch nicht nur die gezielte Bearbeitung kleinerer Teilabschnitte zur Folge. Darüberhinaus können fehlerhafte Entscheidungen an dem Punkt in der Entwicklung entdeckt und behoben werden, an dem sie eingeführt wurden. Oftmals ist es auch möglich, für die bei der Zergliederung entstehenden Teilaufgaben maschinelle Unterstützung anzubieten, oder sie sogar automatisch zu bewältigen.

1.1 Synchrone und asynchron Systeme

Ein verteiltes System wird im allgemeinen als Sammlung von Komponenten (z.B. Prozessen oder Subsystemen) mit einer Möglichkeit zur Kommunikation (z.B. mittels gemeinsamem Speicher oder Nachrichtenaustausch über gemeinsame Kanäle) aufgefaßt (vgl. [Sch93], [BM93]). Bei der formalen Modellierung *nachrichtenorientierter* verteilter Systeme haben sich zwei entgegengesetzte Paradigmen herausgebildet: die Beschreibung verteilter Systeme unter der Annahme *asynchroner*, also unbeschränkt gepufferter Kommunikation, sowie die Beschreibung *synchroner*, also ungepufferter Systeme.[1] Diese beiden Paradigmen unterscheiden sich also hinsichtlich der Modellierung des Sendens von Nachrichten über die für die Kommunikation zur Verfügung stehenden Kanäle.

Bei der Modellierung asynchroner Systeme wird von der Kommunikation mittels nichtblockierendem Senden ausgegangen. Die Vorstellung ist hierbei, daß alle betrachteten Komponenten (System, Subsystem, Systemumgebung) stets empfangsbereit sind; wird eine Nachricht aktuell nicht benötigt, so kann sie in einem unbeschränkten Eingabepuffer

[1] In anderen Kontexten werden die Begriffe *asynchron* und *synchron* - gerade bei der Modellierung digitaler Hardware - oft synonym für *ungetaktet* und *getaktet* verwendet (vgl. [Ber93], [Fuc94]).

zwischengespeichert werden. Bei synchron kommunizierenden Systemen erlaubt die Modellierung, auch solche Systeme zu betrachten, die die Annahme einer Nachricht verweigern können. Die Modellierung unterscheidet sich also im wesentlichen hinsichtlich der Beschreibung der Eingabebereitschaft. Während asynchrone Systeme stets zur Annahme der nächsten anstehenden Eingabe bereit sind, muß diese ständige Eingabebereitschaft für synchrone Systeme nicht gelten.

Auf den ersten Blick mag es unnötig erscheinen, daß zwei unterschiedliche Paradigmen zur Beschreibung verteilter Systeme mit den dazugehörigen Modellen und Beschreibungstechniken entwickelt wurden. Dies läßt sich jedoch aus den unterschiedlichen Ansprüchen an die Systemmodellierung erklären.

Die Beschreibung mittels asynchroner Kommunikation erhebt den Anspruch, eine geeignete und wünschenswerte Abstraktionsebene zur Beschreibung verteilter Systeme zu sein ([Mul89]):

> One of the goals of distributed systems is the exploitation of *parallelism*. And a very natural way to obtain parallelism is to provide *asynchronous* or *non-blocking* communication primitives.

Tatsächlich belegt die Entwicklung aktueller Implementierungsplattformen diesen Anspruch: viele erfolgreiche Plattformen bieten explizit Konzepte zur gepufferten Kommunikation (vgl. MMK [BBB+90, BBLT90], PVM [GBD+94], Hypercube [Wal95]).

Auf der anderen Seite wird diesen Modellen zur Last gelegt, daß sie von der technisch nicht realisierbaren Annahme unbeschränkt gepufferter Kommunikation ausgehen (vgl. z.B. [Spi88]). Da darüberhinaus die Hardware massiv paralleler Systeme prinzipiell auf synchrone Kommunikation ausgerichtet ist, sollte eine geeignete Modellierung diese Tatsache wiederspiegeln.

Doch nicht nur hinsichtlich ihrer Ansprüche an die Modellierung, auch hinsichtlich ihrer semantischen Modelle unterscheiden sich die beiden Paradigmen wesentlich. Gerade bei der Komplexität der denotationellen Semantiken wird dies, wie in Kapitel 2 am Beispiel zweier Semantiken erläutert, deutlich. Während sich die Beschreibungen asynchron kommunizierender Systeme im allgemeinen durch verhältnismäßig einfache Modelle auszeichnen, spiegelt sich die komplexe Sichtweise synchron kommunizierender Systeme auch in deren komplexeren Modellen wider, da hier neben Senden und Empfangen einer Nachricht auch die Verweigerung von Senden und Empfangen beschrieben werden muß.

Es zeigt sich, daß also durchaus Bedarf für beide Arten der Sicht verteilter Systeme besteht, wobei jede ihre spezifischen Vor- und Nachteile aufweist. In Abhängigkeit der Anforderung an das zu entwickelnde System sollte dabei das entsprechende Paradigma, und damit das zugehörige Modell, sorgfältig ausgewählt werden, um die Entwicklung mit vernünftigem Aufwand vornehmen zu können. Das wesentliche Ziel dieser Arbeit ist es, die unterschiedliche Vorteile der verschiedenen Sichten in einer *methodischen Vorgehensweise* zur Entwicklung nachrichtenorientierter Systeme zu verbinden.

1.2 Motivation

Die Existenz zweier unterschiedlicher Sichtweisen zur Beschreibung verteilter, nachrichtenorientierter Systeme, wobei die eine als abstrakter, die andere als implementierungsnäher angesehen wird, wirft die Frage auf, ob sich diese Tatsache nicht für den Entwicklungsprozeß solcher Systeme ausnutzen läßt. Daher wird im folgenden kurz der Nutzen der schrittweisen Entwicklung im allgemeinen, sowie der Nutzen der schrittweisen Verfeinerung im Falle nachrichtenorientierter Systeme im besonderen beleuchtet. Um die in dieser Arbeit eingeführten Neuerungen und damit die Motivation für diese Arbeit zu geben, wird weiterhin ein kurzer Blick auf das Umfeld der Arbeit und verwandte Ansätze geworfen.

1.2.1 Schrittweise Systementwicklung

Als ein grundlegendes Prinzip der methodischen Programmentwicklung wird die Methode der schrittweisen Entwicklung verwendet. Das Ziel ist dabei, aus einer komplexen Problemstellung durch Zerlegung in kleinere, unabhängige Teilaufgaben leichter zu bewältigende Problemstellungen entwickelt. Hierbei ist es wünschenswert, dem Systementwickler eine Unterstützung im Sinne einer Leitlinie an die Hand zu geben, um ihn in einem strukturierten Entwicklungsprozeß zu unterstützen. Diese Leitlinie sollte es ihm ermöglichen, den Prozeß der Systementwicklung möglichst gezielt vorantreiben zu können.

Als besonders vorteilhaft erweist sich eine solche Zergliederung gerade dann, wenn der Entwicklungsprozeß in Phasen der folgenden Art aufgespalten werden kann: in der einen steht die innovative Leistung des Entwicklers im Vordergrund, bei der anderen müssen schematisch zu bewältigende Aufgaben gelöst werden. Da die erste Phase im allgemeinen nicht *top-down* angegangen werden kann, sondern durch wiederholtes *trial and error* bewältigt werden muß, ist es sinnvoll, hier von allen Modellierungsdetails abzusehen, die nicht Teil des eigentlichen Problems sind. Dadurch wird der Aufwand in dieser wiederholt zu durchlaufenden Phase reduziert. Die schematische Vorgehensweise der zweiten Phase erlaubt es dann, die bisher vernachlässigten Aspekte der Aufgabenstellung gezielt zu bewältigen.

Der Sinn dieser Aufspaltung des Entwicklungsprozesses liegt darin, daß der eigentlich innovative der Teil des Entwicklungsprozesses von den eher schematischen Aufgaben deutlich abgetrennt werden soll. Um diese Aufspaltung und die damit einhergehende schrittweise Verfeinerung möglichst gewinnbringend einzusetzen, muß daher möglichst viel Unterstützung für den schematischeren Prozeß der Systementwicklung angeboten werden.

1.2.2 Schrittweise Entwicklung nachrichtenorientierter Systeme

Bei der Modellierung nachrichtenorientierter verteilter Systeme liegt es aus mehreren Gründen nahe, eine asynchrone Sichtweise anzubieten. Dies ist zum ersten darauf zurückzuführen, daß bei der Klasse der nachrichtenorientierten verteilten Systeme der

Schwerpunkt der Entwicklung in erster Linie darin liegt, daß das System auf die ihm von der Umgebung zugestellten Nachrichten mit entsprechenden Nachrichten des gewünschten Inhalts antwortet; das Interesse konzentriert sich also in erster Linie auf den *Datenfluß* (vgl. z.B. [Den85]); die Einzelheiten des Nachrichtenaustauschs, also der Kontrollflußaspekt (vgl. z.B. [Fle94]) wie zum Beispiel Reihenfolge der Aktionen, Verflechtungsgrad, Pufferkapazitäten oder Eingabebereitschaft sind in dieser ersten Näherung nicht von Interesse. Erst auf einer implementierungsnäheren Ebene muß auch dieser Aspekt berücksichtigt werden. Insbesondere sind dann aus Effizienzgründen sogar Fragen nach der Minimalität solcher Verflechtungen oder der Minimalität der Eingabebereitschaft und der damit verbundenen Pufferkapazität von Interesse.

Zweitens erlaubt die asynchrone Sichtweise bereits auf der informellen Ebene wesentlich einfachere Beschreibungen und kommt damit dem Entwickler entgegen. Jener kann nämlich in dieser innovativen Phase auf einfache, sich auf den wesentlichen Aspekt konzentrierende Beschreibungen und die damit verbundenen intuitiven Vorstellungen zurückgreifen.

Zum dritten wird sich in dieser Arbeit auch zeigen, daß gerade auf der Ebene der formalen Beschreibung und der Verifikation verteilter Systeme jene Systeme, die aus asynchron kommunizierenden Agenten aufgebaut sind, einfachere formale Modelle besitzen als ihre entsprechenden synchron kommunizierenden Gegenstücke. Dies spiegelt sich nicht nur in wesentlich eleganteren und kürzeren Charakterisierungen solcher Komponenten, sondern auch in einfacheren Beweisregeln der entsprechenden Kalküle wieder.

1.2.3 Ziele der Arbeit

Wie oben angesprochen, bietet sich aus rein methodischer Sicht eine Trennung der Entwicklung synchron kommunizierender Systeme in zwei Abschnitte an. Im ersten Abschnitt werden die Einschränkungen der Eingabebereitschaft ignoriert und mit einem asynchron orientierten Modell gearbeitet. Erst wenn die Entwicklung des Systems hinsichtlich aller auf dieser Ebene relevanten Aspekte abgeschlossen ist, werden die bisher vernachlässigten Aspekte der synchronen Sichtweise miteinbezogen. In zweiten Abschnitt wird dazu die Annahme der unbeschränkten Eingabebereitschaft aufgehoben.

Ob diese methodisch sinnvolle Vorgehensweise jedoch auch praktikabel ist, ist damit nicht entschieden. Dazu müssen folgende Fragen beantwortet werden:

- Kann die asynchrone Sichtweise von Systemen als Abstraktion synchron kommunizierender Systeme aufgefaßt werden?
- Läßt sich diese Abstraktion methodisch für die Modularisierung des Entwicklungsprozesses ausnutzen?
- Ist eine solche *zweistufige* Vorgehensweise vorteilhaft für den Entwicklungsprozeß synchron kommunizierender Systeme?

Diese Arbeit zeigt, daß eine solche Zweiteilung sinnvoll vorgenommen werden kann und daß dies durchaus von Vorteil für die Entwicklung nachrichtenorientierter verteilter Systeme ist.

Dazu wird eine geeigneten zweistufigen Entwicklungsmethode für nachrichtenorientierte verteilte Systeme vorgestellt. Der Schwerpunkt dieser Arbeit liegt dabei stets auf der Methodik des hier vorgestellten Ansatzes.

1.2.4 Umfeld der Arbeit

Wie bereits in den Abschnitten 1.1 und 1.2.2 angesprochen, wird die methodische Entwicklung verteilter synchroner Systeme durch die Entwicklung und anschließende Verfeinerung asynchroner Systeme in der Praxis vielfach als wünschenswerte Vorgehensweise angesehen. Daher stellt sich naturgemäß die Frage, inwieweit dieses Problem bereits in früheren Arbeiten angegangen worden ist. Um so überraschender ist die Tatsache, daß hierbei der Aspekt der methodischen Vorgehensweise wenig Aufmerksamkeit gefunden hat; bei der gemeinsamen Betrachtung asynchroner und synchroner Systeme standen bisher, ausgenommen [Stø94], eher Fragen semantischer Natur im Vordergrund. Während die Vorgehensweise von [Stø94] in Abschnitt 1.2.5 besprochen wird, sollen im folgenden jene anderen Vorarbeiten kurz näher erläutert werden.

Mächtigkeit der Modelle

Nach der Charakterisierung der asynchron und der synchron kommunizierenden Systeme wurde - vergleiche die folgenden beiden Absätze - untersucht, wie sich spezifische synchrone und asynchrone Paradigmen durch Paradigmen der entsprechenden anderen Sichtweise darstellen lassen. Dabei standen jeweils kaum Fragen der prinzipiellen Umsetzbarkeit im Vordergrund, als vielmehr spezifische Varianten. Im Gegensatz dazu wurde in [Sha92] und [dBP91b, dBP91a] untersucht, welche Sprachkonstrukte auf asynchroner und synchroner Kommunikation basierende Sprachen enthalten sollten, um bezüglich einer für beide definierten operationellen Semantik eine Implementierungsbeziehung zu erlauben. Weiterhin wurde gezeigt, welche Eigenschaften (z.B. Grad der Parallelität) durch einen solchen Übergang verloren gehen.

Realisierung des asynchronen Modells

Abgesehen von der Frage nach der Mächtigkeit der beiden unterschiedlichen Paradigmen wurden auch Untersuchungen angestellt, wie sich Modelle des einen Paradigmas in Modellen des anderen ausdrücken lassen. So untersucht [JJH90] die Frage, wie sich asynchrone Prozesse im allgemeinen innerhalb des Prozeßmodells von TCSP ([Hoa85]) ausdrücken lassen. In ähnlicher Weise wird in [Jos92] untersucht, wie sich asynchrone "rezeptive" Prozesse mit diesem Prozeßmodell darstellen lassen. Beide Ansätze zeigen, daß eine solche Darstellung bis zu einem gewissen Maße möglich ist, jedoch bestehen in beiden Fällen prinzipielle Unzulänglichkeiten hinsichtlich der Behandlung divergenter Systeme.[2]

[2]Der Begriff der "Divergenz" wird in den Kapiteln 3 und A behandelt.

In anderen Arbeiten ([Pan93],[KP94]) wird ein alternatives asynchrones Modell vorgeschlagen, das diese Unzulänglichkeit vermeidet. Die zur Beschreibung verwendete denotationelle Semantik stimmt jedoch nicht mehr mit der ursprünglich verwendeten Semantik zur Beschreibung asynchroner Systeme überein und sollte daher eher als neuer Ansatz zur Beschreibung dieser Systeme aufgefaßt werden. Keine dieser Arbeit behandelt dabei die Frage einer methodischen Kombination der beiden Modelle.

Realisierung des synchronen Modells

Ganz entsprechend zur Behandlung der Frage im vorangegangenen Abschnitt wurden sehr früh mit der Entwicklung synchroner Beschreibungssprachen Untersuchungen angestellt, wie sich diese unter Verwendung asynchroner Kommunikationsmechanismen realisieren lassen. Dabei fand die programmiersprachlich angelegte Darstellung von CSP ("Communicating Sequential Processes", [Hoa78, Hoa83]) besondere Beachtung. Für die dort eingeführten Sprachkonstrukte wurden verschiedene Realisierungen auf Systemen mit asynchroner Kommunikation angegeben und verifiziert (vgl. z.B. [GS85], [Bag86]).

Zweistufiger Ansatz

Auch im ProCoS-Ansatz ([Old91], [ORSS92]) wird ebenso wie in dieser Arbeit ein zweistufiger Ansatz gewählt, der im ersten Schritt eine spurbasierte Beschreibung verwendet, in den späteren Schritten die der Failurebeschreibung entsprechende Readinesssemantik. Dabei verwendet ProCoS die Spurbeschreibung nicht zur Modellierung asynchroner Systeme und zielt daher nicht auf den Einsatz der asynchronen Systemsicht als hilfreiche Entwicklungsmethodik ab. Statt dessen steht hier die transformationelle Systementwicklung im Vordergrund. Zwar wird ein ähnlicher Bezug zwischen Spur- und Readinessbeschreibungen wie in dieser Arbeit hergestellt; trotzdem wird die Spurspezifikation nur als vereinfachte Darstellung einer komplexeren Readinessdarstellung interpretiert. Insbesondere werden unter anderem nichtdeterministische Systeme nicht spur- sondern readinessbasiert beschrieben. Weiterhin wird die Spurspezifikation nur zur Darstellung der Anforderungen an das Gesamtsystem verwendet. Der methodische Einsatz von asynchroner und synchroner Systemsicht spielt also auch in diesem Ansatz keine Rolle.

Zusammenfassung

Insgesamt ist also festzustellen, daß bei der Betrachtung synchron und asynchron kommunizierender Systeme bisher stets Fragen prinzipieller Natur im Vordergrund standen; insbesondere wurde auf Fragen der einheitlichen methodischen Behandlung beider Systemklassen dabei bisher nicht eingegangen. Da aber gerade Punkte wie die Klarheit der Beschreibungstechniken der synchronen und asynchronen Systeme, die Verträglichkeit und die Einfachheit des Modellwechsels für eine solche methodische Vorgehensweise von

grundsätzlicher Bedeutung sind, lassen die hier beschriebenen Ansätze die wesentlichen Fragen für die Behandlung dieses Übergangs offen.

1.2.5 Verwandte Arbeiten

Wie oben besprochen, unterscheiden sich die meisten Arbeiten auf dem Gebiet asynchroner und synchroner Systeme in ihrer Zielsetzung von der hier vorgestellten Arbeit. Eine Ausnahme bilden hierbei lediglich zwei Ansätze.

Dies ist einerseits [Stø94], wo der Übergang von asynchronen zu synchronen Spezifikationen aus technischer Sicht untersucht wird. Dort wird, unter Verwendung von stromverarbeitenden Funktionen als semantisches Modell, dieser Übergang als nichtkompositionale Verfeinerungsrelation aufgefaßt. Auch wird dort weniger auf den methodischen Aspekt der Systementwicklung Wert gelegt, der in der hier vorgestellten Arbeit im Mittelpunkt steht, sondern vielmehr auf die Frage der prinzipiellen Machbarkeit, was letztendlich zu komplexeren Beweisverpflichtungen führt. Insbesondere beschäftigt sich der dort beschriebene Ansatz nicht mit Fragen minimaler Pufferung.

Der zweite, in [ČP93] beschriebene Ansatz beschäftigt sich hingegen mit der Frage der minimalen Pufferung in Datenflußnetzen, allerdings mit der Betonung auf der Entwicklung speichereffizienter Schedulingstrategien. Dazu wird ein nachfrageorientierter Algorithmus ähnlich zu der in [PA85] beschriebenen Strategie zur Bestimmung minimaler Puffergrößen für einen blockierungsfreien Ablauf der Netze angegeben. Diese Netze sind jedoch auf einfache "reguläre" Netze beschränkt. Diese Art von Netzen sind datenunabhängig, d.h. es kann nur die Anzahl der pro Verarbeitungsschritt konsumierten und erzeugten Elemente für jeden Datenflußknoten spezifiziert werden.

1.3 Übersicht

Wie oben besprochen, liegt das Ziel dieser Arbeit darin, eine mehrstufige methodische Vorgehensweise für die Entwicklung nachrichtenorientierter synchron kommunizierender Systeme anzugeben. Der Aufbau dieser Arbeit orientiert sich daher an den in Abbildung 1.1 dargestellten Teilschritten dieser Vorgehensweise. Jedem Teilschritt wird ein eigener Abschnitt gewidmet. Insgesamt ergibt sich damit folgender Aufbau der Arbeit:

- In dem einführenden Kapitel 2 werden die Grundlagen der hier verwendeten Modellierungsansätze für asynchron und synchron kommunizierende Systeme beschrieben. Dazu werden die Spursemantik auf der einen und die Failuresemantik auf der anderen Seite vorgestellt sowie entsprechende Kalküle angegeben und Beispiele für Spezifikationen mit diesen Modellierungsmitteln angeführt. Weiterhin werden diese beiden hinsichtlich ihrer Komplexität bei der Spezifikation und Verifikation von Systemen verglichen. Schließlich werden methodische Aspekte der Spurspezifikation erläutert, die besonders für den in dieser Arbeit vorgestellten Ansatz von Bedeutung sind.

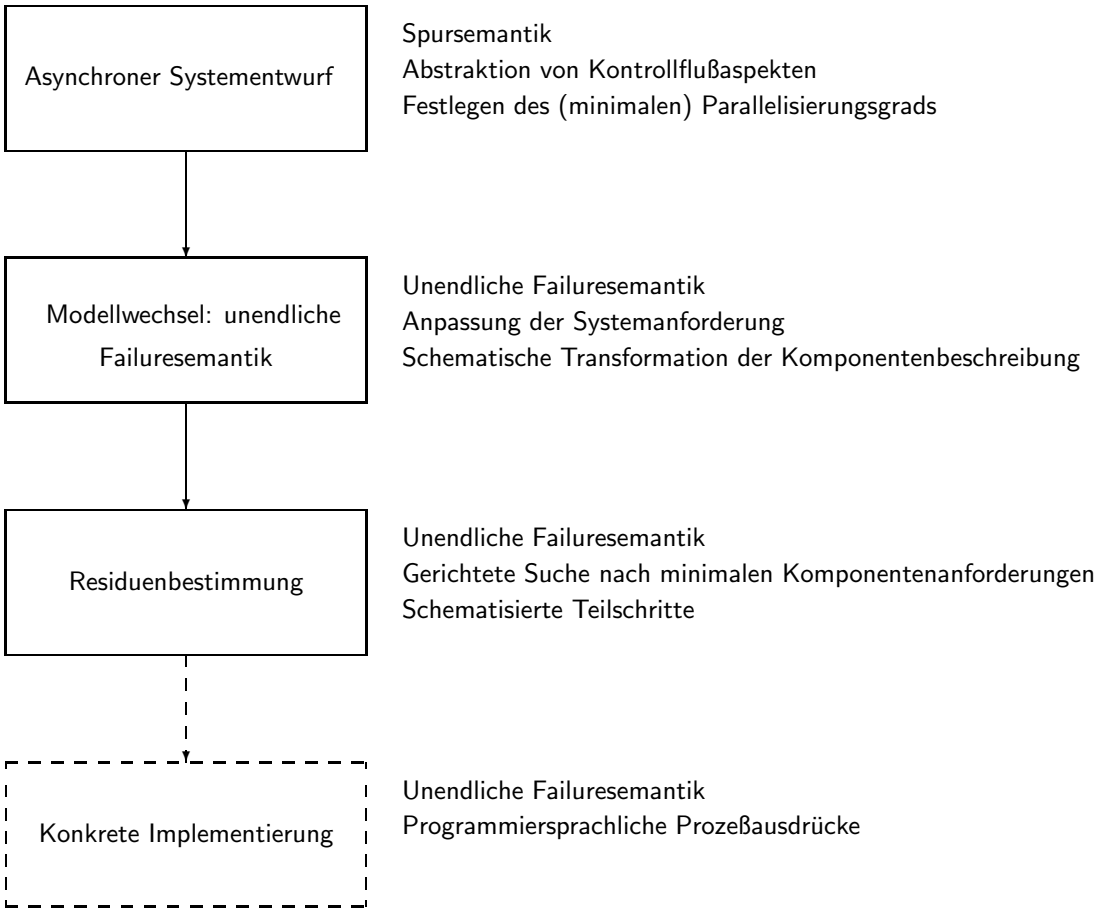

Abbildung 1.1: Übersicht über die Vorgehensweise

- In Kapitel 3 wird der Wechsel von der Spursemantik zur unendlichen Failuresemantik besprochen. Ziel dieses Übergangs ist es, das mittels der Spursemantik beschriebene asynchron kommunizierende System in der komplexeren Failuresemantik entsprechend zu beschreiben; dieser Übergang kann dabei schematisch durchgeführt werden. Als Spezialfall wird dabei die Transformation von Beschreibungen sequentiell ablaufender Prozesse behandelt. Weiterhin wird die wesentliche Eigenschaft dieser Transformation, nämlich ihre Verträglichkeit mit dem strukturellen Systemaufbau, gezeigt. Schließlich werden methodische Aspekte der Verfeinerung von Failurebeschreibungen behandelt, die für die in dieser Arbeit beschriebene Vorgehensweise von Bedeutung sind.
- In Kapitel 4 wird die Bestimmung minimaler Anforderungen an die Komponenten eines Systems mittels *Residuen* beschrieben. Dazu wird zuerst der Begriff des Residuums allgemein eingeführt und dann speziell auf die beim Entwurf verteilter Systeme verwendeten Kompositionsoperatoren zugeschnitten. Auf diese Ergebnisse aufbauend wird dann eine Methode zur Bestimmung minimaler Anforderungen an die Kommunikationseigenschaften synchron kommunizierender Systeme entwickelt. Abschließend wird diese Methode zusammenhängend von der Spurbeschreibung bis zur Bestimmung der minimalen Anforderungen anhand eines Beispiels demonstriert.
- Abschließend wird der hier vorgestellte Ansatz zusammengefaßt und kritisch beleuchtet. Dabei wird das Augenmerk besonders auf die zuvor angesprochene Durchgängigkeit und Einfachheit der Vorgehensweise gerichtet werden.
- Anhang A stellt das unendliche Failuremodell vor, das einerseits die Voraussetzung für das hier beschriebene Verfahren darstellt, andererseits auch davon unabhängig von Interesse ist. Dazu wird detaillierter auf die klassische endliche Failuresemantik eingegangen und deren Unzulänglichkeiten demonstriert. Darauf aufbauend werden die Erweiterung der Failuresemantik zur unendlichen Failuresemantik motiviert und erläutert und deren wesentliche Unterschiede anhand von Beispielen gezeigt.
- In Anhang B finden sich schließlich wesentliche Aussagen über die unendliche Failuresemantik, insbesondere eine Rechtfertigung für das Modell in Form der Verträglichkeit mit der endlichen Failuresemantik.

Kapitel 2

Modelle verteilter Systeme

> *For the top-down design of concurrent communicating (...) systems a specification formalism is an indespensible requisite. Only if one is able to give modular specifications, i.e. self contained abstract specifications also for behaviours of subcomponents of a distributed system, then the decomposition of the system can be done properly and the subcomponents can be developed and verified separately.*
>
> M. Broy, [Bro87b]

Ziel der in dieser Arbeit vorgestellten Vorgehensweise ist die Vereinfachung der Entwicklung verteilter synchron kommunizierender Systeme mittels Aufspaltung in zwei Phasen:

- In der ersten Phase wird durch Abstraktion eine asynchrone Sicht der Problematik ermöglicht.
- Die nachfolgende Phase beschäftigt sich mit spezifischen Problemen, die sich aus den Aspekten der synchronen Kommunikation ergeben.

Ziel dieses Kapitels ist es, die Grundlagen für die Vorgehensweise im allgemeinen und für die Durchführung der ersten Phase bereitzustellen. Dazu werden die folgenden Schwerpunkte behandelt:

Einführung der Modelle: Für die asynchrone und synchrone Systemsicht werden jeweils semantische Modelle benötigt. Abschnitt 2.1 beschäftigt sich dabei mit allgemeinen Fragen der Modellierung verteilter nachrichtenorientierter Systeme, in den Abschnitten 2.2 und 2.3 werden dann Modelle für die asynchrone bzw. synchrone Systemsicht eingeführt, nämlich die *Spursemantik* und die *Failuresemantik*.

Tauglichkeit der Modelle: Die in dieser Arbeit beschriebene Vorgehensweise beruht auf der Annahme, daß die asynchrone Systemsicht ein abstrakteres Modell als die

synchrone zugrundelegt. Dies wird bereits durch die Einführung des synchronen Modells als Erweiterung des asynchronen Modells, besonders aber durch die Einführung des sat-Kalküls und die Vergleichbarkeit der Regeln im asynchronen und synchronen Fall deutlich gemacht. Wichtig ist dabei die Aussage, daß das asynchrone Modell daher eine einfachere Modellierung erlaubt. In Abschnitt 2.5 wird exemplarisch gezeigt, daß diese intuitive Aussage auch formal gerechtfertig ist.

Methodische Aspekte der Abstraktion: Da der Systementwurf auf der asynchronen Ebene nicht das Ziel sondern den Ausgangspunkt der in dieser Arbeit beschriebenen Vorgehensweise darstellt, spielen methodische Fragen bei der Darstellung von System- oder Komponenteneigenschaften eine wichtige Rolle. In Abschnitt 2.2.5 werden daher spezielle Darstellungen für asynchrone sequentielle Prozesse eingeführt. Abschnitt 2.6 behandelt die Frage, welche weiteren Aspekte bei der Entwicklung der asynchronen Systembeschreibung zu berücksichtigen sind.

Die eingeführten Semantiken werden jeweils an Beispielen illustriert. Die Abschnitte 2.1, 2.2 und 2.3 wenden sich an Leser mit keinen oder wenig Vorkenntnissen im Umgang mit diesen Modellen und können von Lesern, denen diese geläufig sind, übersprungen werden.

2.1 Modellierung verteilter Systeme

Da in dieser Arbeit mit unterschiedlichen Sichtweise, nämlich der asynchronen und der synchronen, gearbeitet wird, ist es hier besonders wichtig zu klären, welche Aspekte modelliert werden sollen. Dazu muß einerseits festgelegt werden, welche Verhaltensaspekte des Systems modelliert werden, und andererseits welche strukturellen Merkmale beschrieben werden. Abschnitt 2.1.1 begründet, warum die gewählten Modelle besonders geeignet sind. In Abschnitt 2.1.2 werden die Kompositionsoperatoren zum Aufbau von Systemen eingeführt, die für den in dieser Arbeit vorgestellten Ansatz benötigt werden.

2.1.1 Auswahl der Modelle

Um Aussagen über Systeme mit mathematischer Genauigkeit verifizieren zu können, ist es notwendig, *geeignete* mathematische Modelle zur Verfügung zu haben. Unter "geeignet" ist dabei zu verstehen,

- daß die Modelle hinreichend *abstrakt* sind, um den Nachweis der gewünschten Eigenschaften mit vertretbarem Aufwand durchführen zu können,
- daß die Modelle hinreichend *genau* sind, um alle relevanten Eigenschaften der betrachteten Systeme nachweisen zu können,
- daß die Modelle hinreichend *verträglich* sind, um eine durchgängige Vorgehensweise zu erlauben.

Während die ersten beiden Punkte grundsätzlich von Bedeutung sind, ist der letzte nur im Falle eines Wechsels zwischen diesen Modellen notwendig. Die ersten beiden Punkte werden in der Literatur ausreichend diskutiert und sollen daher hier nicht weiter besprochen werden.[1] Beide gewählten Modelle, die Spur- und die Failuresemantik, sind hinsichtlich ihrer Abstraktheit und Genauigkeit etabliert, insbesondere die Failuresemantik im synchronen Fall. Während jedoch im asynchronen Fall auch andere Modelle geeignet gewesen wären, bietet sich hinsichtlich der Verträglichkeit jedoch gerade diese Kombination und damit die Spursemantik an, da die Failuresemantik als Modell der konkreteren, synchronen Systeme eine Erweiterung des Spurmodells als Modell der abstrakteren, asynchronen Systeme darstellt.[2] Damit spiegelt sich der Wechsel von den asynchron zu den synchron kommunizierenden Systeme auf der methodischen Ebene durch die Erweiterung der relevanten Eigenschaften auch unmittelbar auf der semantischen Ebene wieder.

Vor der Beschreibung der hier verwendeten Modelle in den Abschnitten 2.2 und 2.3 werden zuerst Operatoren eingeführt, die die strukturierte Beschreibung von Systemen erlauben. Weiterhin wird erläutert, welche Eigenschaften wesentlich sind, um Modelle für die Beschreibung verteilter Systeme geeignet zu machen.

2.1.2 Kompositionsoperatoren

Verteilte Systeme werden *modular* aus Subsystemen konstruiert, also ähnlich wie sequentielle Programme aus Unterprogrammen aufgebaut. Während letztere z.B. mittels Prozeduraufrufen kommunizieren, erfolgt bei ersteren der Datenaustausch über Kommunikationsschnittstellen wie z.B. Kanäle. Um verteilte Systeme aufzubauen, werden Kompositionsoperatoren benötigt, mittels derer die Untersysteme zu komplexeren Systemen zusammengesetzt werden können. Viele Formalismen (z.B. [Mil83], [Hoa85], [LT89], [Dil89]) verwenden dazu die zwei folgenden Operatoren:

- den Parralleloperator
- den Abstraktionsoperator

Andere Ansätze (z.B. [Bro87a]) verwenden hierzu andere Operatoren (z.B. Hintereinanderschaltung, interaktionsfreie Nebeneinanderschaltung und Rückkopplung).

Der Parralleloperator dient dazu, zwei Systemkomponenten zu einem System zusammenzuschalten. Er findet sich auch in Sprachen zur Programmierung verteilter Systeme wieder, wie z.B. das `PAR`-Konstrukt in `occam`®[3] ([Inm84]). Bei dieser Zusammenschaltung müssen Aktionen, die beide Komponenten betreffen, gemeinsam ausgeführt werden. Diese Art der Komposition wird daher als *synchronisierende* Parallelkomposition bezeichnet. Aktionen, die die jeweils andere Komponente nicht betreffen, werden von der zweiten Komponente unabhängig, also "parallel" ausgeführt. Als Notation wird in den meisten Ansätzen das

[1] Siehe z.B. [Old85] und [Old86] für eine Diskussion im Falle synchron kommunizierender Systeme.
[2] Siehe [vG96] für einen Vergleich verschiedener semantischer Modelle verteilter Systeme.
[3] `occam`®ist ein eingetragenes Warenzeichen von INMOS Ltd.

Symbol "$\|$" verwendet. Das System bestehend aus den Komponenten P_1 und P_2 wird mit "$P_1 \parallel P_2$" bezeichnet.[4]

Der Abstraktionsoperator dient als Mittel zur hierarchischen Strukturierung der beschriebenen verteilten Systeme. Es gibt kein entsprechendes Konstrukt bei Sprachen zur Programmierung verteilter Systeme.[5] Mit dem Abstraktionsoperator können Aktionen eines Systems nach außen hin verborgen werden; es wird damit von internen Details dieses Systems "abstrahiert". Notationell wird häufig das Symbol "$\backslash$" verwendet. Beispielsweise bezeichnet "$P\backslash H$" das System, das aus P durch Verbergen aller Aktionen der Menge H entsteht.

Da in dieser Arbeit der Aspekt der Entwicklung von Anforderungen an Komponenten im Vordergrund steht, nicht aber die Implementierung dieser Komponenten, werden hier nur solche Operatoren verwendet, die zur Strukturierung eines verteilten Systems notwendig sind. Basiskomponenten und Operatoren, die die Beschreibung implementierungsnaher Komponenten erlauben, wie zum Beispiel in TCSP der Prozeß "STOP", der Präfixoperator $a \rightarrow P$ oder der Auswahloperator "$\Box$", werden hier nicht betrachtet. Wie in [Bro91] festgestellt wird, ist mit den Möglichkeiten, von internen Eigenschaften zu abstrahieren und ein System mittels seiner Komponenten zu beschreiben, die Kernproblematik verteilter Systeme abgedeckt:

> *The key issues in concurrency are abstraction and compositionality.*

2.2 Spursemantik

Erste Versuche zur Beschreibung von verteilten Systemen mittels Spursemantik wurden bereits in [Hoa78] unternommen; dort wurden Spuren jedoch eingesetzt, um Abläufe prozeduraler sequentieller Programme zu charakterisieren. In [Hoa80] wurden sie zur Modellierung synchron kommunizierender Systeme eingesetzt. Die Modellierung beschränkt sich jedoch dabei auf Fragen der partiellen Korrektheit. Zur Beschreibung asynchron kommunizierender System wurden Spuren erst später eingesetzt, zum Beispiel in [Jon87].

In [Hoa94] wird der Begriff der Spur ("trace") im Sinne einer partiellen Beobachtung folgendermaßen eingeführt:[6]

> A communicating process is intended to interact with its environment at certain distinct points in time. Each individual interaction can be recorded as a value from a certain set $\mathcal{A}$ of event names (often called the alphabet of the process). An observation of the behaviour of the process up to a given moment of time

[4] Andere Ansätze, vgl. [Kah74], [Bro87a], verwenden den Operator "$\|$" zur Nebeneinanderschaltung ohne Interaktion zwischen den Komponenten.

[5] Der Abstraktionsoperator ist in Programmiersprachen nicht notwendig, da dort die Parallelkomposition implizit alle gemeinsamen Aktionen verbirgt.

[6] Der Begriff "trace" wird auch in[Maz86] verwendet, dort jedoch verallgemeinert im Sinne einer Halbordnung von Aktionen.

> can be recorded as the sequence of events in which it has engaged so far. This is known as a *trace* (...).

Für die Beschreibung asynchron kommunizierender Systeme wird der Begriff der Spur strenger gefaßt. Das Prinzip besteht hier darin, das Verhalten eines Systems durch die Angabe aller *ausgabevollständigen* Spuren zu beschreiben. Ein System wird also durch die Menge aller Abläufe beschrieben, bei denen - zumindest ohne zusätzliche Eingaben - keine weitere Ausgabeaktion zu erwarten ist.

2.2.1 Grundbegriffe des Spurmodells

In ihrer ursprünglichen Version (z.B. [Hoa78]) werden Spuren als *endliche* Sequenzen von Symbolen eingeführt. Die Menge der Symbole, die zur Bildung dieser Sequenzen verwendet werden, wird oft als *Alphabet* bezeichnet. Im Falle der nachrichtenorientierten Systeme stellen diese Symbole die Nachrichten dar, die vom System empfangen oder versendet werden können. Mit Hilfe von Konstruktoren, Spurfunktionen und -relationen läßt sich zusammen mit einem Alphabet ein abstrakter Datentyp der Spuren definieren (vgl. [Bro89],[Web91]). Im folgenden werden kurz die Konstruktoren und die wichtigsten Funktionen bzw. Relationen charakterisiert.[7]

Konstruktoren

Zur Definition der Spuren werden zwei Konstruktoren verwendet:

$\langle\rangle$**:** Die leere Spur

$a \circ t$**:** Die Spur, deren erstes Element a und deren Rest die Spur t ist

Weiterhin wird oft die Konkatenation von Spuren definiert. Hier wird wegen der Nähe zum Voranstellen von Elementen das gleiche Operationssymbol verwendet:

$s \circ t$**:** Die Spur bestehend aus der Konkatenation der Spuren von s und t

Durch *endliche Anwendung* der beiden Konstruktoren wird unter Verwendung von Elementen aus dem Alphabet A die Menge der Spuren über A, bezeichnet mit A^*, definiert.

Funktionen und Relationen auf Spuren

Die folgenden drei Funktionen werden generell bei Spurspezifikationen eingesetzt und finden auch in dieser Arbeit wiederholt Anwendung ($a \in A, s, t \in A^*$):

[7] Ansätze zur bereichstheoretischen Definition unter Verwendung des Theorembeweisers *Isabelle* ([Pau94a]) finden sich in [Reg94].

B©t: Die Beschränkung einer Spur auf Elemente aus B:

$$\begin{array}{lcll} B©\langle\rangle & = & \langle\rangle & \\ B©(a \circ t) & = & B©t & \text{falls } a \notin B \\ B©(a \circ t) & = & a \circ (B©t) & \text{falls } a \in B \end{array}$$

$\#t$: Die Länge einer Spur:

$$\begin{array}{lcl} \#\langle\rangle & = & 0 \\ \#(a \circ t) & = & 1 + \#t \end{array}$$

$s \sqsubseteq t$: Die Präfixordnung auf Spuren:

$$s \sqsubseteq t \stackrel{\text{def}}{=} \exists r.s \circ r = t$$

Als Abkürzung für $\{a\}©t$ wird stellenweise auf die Mengenschreibweise $\{a\}$ verzichtet und statt dessen kurz $a©t$ verwendet. Zusätzlich wird die Notation $s \sqsubset t$ als Abkürzung für $s \sqsubseteq t \wedge s \neq t$ verwendet.

Punktweise definierte Spurfunktionen

Über die oben definierten Funktionen hinaus werden oft Abbildung zwischen Spurmengen mit unterschiedlichen Alphabeten benötigt. Die Definition solcher Abbildungen werden dabei im allgemeinen "punktweise", d.h. für die einzelnen Elemente des Ausgangsalphabets definiert; dieses Definition wird dann in natürlicher Weise auf die Menge der Spuren über diesem Alphabet erweitert.

Definition 2.2.1 (Punktweise Erweiterung) Sei $f : A \to B$ eine totale Abbildung zwischen zwei Alphabeten A und B. Dann heißt die dadurch definierte Abbildung $f^* : A^* \to B^*$ mit

$$\begin{array}{rcl} f^*(\langle\rangle) & = & \langle\rangle \\ f^*(a \circ t) & = & f(a) \circ f^*(t) \text{ mit } a \in A, t \in A^* \end{array}$$

die *punktweise Erweiterung der Alphabetsfunktion* f. ∘

Erweiterung der Spuren

In der ursprünglichen Version ([Hoa78]) wurde das Spurmodell unter Verwendung endlicher Sequenzen über dem gegebenen Grundalphabet definiert. Dies erwies sich für die Behandlung partieller Korrektheit synchron kommunizierender Systeme als ausreichend. Die vollständige Behandlung asynchron kommunizierender Systeme ist damit jedoch nicht möglich (vgl. auch [Dil89]). Beispielweise kann damit kein System beschrieben werden, das unbeschränkt Ausgaben produziert. Dieses Problem wird durch die Erweiterung der Menge der Spuren A^* um unendliche Spuren A^∞ auf potentiell unendliche Abläufe $A^\omega = A^* \cup A^\infty$, auch *Ströme* genannt, gelöst.

Die Konstruktoren und Funktionen auf den Spuren werden in entsprechender Weise angepaßt. Dabei gilt für die Konkatenation:

$$s \circ t = s \text{ falls } s \in A^{\infty} \tag{2.1}$$

Die Menge der Ströme über einem gegebenen Alphabet stellt hinsichtlich der Präfixordnung $\sqsubseteq$ eine vollständige Halbordnung mit $\langle\rangle$ als kleinstem Element dar. Die Anwendung der Funktionen auf unendliche Objekte wird dabei als kleinste obere Schranke ("lub") der Anwendung der Funktionen auf alle endlichen Präfizes verstanden.[8]

2.2.2 Spurbeschreibung asynchroner Systeme

Formal wird ein asynchron kommunizierendes System durch eine *Prozeß* beschrieben, der aus

- die Menge der Eingabeaktionen der Umgebung an das System,
- die Menge der Ausgabeaktionen des Systems an die Umgebung und
- die Menge der Abläufe (Spuren der Ein- und Ausgabeaktionen) des Systems

besteht. Die Aktionsmengen werden im allgemeinen und auch in dieser Arbeit auf endliche Mengen beschränkt. Dies führt zu folgender Definition:

Definition 2.2.2 (Asynchroner Prozeß) Ein Tripel (I, O, T) mit $I \cap O = \emptyset$ und $T \subseteq (I \cup O)^{\omega}$ wird als *asynchroner Prozeß* bezeichnet. Dabei bezeichnet man

- I als die *Menge der Eingabeaktionen*
- O als die *Menge der Ausgabeaktionen*
- T als die *Menge der Abläufe (Spuren)*

∘

Dabei enthält T nur *(ausgabe-)vollständige* Abläufe, d.h. Abläufe, bei denen das beschriebene System in einem Zustand ist, in dem - ohne weitere Eingaben - keine weiteren Ausgaben mehr erzeugt werden.

Für T werden dazu im allgemeinen noch weitere Abschlußeigenschaften gefordert. Da in dieser Arbeit jedoch die methodischen Aspekte im Vordergrund stehen, wird auf die Behandlung dieser Eigenschaften verzichtet. Eine ausführliche Behandlung dieser Eigenschaften findet sich zum Beispiel in [BDDW91] oder auch [Jon87].

[8] Für die Definition der Begriffe "vollständige Halbordnung", "kleinstes Element" und "kleinste obere Schranke" siehe z.B. [Win93].

2.2.3 Kompositionsoperatoren

Bisher wurden die beide Operatoren "$\|$" und "$\backslash$" lediglich als *syntaktische* Operatoren zum Aufbau verteilter, hierarchischer Systeme eingeführt. In diesem Abschnittes werden nun zu diesen syntaktischen Operatoren die entsprechenden *semantischen* Operatoren definiert. Diese operieren auf asynchronen Prozessen, die in Definition 2.2.2 eingeführt worden sind.

Parallelkomposition

Wie bereits erläutert, wird durch $P_1 \parallel P_2$ die synchronisierende Parallelkomposition zweier Prozesse beschrieben. Diese Komposition ist eine partielle Operation und wird aus Gründen der Kompositionalität nur für den Fall definiert, daß die Ausgabemengen der beiden Prozesse disjunkt sind. Der entstehende Gesamtprozeß besitzt als Eingabemenge die Eingaben der Prozesse P_1 und P_2 abzüglich der Ausgabemengen der Prozesse. Die Ausgaben von $P_1 \parallel P_2$ sind die Ausgaben der beiden Prozesse P_1 und P_2.

Da die Parallelkomposition synchronisierend auf die beiden Prozesse P_1 und P_2 wirkt, müssen Aktionen, die von beiden gemeinsam ausgeführt werden können, auch von beiden Prozessen synchron ausgeführt werden. Wird ein beliebiger Ablauf aus der Menge der Abläufe des Gesamtsystems ausgewählt, und wird dieser auf Aktionen des Prozesses P_1 bzw. P_2 eingeschränkt, so muß sich dadurch jeweils ein Ablauf von P_1 bzw. P_2 ergeben.

Definition 2.2.3 (Parallelkomposition) Seien P_1 und P_2 zwei Prozesse mit $P_1 = (I_1, O_1, T_1)$ und $P_2 = (I_2, O_2, T_2)$, wobei $O_1 \cap O_2 = \emptyset$ gelte. Die *Parallelkomposition* $P_1 \parallel P_2$ wird definiert als der Prozeß mit

$$P_1 \parallel P_2 = ((I_1 \cup I_2)\backslash(O_1 \cup O_2), O_1 \cup O_2, T_{12})$$

wobei

$$T_{12} \stackrel{\text{def}}{=} \{t \mid (I_1 \cup O_1)©t \in T_1 \wedge (I_2 \cup O_2)©t \in T_2\}$$

gilt. ∘

Abstraktion

Bei der Abstraktion wird die Menge der Aktionen, die verborgen werden können, auf die Ausgabeaktionen des zu abstrahierenden Prozesses eingeschränkt. Dies ist ausreichend, da ja bereits durch die Parallelkomposition die Menge der Eingabeaktionen eingeschränkt wird, indem Eingabeaktionen in Ausgabeaktionen umgesetzt werden. Der entstehende Prozeß besitzt daher die gleiche Eingabemenge wie der ursprüngliche, die Menge der Ausgabeaktionen wird um diejenigen vermindert, von denen abstrahiert werden soll.

Mittels der Abstraktion wird jeder Ablauf aus der Menge der ursprünglichen Abläufe auf alle Aktionen abzüglich der abstrahierten eingeschränkt. Zu jedem abstrahierten Ablauf von $P\backslash O'$ muß also ein Ablauf von P existieren, dessen Einschränkung auf Aktionen von $P\backslash O'$ den abstrahierten Ablauf ergibt.

Definition 2.2.4 (Abstraktion) Sei P ein Prozeß mit $P = (I, O, T)$ und O' eine Menge von Ausgabeaktionen mit $O' \subseteq O$. Die Abstraktion $P\backslash O'$ von P hinsichtlich O' wird dabei definiert als

$$P\backslash O' = (I, O\backslash O', T')$$

wobei

$$T' \stackrel{\text{def}}{=} \{((I \cup O)\backslash O')©t \mid t \in T\}$$

∘

2.2.4 Beispiel

In diesem Abschnitt werden die beiden Kompositionsoperatoren "$\|$" und "$\backslash$" am Beispiel der Komposition zweier unbeschränkter Puffer demonstriert.

Beispiel 2.2.1 (Unbeschränkte Puffer) Zuerst wird die Spurdefinition eines unbeschränkten Puffers eingeführt. Zu einer gegebenen Datenmenge D werden definiert:

$$\begin{aligned} I &\stackrel{\text{def}}{=} \{i.d \mid d \in D\} \\ O &\stackrel{\text{def}}{=} \{o.d \mid d \in D\} \end{aligned}$$

Damit bezeichnet I bzw. O die Menge der Kommunikationsaktionen auf dem Kanal I bzw. O, wobei Daten aus D übertragen werden. Um die bei einer Kommunikationsaktion übertragenen Daten zu kennzeichnen, wird die Abbildung data verwendet, die auf Spuren punktweise wie folgt definiert ist:

$$\begin{aligned} \mathsf{data} : (I \cup O) &\to D \\ \mathsf{data}(i.d) &\stackrel{\text{def}}{=} d \\ \mathsf{data}(o.d) &\stackrel{\text{def}}{=} d \end{aligned}$$

data^* bezeichne die in Definition 2.2.1 eingeführte punktweise Erweiterung von data.

Weiterhin wird ein Puffer P_1 definiert als ein asynchroner Prozeß mit der Eigenschaft

$$P_1 \stackrel{\text{def}}{=} (I_1, O_1, T_1)$$

mit entsprechend definierten I_1, O_1 und data^*, wobei für seine Spurmenge gelten soll

$$T_1 \subseteq \{t \mid \ (\forall s \sqsubseteq t.\ \mathsf{data}^*(O_1©s) \sqsubseteq \mathsf{data}^*(I_1©s)) \wedge (\mathsf{data}^*(I_1©t) = \mathsf{data}^*(O_1©t))\}$$

Anschaulich bedeutet dies, daß alle ausgegebenen Daten vorher als Eingabe empfangen wurden, und mindestens so viele Daten ausgegeben werden wie zuvor als Eingabe empfangen wurden. Der Puffer P_2 wird analog beschrieben:

$$P_2 \stackrel{\text{def}}{=} (I_2, O_2, T_2)$$

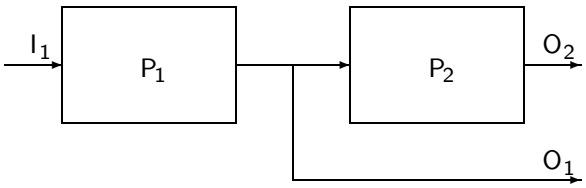

Abbildung 2.1: Die Parallelkomposition zweier Puffer

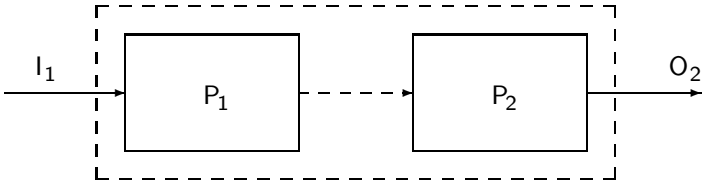

Abbildung 2.2: Die Parallelkomposition mit verborgenem internen Kanal

wobei I_2,O_2 und T_2 entsprechend definiert sind mit $I_2 = O_1$.
Das System $P_1 \parallel P_2$ ist definiert als

$$P_1 \parallel P_2 \stackrel{\text{def}}{=} (I_1, O_1 \cup O_2, T)$$

mit

$$T \subseteq \{t \mid \ (\forall s \sqsubseteq t.\ \mathsf{data}^*(O_1 \copyright s) \sqsubseteq \mathsf{data}^*(I_1 \copyright s) \wedge \mathsf{data}^*(O_2 \copyright s) \sqsubseteq \mathsf{data}^*(O_1 \copyright s)) \wedge \mathsf{data}^*(I_1 \copyright t) = \mathsf{data}^*(O_1 \copyright t) \wedge \mathsf{data}^*(O_1 \copyright t) = \mathsf{data}^*(O_2 \copyright t)$$

Das System $P_1 \parallel P_2$ wird dargestellt in Abbildung 2.1. Die Tatsache, daß die Nachrichten von P_1 nach P_2 ebenso wie die Ausgaben von P_2 nach außen hin sichtbar sind, wird dabei durch den aufgespaltenen Kanal O_1 dargestellt.

Im Anschluß an die Parallelkomposition wird nun der interne Kanal, über den P_1 und P_2 kommunizieren, nach außen hin verborgen. Dies geschieht durch das Verbergen aller Interaktionen, die auf diesem Kanal stattfinden, also durch die Abstraktion von allen Aktionen aus O_1. Das Gesamtsystem $((P_1 \parallel P_2)\backslash O_1)$, wie in Abbildung 2.2 dargestellt, wird also folgendermaßen beschrieben:

$$(P_1 \parallel P_2)\backslash O_1 \stackrel{\text{def}}{=} (I_1, O_2, T')$$

wobei

$$T' \subseteq \{t \mid \ (\forall s \sqsubseteq t.\ \mathsf{data}^*(O_2 \copyright s) \sqsubseteq \mathsf{data}^*(I_1 \copyright s)) \wedge \mathsf{data}^*(I_1 \copyright t) = \mathsf{data}^*(O_2 \copyright t)\}$$

Die Komposition zweier Puffer auf die obige Art ergibt also wiederum einen Puffer. $\diamond$

2.2.5 Spezielle Formen der Spurdarstellung

In den Abschnitten 2.2.2 und 2.2.3 wurde das semantische Modell zur Beschreibung asynchron kommunizierender Prozesse eingeführt. Dabei lag der Schwerpunkt auf der mathematischen Modellierung und weniger auf der systematischen Beschreibung von Prozessen.

In diesem Abschnitt wird daher kurz gezeigt, wie spezielle Formen der Spurdarstellung zur Spezifikation asynchroner Prozesse eingesetzt werden können.

Dazu wird zunächst eine sehr allgemeine Form der Spurdarstellung eingeführt, die auf der Zerlegung der Spurbeschreibungen in zwei Anteile, die Beschreibung der *partiellen* und die der *vollständigen Abläufe*, beruht. Diese allgemeine Darstellungsform wird dann weiter eingeschränkt, um bestimmte Klassen asynchroner Prozesse schematisch zu beschreiben. Dazu werden Prozesse mit den folgenden Eigenschaften herangezogen:

Eingabebereitschaft: Die *Eingaben* eines Prozesses werden nur von seiner Umgebung kontrolliert; eine Eingabe ist stets möglich.

Sequentieller Ablauf: Die *Ausgaben* des Prozesses werden sequentiell erzeugt; unabhängig davon können Ein- und Ausgaben parallel stattfinden.

Diese Einschränkungen werden einerseits gewählt, da sich die entsprechenden Eigenschaften vielfach leicht nachweisen lassen und Prozesse mit diesen Einschränkungen sehr einfach schematisch dargestellt werden können. Andererseits sind diese Einschränkungen allgemein genug, um das Ergebnis der Entwicklung eines Systems auf Spurebene darzustellen; tatsächlich liegen viele Spurbeschreibungen von System- oder Komponentenverhalten natürlicherweise in dieser Form vor (vgl. [Web91]). Da - wie in 3.4 beschrieben wird - für diese Prozesse schließlich der Wechsel von der Spur- zur Failureebene durch eine einfache Transformation möglich ist, stellen solche Beschreibungen eine geeignete Darstellungsform von Prozessen für die in dieser Arbeit beschriebene Vorgehensweise dar.

Partielle und vollständige Abläufe

Wie das Beispiel 2.2.4 sowie beispielsweise [DW92], [Web91], [BDDW91] und [Rud95] zeigen, werden Spurbeschreibungen im allgemeinen ganz natürlich in zwei unterschiedliche Anteile zerlegt:

Partielle Eigenschaften: Eigenschaften, die stets während eines Ablaufs gelten müssen, unabhängig davon, ob die Ausgabe vollständig produziert wurde oder nicht.

Vollständige Eigenschaften: Eigenschaften, die beschreiben, ob die Ausgabe noch unvollständig oder bereits vollständig ist.

Da partielle Eigenschaften stets während eines Ablaufs gelten sollen, müssen mit einem Ablauf auch für dessen Teilabläufe gelten. Formal bedeutet dies:

$$P(t_1 \circ t_2) \Rightarrow P(t_1) \tag{2.2}$$

Die vollständige Spurbeschreibung ergibt sich durch die Kombination dieser beiden Eigenschaften. Diese Form der Spurbeschreibung wird in der folgenden Definition charakterisiert.

Definition 2.2.5 (Partielle und vollständige Abläufe) Seien I das Eingabealphabet und O das Ausgabealphabet eines Prozesses sowie $P, C : (I \cup O)^\omega \to \mathbb{B}$ zwei Spurprädikate, wobei P die Anforderung 2.2 erfüllt. Dann heißt $T_{P,C}$ mit

$$T_{P,C}(t) \stackrel{\text{def}}{=} P(t) \wedge C(t)$$

die Spurmenge der *Beschreibung mittels P und C*, P die *Beschreibung der partiellen Eigenschaften* und C *die Beschreibung der vollständigen Eigenschaften.* ∘

Diese Form der Zerlegung von Spurbeschreibungen ist besonders für die in dieser Arbeit beschriebene Vorgehensweise wichtig. Beim Wechsel von der asynchronen zur synchronen Systemsicht, wie in Kapitel 3 beschrieben, sind die partiellen Eigenschaften diejenigen Eigenschaften, die unverändert erhalten bleiben. Damit werden bereits mit den partiellen Eigenschaften auf der Ebene der Spurbeschreibungen die partiellen Eigenschaften der Abläufe der Failurebeschreibung festgelegt. Die Aufgabe der in den weiteren Kapiteln beschriebenen Vorgehensweise wird es daher sein, dafür zu sorgen, daß auch die vollständigen Eigenschaften mit entsprechenden Anpassungen von der asynchronen auf die synchrone Systemsicht übertragen werden können.

Die Spezifikation mit partiellen und vollständigen Eigenschaften ist sehr ähnlich zur Spezifikation mit Sicherheits- und Lebendigkeitseigenschaften, wie sie in beispielweise in [Lam89] und [DW89] beschrieben wird. In beiden Fällen wird eine Spezifikation in zwei Anteile zerlegt; davon ist jeweils ein Anteil, der Sicherheitsanteil bzw. der partielle Anteil, präfixabgeschlossen. Bei den Sicherheitseigenschaften wird jedoch zusätzlich noch die Zulässigkeit gefordert. Das folgende Beispiel zeigt, daß der Abschluß durch die Zulässigkeitsforderung Sicherheitseigenschaften für die Beschreibung partieller Eigenschaften ungeeignet macht. Diese partiellen Eigenschaften spielen jedoch gerade beim Wechsel von der Spur- zur Failuredarstellung eine wichtige Rolle.

Beispiel 2.2.2 (Unterbrechbarer Prozeß) Seien I und O Ein- und Ausgabealphabet eines asynchronen Prozesses $(I, O, \{t \mid T(t)\})$ mit

$$T(t) \stackrel{\text{def}}{=} I \copyright t = \langle\rangle \Leftrightarrow O \copyright t \in O^\infty$$

Dieser Prozeß kann Ausgaben produzieren, solange keine Eingabe vorliegt. Erhält er eine Eingabe, so bricht er endlich viele Ausgaben später ab.

Die Zerlegung in Sicherheits- und Lebendigkeitsanteile $S(t)$ bzw. $L(t)$ liefert hier

$$\begin{aligned} S(t) &\stackrel{\text{def}}{=} \mathbf{T} \\ L(t) &\stackrel{\text{def}}{=} T(t) \end{aligned}$$

Damit ist aber der Ablauf $i \circ o$ mit $i \in I$ und $o \in O^\infty$ ein sicherer Ablauf. Er stellt jedoch keinen geeigneten partiellen Ablauf dar, da $i \circ o$ mehr Ausgaben enthält als in einem Ablauf möglich sind. Eine geeignete Aufspaltung in partielle und vollständige Abläufe ist

$$\begin{aligned} P(t) &\stackrel{\text{def}}{=} O \copyright t \in O^\infty \Rightarrow I \copyright t = \langle\rangle \\ C(t) &\stackrel{\text{def}}{=} I \copyright t = \langle\rangle \Rightarrow O \copyright t \in O^\infty \end{aligned}$$

◇

Die Einschränkung der Sicherheitseigenschaften auf zulässige Eigenschaften ist auf den methodischen Vorteil beim Nachweis der Gültigkeit von zustandsbasierten Prozeßbeschreibungen mittels Induktionsbeweisen oder bei der konstruktiven Entwicklung von zustandsbasierten Prozeßbeschreibungen aus Verhaltensbeschreibungen zurückzuführen. Bei der hier vorgestellten Vorgehensweise stehen jedoch andere Aspekte im Vordergrund: die partiellen Eigenschaften sollen sowohl in der asynchronen als auch der synchronen Systemsicht gelten; daher ist es nötig, bereits auf der Ebene der Spurbeschreibung die partiellen Eigenschaften zu identifizieren, die auch für alle Abläufe auf der Ebene der Failuredarstellung gelten müssen. Hier werden daher keine grundlegenden Fragen wie die Zerlegbarkeit von Anforderungen in partielle und vollständige Eigenschaften untersucht. Statt dessen werden im folgenden spezielle, einfach zu identifizierende Darstellungsformen für partielle und vollständige Eigenschaften beschrieben, die besonders für die in dieser Arbeit vorgestellte Vorgehensweise geeignet sind.

Relationale Beschreibung von Prozessen

Wie oben erwähnt, lassen sich für eine Teilklasse der asynchronen Prozesse einfache Transformationsregeln angeben. Die hier betrachteten *Spurbeschreibungen* müssen dazu in einer einfachen syntaktischen Form vorliegen. Hierbei wird das Verhalten eines Prozesses - also die Menge der ihn charakterisierenden Spuren - im wesentlichen durch die Charakterisierung der *Ein- und Ausgabeanteile der Spuren* beschrieben. Dabei ergibt sich die Beschreibung der Spuren des Prozesses schematisch aus der Beziehung der Ein- und Ausgabeanteile. Da also der wesentliche Anteil der Spurbeschreibung des Prozesses durch eine Ein-/Ausgaberelation gegeben ist und die eigentliche Charakterisierung des Prozesses durch Spurmengen - wie in Definition 2.2.6 festgelegt wird - schematisch anhand der Relation erfolgt, wird im folgenden von einer *relationalen Beschreibung* des Prozesses gesprochen, auch wenn als Prozeßmodell weiterhin die Menge der Spuren verwendet wird.[9]

Diese *relationalen* Prozeßbeschreibungen finden gerade im Bereich der asynchronen Systeme vielfach Anwendung (vgl. z.B. [DW92], [Web91], [BDDW91]). Auch die in den folgenden Kapiteln verwendeten Beispiele lassen sich im allgemeinen in dieser syntaktischen Form darstellen.

Wie oben besprochen, spielen dabei die Begriffe “partielle Ausgabe” und “vollständige Ausgabe” eine wichtige Rolle. In dem in dieser Arbeit vorgestellten Ansatz werden jedoch keine Relationen auf Spuren zur Beschreibung asynchron kommunizierender Systeme verwendet, sondern Mengen von Spuren. Die folgende Definition beschreibt die Beziehung zwischen diesen Begriffen und der Spurdarstellung eines asynchronen Prozesses. Dazu wird für solche Ein/Ausgaberelationen ein Spurdarstellungsschema verwendet. Dazu wird gefordert, daß

[9] Eine ähnliche Darstellungsweise, bei der die syntaktische Darstellung der Spezifikation, nicht aber das semantische Modell relational orientiert ist, findet sich in [BS94a] und [BS94b] für stromverarbeitende Funktionen; dort werden allerdings auch die Spezifikation direkt relational notiert und Relationen nicht als Strukturierungsmittel angesehen.

- einerseits zu jedem Zeitpunkt des Ablaufs jede bisher produzierte Ausgabe zu einer Ausgabe ergänzt werden kann, die aufgrund der aktuell vorliegenden Eingabe möglich ist,
- und andererseits am Ende des Ablaufs auch die ganze Ausgabe produziert wurde.

Diese informelle Anforderung wird in der folgenden Definition präzisiert. Die Charakterisierung "partieller" und "vollständiger Ausgaben" wird dabei erst im Anschluß behandelt.

Eine ähnliche Zuordnung von Spuren, allerdings für Funktionen auf Spuren bzw. Strömen anstatt Relationen, wird in [Web91] definiert. Dort wird allerdings keine eigene Charakterisierung eingeführt, sondern der Zusammenhang zwischen Spuren und den in [BDD+93] eingeführten stetigen Funktionen hergestellt. Insbesondere sind damit die im folgenden beschriebenen unterbrechbaren Prozesse nicht beschreibbar.

Definition 2.2.6 (Relationale Prozeßbeschreibung) Seien I und O das Eingabe- bzw. Ausgabealphabet eines asynchrones Prozesses und $P, C : I^\omega \times O^\omega \to \mathbb{B}$ Relationen auf Ein- und Ausgabespuren. gilt. Dann heißt $T_{P,C}$ mit

$$T_{P,C}(t) \stackrel{\text{def}}{=} (\forall s \sqsubseteq t.P(I©s, O©s)) \wedge C(I©t, O©t)$$

die Spurmenge der *relationale Beschreibung* (P, C), P die *relationale Beschreibung der partiellen Ausgaben* und C *die relationale Beschreibung der vollständigen Ausgaben.* ∘

Offensichtlich liefert die relationale Beschreibung eine Beschreibung mittels partieller und vollständiger Eigenschaften, wie in Definition 2.2.5 beschrieben, da

$$(\forall s \sqsubseteq t_1 \circ t_2.P(I©s, O©s)) \Rightarrow (\forall s \sqsubseteq t_1.P(I©s, O©s))$$

gilt. Damit beschreibt also $(\forall s \sqsubseteq t.P(I©s, O©s))$ die partiellen Eigenschaften sowie $C(I©t, O©t)$ die vollständigen.

Um sicherzustellen, daß die Spurmenge einer relationalen Beschreibung (P, C) auch tatsächlich einen asynchronen Prozeß beschreibt, müssen die Beschreibungen der partiellen und vollständigen Ausgaben darüberhinaus noch weitere Anforderungen erfüllen. Diese werden in den folgenden Abschnitten eingeführt.

Asynchrone Prozesse

Da auf der Spurebene nur asynchron kommunizierende Systeme betrachtet werden, müssen die relationalen Beschreibungen einer einfachen Anforderung genügen, nämlich der *ständigen Eingabebereitschaft.* Eine erste - offensichtlich operationell zu schwache - Anforderung ist in dieser Hinsicht die Forderung nach der Totalität der relationalen Beschreibung der partiellen Ausgaben:

$$\forall i.\exists o.P(i, o) \tag{2.3}$$

Diese Anforderung läßt sich unterschiedlich verschärfen. In dieser Arbeit wird dies durch die Verschärfung der Anforderungen zu *monotonen* und *unterbrechbaren* Prozessen in den folgenden Abschnitten demonstriert. Darüberhinaus gelten für asynchrone Prozesse noch weitere Eigenschaften, wie beispielsweise die Verzögerbarkeit der Ausgabe gegenüber der Eingabe,

$$P(i, o_1 \circ o_2) \Rightarrow P(i, o_1)$$

Für die in dieser Arbeit beschriebene Vorgehensweise wird jedoch nur Eigenschaft 2.3 bzw. die entsprechenden Verschärfungen eine Rolle spielen.

Sequentielle Prozesse

Wie zu Beginn von 2.2.5 angesprochen, wird unter einem sequentiellen Prozeß ein Prozeß verstanden, dessen *Ausgabeaktionen* sequentiell erzeugt werden. Ein- und Ausgabe hingegen sind wegen des asynchronen Charakters auch weiterhin entkoppelt und können daher auch parallel stattfinden. Da in diesem Abschnitt nur sequentielle Prozesse modelliert werden sollen, gilt für diese Prozesse die folgende Anforderung

$$P(i, o_1) \Rightarrow \exists o_2 . P(i, o_1 \circ o_2) \wedge C(i, o_1 \circ o_2) \tag{2.4}$$

Es wird also verlangt, daß sich jede partielle Ausgabe zu einer vollständigen Ausgabe ergänzen läßt.[10] Zu beachten ist weiterhin, daß so implizit gefordert wird, daß eine unendliche partielle Ausgabe auch stets bereits eine vollständige Ausgabe darstellt, da - laut Gleichung 2.1 in Abschnitt 2.2.1 -

$$\forall o_1 \in O^\infty, o_2 \in O^\omega . o_1 \circ o_2 = o_1$$

Diese Anforderung wird von Prozessen, die ihr Ausgabe sequentiell erzeugen, durchaus erfüllt. Dieser Aspekt wird im folgenden Abschnitt im Zusammenhang mit der Realisierbarkeit von Bedeutung sein. Die Anforderung 2.4 gilt jedoch nicht für parallele Systeme, wie zum Beispiel den in 2.6.1 beschriebenen Doppelkanalpuffer. Solche Systeme lassen sich jedoch als Parallelkomposition mehrerer sequentieller Prozesse darstellen, wie in 2.6 beschrieben.[11]

[10] Diese Abhängigkeit zwischen P und C weist eine ähnliche Beziehung auf wie die *Maschinenabgeschlossenheit* bei Sicherheits- und Lebendigkeitseigenschaften: jede *endliche* sichere Spur kann zu einer sicheren und lebendigen Spur ergänzt werden (vgl. [AL91], [Web91]). Im Gegensatz dazu kann jede, auch nichtendliche partielle Ausgabe zu einer partiellen und vollständigen Ausgabe ergänzt werden.

[11] Alternativ ist eine Darstellung mit mehrstelligen Relationen denkbar, bei der die Stelligkeit durch die Anzahl der Kanäle bestimmt wird.

Monotone Prozesse

Während die geforderten Eigenschaften der in diesem Abschnitt betrachteten Prozesse, nämlich die Sequentialität und die Realisierbarkeit, Einschränkungen hinsichtlich des Ausgabeverhaltens darstellen, wurde für die Eingabe nur die schwache Forderung der Totalität (2.3) erhoben. Wie bei der Einführung dieser Anforderung bereits erwähnt, soll diese durch stärkere Anforderungen ersetzt werden. Dabei lassen sich verschiedene Klassen von Prozessen unterscheiden, abhängig vom Verhalten beim Empfang zusätzlicher Eingaben von der Umgebung. In diesem Abschnitt soll dazu die Klasse *monotoner Prozesse* eingeführt werden. Dabei handelt es sich um Prozesse, bei denen der Empfang zusätzlicher Eingaben von der Umgebung stets nur zu einer Erweiterung der partiellen - und damit vollständigen - Ausgaben führen kann, nicht jedoch zu einer Änderung der bisher erfolgten partiellen Ausgaben. Daß diese Klasse von Prozessen gerade für den in dieser Arbeit vorgestellen Ansatz von Interesse ist, wird in [Pan93] gezeigt: werden die in TCSP eingeführten Operatoren zur Konstruktion von Prozessen entsprechend eingesetzt, um asynchrone Prozesse zu definieren, so wird damit die Klasse monotoner Prozesse beschrieben.[12] Darüberhinaus existieren jedoch auch Prozesse, die diese Eigenschaft nicht erfüllen. Eine spezielle Klasse solcher Prozesse wird im nächsten Abschnitt eingeführt.

Für monotone Prozesse läßt sich die Eingabe bereits durch die beiden folgenden Anforderungen charakterisieren:

- Die leere Sequenz ist immer eine mögliche partielle Ausgabe für die leere Sequenz von Eingaben.
- Ist zu einer Eingabe i_1 eine partielle Ausgabe o möglich, so ist diese partielle Ausgabe auch nach einer Erweiterung der Eingabe um i_2 zu $i_1 \circ i_2$ möglich.

Diese informellen Anforderungen lassen sich formal darstellen als

$$P(\langle\rangle, \langle\rangle) \tag{2.5}$$

$$P(i_1, o) \Rightarrow P(i_1 \circ i_2, o) \tag{2.6}$$

Dabei ist die erste Anforderung unabhängig von der Charakterisierung monotoner Prozesse und muß stets für alle partiellen Abläufe gelten. Offensichtlich folgt die Forderung der Totalität 2.3 sofort aus den Anforderungen 2.5 und 2.6. Insgesamt ergibt sich damit die folgende Definition.

Definition 2.2.7 (Monotoner Prozeß) Eine relationale Beschreibung (P, C) eines Prozesses mit Eingabealphabet I und Ausgabealphabet O heißt *monoton*, wenn sie die Anforderungen 2.4, 2.5 und 2.6 erfüllt. ∘

Dabei steht die Verwendung des Begriffs "monoton" in direktem Bezug zur klassischen Verwendung im Kontext von Funktionen:

[12] Zur Definition dieser Prozesse für den synchronen Fall siehe auch Anhang A.

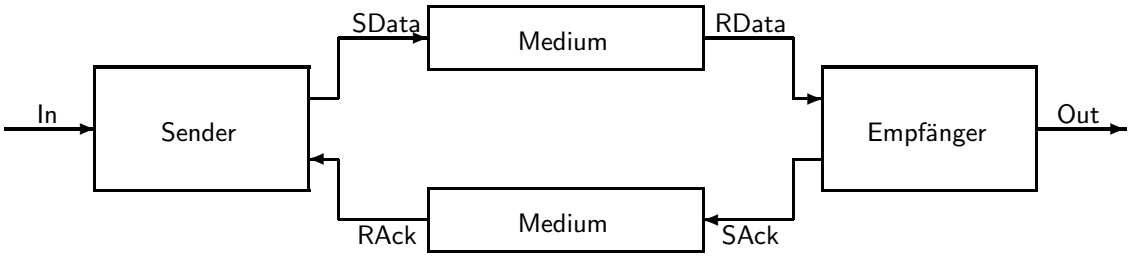

Abbildung 2.3: Aufbau eines "Alternating Bit"-Systems

Hilfsatz 2.2.1 (Monotonie) Sei (P, C) eine relationale monotone Prozeßbeschreibung mit Eingabealphabet I und Ausgabealphabet O. Dann gilt

$$P(i_1, o_1) \wedge C(i_1, o_1) \wedge i_1 \sqsubseteq i_2 \Rightarrow (\exists o_2.P(i_2, o_2) \wedge C(i_2, o_2) \wedge o_1 \sqsubseteq o_2)$$

△

Der Beweis folgt unmittelbar aus 2.6 und 2.4:

Beweis 2.2.1 (Hilfsatz 2.2.1)

$$\begin{array}{ll} & P(i_1, o_1) \wedge C(i_1, o_1) \wedge i_1 \sqsubseteq i_2 \\ \Longrightarrow & [\text{Def. } \sqsubseteq] \\ & \exists i.i_2 = i_1 \circ i \wedge P(i_1, o_1) \wedge C(i_1, o_1) \\ \Longrightarrow & [2.6] \\ & \exists i.i_2 = i_1 \circ i \wedge P(i_1 \circ i, o_1) \\ \Longrightarrow & [2.4] \\ & \exists i, o.i_2 = i_1 \circ i \wedge P(i_1 \circ i, o_1 \circ o) \wedge C(i_1 \circ i, o_1 \circ o) \\ \Longrightarrow & [\text{Def. } \sqsubseteq,\ o_2 = o_1 \circ o] \\ & \exists o_2.P(i_2, o_2) \wedge C(i_2, o_2) \wedge o_1 \sqsubseteq o_2 \end{array}$$

□

Anschaulich stellt die Aussage von Hilfsatz 2.2.1 genau die am Anfang des Abschnitts informell gegebene Definition des monotonen Prozesses dar, daß zusätzliche Eingaben lediglich die vollständige Ausgabe des Prozesses verlängern. Die Praxisrelevanz monotoner relationaler Prozeßbeschreibungen zeigt das nachfolgende Beispiel eines Prozesses aus dem Protokollbereich.

Beispiel 2.2.3 (Empfänger) Im folgenden wird die Empfängerkomponente aus dem "Alternating Bit Protocol" (siehe z.B. [DS92a]) verwendet, um die relationale Beschreibung monotoner Prozesse zu verdeutlichen. Insbesondere wird hier auch gezeigt, wie sich nichtsequentielle Prozesse, also Prozesse mit paralleler Ausgabe, mit dem eingeführten relationalen Schema beschreiben lassen.

Die Empfängerkomponente arbeitet nach dem folgenden Prinzip:

- Sie nimmt auf ihrem Eingabekanal RData Pakete bestehend aus dem eigentlichen Datenanteil sowie einem Sequenzbit entgegen.

- Unterscheidet sich das aktuelle Sequenzbit vom Sequenzbit des zuletzt empfangenen Pakets, so wird der Datenanteil auf dem Ausgabekanal Out ausgegeben.
- Das Sequenzbit des Pakets wird in jedem Fall als Bestätigung an die Senderkomponente über den Ausgabekanal SAck zurückgeschickt.

Zusammen mit einer Senderkomponente wie aus Beispiel 2.2.4, die Datenpakete mit einem alternierenden Sequenzbit versieht und jedes Paket bis zum Empfang einer Bestätigung vom Empfängerprozeß wiederholt sendet, läßt sich so eine sichere Datenübertragung über ein verlustbehaftetes, aber faires Medium realisieren. Der Aufbau eines solchen Systems wird in Abbildung 2.3 dargestellt.

Zur Definition der Empfängerkomponente wird eine stetige Funktion dup zur Elimination von direkt aufeinanderfolgenden Duplikaten von Nachrichten verwendet:

$$
\begin{array}{lcl}
\mathsf{dup}(\langle\rangle) & \stackrel{\text{def}}{=} & \langle\rangle \\
\mathsf{dup}(a \circ s) & \stackrel{\text{def}}{=} & a \circ \mathsf{hdup}(a, s) \\
\mathsf{hdup}(a, \langle\rangle) & \stackrel{\text{def}}{=} & \langle\rangle \\
\mathsf{hdup}(a, b \circ s) & \stackrel{\text{def}}{=} & \begin{cases} \mathsf{hdup}(a, s) & \text{falls } a = b \\ b \circ \mathsf{hdup}(b, s) & \text{falls } a \neq b \end{cases}
\end{array}
$$

Zusammen mit einer vorgegebenen Datenmenge D werden die folgenden Nachrichtenmengen definieren:

- $\mathbb{D} = \{0, 1\}$ ist die Menge der Sequenzbits.
- $U = \mathbb{D} \times D$ ist die Menge der Datenpakete.

Damit ergeben sich die folgenden Kanalalphabete für die Empfängerkomponente:

- $RData = \{\mathsf{RData}\} \times U$
- $Out = \{\mathsf{Out}\} \times D$
- $SAck = \{\mathsf{SAck}\} \times \mathbb{D}$

Als Gesamtausgabealphabet für den Empfänger ergibt sich weiterhin $O = Out \cup SAck$. Weiterhin werden die folgenden Hilfsfunktionen als punktweise Erweiterung der Selektorfunktionen eingeführt:

- $\mathsf{data}^* : RData^\omega \to D^\omega$ mit[13]
  $\mathsf{data}(\mathsf{RData}.b.d) = d$
- $\mathsf{data}^* : Out^\omega \to D^\omega$ mit
  $\mathsf{data}(\mathsf{Out}.d) = d$
- $\mathsf{bit}^* : RData^\omega \to \mathbb{D}^\omega$ mit
  $\mathsf{bit}(\mathsf{RData}.b.d) = b$

[13]Statt der Klammerschreibweise (x, y) wird für das Produkt zweier Elemente die Infixnotation $x.y$ verwendet.

- $\mathsf{bit}^* : SAck^\omega \to \mathbb{D}^\omega$ mit
  $\mathsf{bit}(\mathsf{SAck}.b) = b$

Mit diesen Definitionen wird nun das Verhalten des Empfängers mit der relationalen Beschreibung der partiellen und totalen Ausgaben definiert. Dabei wird der Empfänger durch zwei Prozesse dargestellt, jeweils einen für jeden Ausgabekanal.

$$\begin{array}{lcl} P_{\mathsf{Out}}(i,o) & \stackrel{\text{def}}{=} & \mathsf{data}^*(o) \sqsubseteq \mathsf{data}^*(\mathsf{dup}(i)) \\ P_{\mathsf{SAck}}(i,o) & \stackrel{\text{def}}{=} & \mathsf{bit}^*(o) \sqsubseteq \mathsf{bit}^*(i) \\ C_{\mathsf{Out}}(i,o) & \stackrel{\text{def}}{=} & \mathsf{data}^*(o) = \mathsf{data}^*(\mathsf{dup}(i)) \\ C_{\mathsf{SAck}}(i,o) & \stackrel{\text{def}}{=} & \mathsf{bit}^*(o) = \mathsf{bit}^*(i) \end{array}$$

Offensichtlich gelten 2.4, sowie 2.5 und 2.6. Damit sind obige Relationen jeweils monotone relationale Prozeßbeschreibungen. Entsprechend der Definition der Spuren einer relationalen Beschreibung 2.2.6 sowie der Definition der Parallelkomposition 2.2.3 lassen sich die Spuren der relationalen Darstellung des Empfangsprozesses $T_{P_{\mathsf{Out}},C_{\mathsf{Out}}} \wedge T_{P_{\mathsf{SAck}},C_{\mathsf{Out}}}$ charakterisieren mittels

$$\begin{array}{lll} T_{P_{\mathsf{Out}} \wedge P_{\mathsf{SAck}}, C_{\mathsf{Out}} \wedge C_{\mathsf{Out}}}(t) = & (\forall s \sqsubseteq t.\ P_{\mathsf{Out}}(RData@t, Out@t) \wedge & \\ & \qquad P_{\mathsf{SAck}}(RData@t, SAck@t)) \wedge & \\ & C_{\mathsf{Out}}(RData@t, Out@t) \wedge & \\ & C_{\mathsf{SAck}}(RData@t, SAck@t) & \end{array}$$

◇

Unterbrechbare Prozesse

Wie bereits erwähnt, muß die Monotonieforderung nicht für alle asynchronen Prozesse erfüllt sein. Insbesondere gilt sie nicht für Prozesse, für die Unterbrechungsbehandlungen definiert werden können. Darunter sind solche Prozesse zu verstehen, bei denen das Eintreffen einer neuen Nachricht die durch die bisher empfangenen Eingaben bestimmte, aber noch nicht vollständig erzeugte Ausgabe verändern kann: anstatt die Ausgabe wie durch die bisherige Eingabe vorgegeben zu vervollständigen, kann nun eine andere Erweiterung der Ausgabe vorgenommen werden. Für diese Klasse der *unterbrechbaren Prozesse* gilt nicht mehr, daß eine Erweiterung der Eingabe die Erweiterung der Ausgabe bewirkt. In Beispiel 2.2.4 wird ein solcher unterbrechbarer Prozeß vorgestellt.

Aufbauend auf den Anforderungen der Sequentialität und der Realisierbarkeit wird, ähnlich wie im Fall der monotonen Prozesse, eine geeignete Anforderung an das Eingabeverhalten eines unterbrechbaren Prozesses definiert. Dazu wird die Anforderung 2.6 durch eine schwächere Anforderung ersetzt. Für unterbrechbare Prozesse wird nur noch gefordert, daß mit einer partiellen Ausgabe o zu einer Eingabe i_1 diese Ausgabe auch zu jeder Erweiterung der Eingabe um i_2 zu $i_1 \circ i_2$ möglich ist, *falls es sich um eine endliche partielle Ausgabe handelt.* Diese Anforderung beschreibt dabei das Verhalten eines asynchronen unterbrechbaren Prozesses, da

- eine beliebige aber endliche Verzögerung der Ausgabe eines Prozesses gegenüber der durch die Umgebung kontrollierten Eingabe möglich ist;
- diese Eingabebereitschaft aber auf endliche Ausgaben beschränkt bleibt; insbesondere ist es möglich, daß das vollständige unendliche Ausgabeverhalten durch eine Erweiterung der Eingabe verändert wird.

Formal läßt sich diese Anforderung definieren als

$$\forall o \in O^*.P(i_1, o) \Rightarrow P(i_1 \circ i_2, o) \tag{2.7}$$

Daß diese Anforderung eine Abschwächung der Anforderung 2.6 ist, ist offensichtlich. Insgesamt ergibt sich damit die Definition der unterbrechbaren Prozesse.

Definition 2.2.8 (Unterbrechbarer Prozeß) Eine relationale Beschreibung (P, C) eines Prozesses mit Eingabealphabet I und Ausgabealphabet O heißt relationale Beschreibung eines *unterbrechbaren* Prozesses, wenn sie die Anforderungen 2.4, 2.5 und 2.7 erfüllt. ∘

Ähnlich wie im Fall der monotonen Prozesse wird hier die Anforderung 2.3 fallengelassen, da die Anforderung 2.7 eine Verschärfung dieser Anforderung darstellt. Offensichtlich sind alle monotonen Prozeßbeschreibungen auch Beschreibungen unterbrechbarer Prozesse. Die Begründung für die Einführung der Klasse der unterbrechbaren Prozesse ist vergleichbar mit dem Fall der monotonen Prozesse. Um auch diese Klasse durch entsprechende Prozeßkonstruktoren definieren zu können, muß lediglich eine asynchronen Variante des in Anhang A in Definition A.5.1 eingeführten Unterbrechungsoperators für synchrone Prozesse eingeführt werden. Daß unterbrechbare Prozesse auch in der Praxis eine wichtige Rolle spielen, zeigt das folgende Beispiel.

Beispiel 2.2.4 (Sender) Wie bereits in Beispiel 2.2.3 angesprochen, wird zur Realisierung des alternierenden Bitprotokolls ein entsprechender Senderprozeß zum Empfängerprozeß benötigt. Im folgenden wird nun eine geeignete Senderkomponente mit den in Abbildung 2.3 gezeigten Ein- und Ausgabekanälen In und RAck sowie SData definiert. Dazu werden aus Beispiel 2.2.3 die Funktionen dup, bit^* und data^* angepaßt an diese Kanäle übernommen. Weiterhin werden die folgenden Kanalalphabete definiert:

- $SData = \{\mathsf{SData}\} \times U$
- $In = \{\mathsf{In}\} \times D$
- $RAck = \{\mathsf{RAck}\} \times \mathbb{D}$

Die Aufgabe des Sender besteht darin, auf In empfangene Daten mit einem Sequenzbit zu versehen und dieses Paket über SData solange an den Empfänger zuzustellen, bis von diesem auf RAck eine Bestätigung für dieses Paket in Form des enthaltenen Sequenzbits eingetroffen ist. Für teilweise oder vollständige Ausgaben des Sender muß daher gelten:

- Es werden nur Datenanteile verschickt, die als Nachrichten empfangen wurden.

- In den verschickten Datenpaketen ist höchstens eine Eingabe mehr enthalten als passende Bestätigungen empfangen werden.
- Die Übertragung eines bereits bestätigten Pakets wird schließlich eingestellt.

Die letzte Anforderung kann in einem ungezeiteten Modell[14] wie dem Spurmodell mit einem Prozeß für die Eingabepufferung nicht verschärft werden, da wegen der möglichen endlichen Verzögerung durch den Eingabepuffer eine empfangene Nachricht nicht sofort dem Sender zugestellt werden muß. Diese drei informellen Anforderungen lassen sich als Charakterisierung partieller Abläufe mittels $P : (In \cup RAck)^\omega \times SData^\omega \to \mathbb{B}$ mit

$$\begin{aligned} P(i,o) \stackrel{\text{def}}{=}\ & \mathsf{data}^*(\mathsf{dup}(o)) \sqsubseteq \mathsf{data}^*(In \copyright i) \wedge \\ & \#\,\mathsf{dup}(o) \leq \#\,\mathsf{dup}(RAck \copyright i) + 1 \wedge \\ & (\#\,\mathsf{dup}(o) > \#\,\mathsf{dup}(RAck \copyright i) \vee o \in O^*) \end{aligned}$$

darstellen. Von einem vollständigen Ablauf wird darüberhinaus erwartet, daß

- entweder alle Eingaben übertragen wurden oder
- der Sender die Übertragung niemals einstellt.

Entsprechend werden die vollständigen Abläufe durch $C : (In \cup RAck)^\omega \times SData^\omega \to \mathbb{B}$ charakterisiert mit

$$C(i,o) \stackrel{\text{def}}{=} \#In \copyright i = \#\,\mathsf{dup}(o) \vee o \notin O^*$$

Insgesamt wird der Senderprozeß entsprechend Definition 2.2.6 durch die Spurmenge der relationalen Beschreibung $T_{P,C} : (In \cup RAck \cup SData)^\omega \to \mathbb{B}$ beschrieben.

Offensichtlich ist der mittels $T_{P,C}(t)$ beschriebene Prozeß nicht monoton, da zwar

$$P(\mathsf{In}.d, \mathsf{SData}.(1.d)^\infty)$$

aber nicht

$$P(\mathsf{In}.d \circ \mathsf{RAck}.1, \mathsf{SData}.(1.d)^\infty)$$

sondern nur

$$P(\mathsf{In}.d \circ \mathsf{RAck}.1, \mathsf{SData}.(1.d)^n)$$

für ein beliebiges $n \in \mathbb{N}$ gilt. Dies folgt unmittelbar aus

$$\#\,\mathsf{dup}(\mathsf{SData}.(1.d)^\infty) = \#\,\mathsf{dup}(\mathsf{RAck}.1)$$

sowie

$$\mathsf{SData}.(1.d)^\infty \notin SData^*$$

und

$$\mathsf{SData}.(1.d)^n \in SData^*$$

[14] Unter einem ungezeiteten Modell wird hier ein Modell verstanden, das nur qualitative aber keine quantitativen Aussagen über die zeitliche Beziehung von Aktionen erlaubt.

für ein beliebiges $n \in \mathbb{N}$ gilt.

Der Nachweis, daß die relationale Darstellung der partiellen Abläufe des Senders den Anforderungen asynchroner Systeme genügt, folgt unmittelbar. Offensichtlich gilt 2.4, 2.5 sowie 2.7. Damit sind obige Relationen jeweils relationale Beschreibungen unterbrechbarer Prozesse. ◇

2.3 Failuresemantik

Wie bereits in Abschnitt 2.2 angesprochen, wurde ursprünglich versucht, synchron kommunizierende Systeme durch Spuren zu beschreiben (vgl. [Hoa80]). Es zeigt sich jedoch, daß diese Modellierungstechnik nicht ausreicht, um adäquate kompositionale Beschreibungen zu erhalten (siehe [Old85, Old86]). Erst die Verwendung einer komplexeren Semantik lieferte die gewünschten Resultate (siehe z.B. [BHR84], [AWO85]).[15] Dazu wird das Spurmodell um eine zusätzliche Komponente zum *Failuremodell* erweitert.

Diese komplexere Semantik basiert in ihrer einfachsten Form auf der Darstellung von Prozessen als Menge von *Failures*. Diese werden in [Hoa85] folgendermaßen eingeführt:

> If (s, X) is a failure of P, this means that P can engage in the sequence of events recorded by s, and then refuse to do anything more in spite of the fact that its environment is prepared to engage in any of the events of X.

Damit ist also ein Failureelement ein Paar bestehend aus einer Spurkomponente und einer Mengenkomponente. Die Spurkomponente beschreibt die endliche Folge von Aktionen, nach deren beobachtetem Ablauf der beschriebene Prozeß sich weigern kann, Aktionen aus der Mengenkomponente auszuführen. Die Mengenkomponente wird auch oft als *Refusalmenge* bezeichnet.

Durch die Beschränkung auf die Beschreibung endlichen Verhaltens ergibt sich ein Problem, das bei der Beschreibung der asynchron kommunizierenden Systeme mittels unendlicher Spuren nicht auftritt, nämlich die sogenannte *Divergenz*. Unter einem divergenten System wird dabei ein solches System verstanden, das in der Lage ist, eine unbeschränkte Anzahl von verborgenen ("internen") Aktionen auszuführen, ohne gezwungen zu sein, eine sichtbare ("externe") Aktion ausführen zu müssen. Solche Systeme sind mittels der hier beschriebenen Semantik nicht geeignet zu beschreiben. Um dieses Problem zu vermeiden, wurde die "Failure-Divergence-Semantik" eingeführt. Da diese Problematik für das Verständnis der Systemmodellierung nicht von grundsätzlicher Bedeutung ist, wird sie erst später behandelt.

Hier soll zunächst der Einfachheit halber, unter der Beschränkung auf die Failuremodellierung, die Definition gemäß [Hoa85] verwendet werden:

Definition 2.3.1 (Synchroner Prozeß) Ein *synchroner Prozeß* ist ein Tupel (A, F) mit

[15] Für einen Überblick über die Entwicklung von CSP siehe auch [HJ95].

- A ist eine Menge von Aktionssymbolen
- $F \subseteq A^* \times \mathbb{P}(A)$

mit den Nebenbedingungen

1. $(\langle\rangle, \emptyset) \in F$
2. $(s \circ t, X) \in F \Rightarrow (s, \emptyset) \in F$
3. $(s, Y) \in F \wedge X \subseteq Y \Rightarrow (s, X) \in F$
4. $(s, X) \in F \wedge a \in A \Rightarrow ((s \circ a, \emptyset) \in F \vee (s, X \cup \{a\}) \in F)$

A heißt *das Alphabet des Prozesses.* ∘

Das Alphabet eines Prozesses P wird im folgenden auch mit αP gekennzeichnet. Weiterhin wird im folgenden für beliebige Paare (A, F), bei denen F nicht die Nebenbedingungen erfüllen muß, der Begriff *Failuremenge mit Alphabet* verwendet.

2.3.1 Kompositionsoperatoren

In Abschnitt 2.2.3 wurden den syntaktischen Kompositionoperatoren "$\|$" und "$\backslash$" entsprechende semantische Operationen zugeordnet, die auf asynchronen Prozessen operieren. In entsprechender Weise werden in diesem Abschnitt den oben definierten synchronen Prozessen die dazugehörigen semantischen Kompositionsoperatoren zugewiesen.

Parallelkomposition

Wie bereits in Abschnitt 2.2.3 besprochen, beschreibt der Operator "$\|$" die synchronisierende Komposition zweier Prozesse. Daher verhält sich die Parallelkomposition innerhalb des Failuremodells hinsichtlich der Spuranteile der Failurepaare entsprechend zur Spursemantik. Bezüglich der Refusalmengen gilt, daß das Gesamtsystem $P_1 \parallel P_2$ alle Aktionen ablehnen kann, die entweder von P_1 oder P_2 abgelehnt werden können.

Im Gegensatz zum Spurmodell ist die Parallelkomposition des Failuremodells total, d.h. für alle Prozesse P_1 und P_2 unabhängig von ihren Alphabeten definiert. Dabei enthält das Alphabet des Gesamtsystems alle Aktionen aus den Alphabeten von P_1 und P_2.

Definition 2.3.2 (Parallelkomposition) Seien P_1 und P_2 zwei synchrone Prozesse mit $P_1 = (A_1, F_1)$ und $P_2 = (A_2, F_2)$. Dann wird die Parallelkomposition von P_1 und P_2 definiert als der Prozeß mit

$$P_1 \parallel P_2 \stackrel{\text{def}}{=} (A_1 \cup A_2, F)$$

wobei

$$F = \{(t, R_1 \cup R_2) \mid \ (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}$$

∘

Entsprechend läßt sich die Parallelkomposition auch für Failuremengen mit Alphabet definieren.

Abstraktion

In [Hoa85] wird die Bedeutung der Abstraktion von internen Aktionen bei synchronen Prozessen folgendermaßen umrissen:

> (We) ... wish to conceal all occurrences of actions internal to the mechanism. In fact, we want these actions to occur automatically and instantaneously as soon as they can, without being observed or controlled by the environment of the process.

Wie bereits bei der Parallelkomposition festgestellt wurde, verhalten sich auch die Spuranteile der Failurepaare bei der Abstraktion ganz entsprechend zur Spursemantik. Während jedoch die asynchronen Prozesse durch vollständige Abläufe charakterisiert werden, stellen die Failurepaare auch "unvollständige" Abläufe eines Prozesses dar; entsprechend obiger Vorstellung werden also nur solche Abläufe berücksichtigt, bei denen keine der zu verbergenden Aktionen mehr ausgeführt werden kann. Auch hier enthält das Alphabet des abstrahierten Systems alle Aktionen des ursprünglichen Systems abzüglich aller zu verbergenden.

Definition 2.3.3 (Abstraktion) Sei P ein synchroner Prozeß mit $P = (A, F)$. Dann wird die Abstraktion von P hinsichtlich von Aktionen aus A' definiert als der Prozeß mit

$$(P \backslash A') \stackrel{\text{def}}{=} (A \backslash A', F')$$

wobei

$$F' = \{((A \backslash A') \copyright t, R \backslash A') \mid (t, R \cup A') \in F\}$$

∘

Auch die Abstraktion läßt sich entsprechend für Failuremengen mit Alphabet definieren.

2.3.2 Eigenschaften der Failuresemantik

In der weiteren Arbeit werden einige Eigenschaften der eingeführten Semantiken benötigt, die in diesem Abschnitt eingeführt werden. Dazu kann auf die Tatsache zurückgegriffen werden, daß TCSP nicht nur eine denotationelle Semantik besitzt, sondern auch als Prozeßalgebra interpretiert wird. Dazu werden in [Hoa85] für TCSP-Operatoren eine Vielzahl von Eigenschaften in Form von algebraischen Gesetzen auf der syntaktischen Ebene für diese definiert. Da aber die Failuresemantik von TCSP korrekt hinsichtlich der Prozeßalgebra von TCSP ist, gelten diese Aussagen auch für die denotationelle Semantik. Tatsächlich gelten diese Aussagen nicht nur für die Prozesse der Failuresemantik, sondern für beliebige Failuremengen. Da der Nachweis dieser Eigenschaften bekannt und für das weitere Vorgehen nicht von Interesse ist, wird hier auf die Beweise dieser Eigenschaften verzichtet. Die Aussagen wird direkt über Failuremengen anstatt über Prozessen definiert.

Satz 2.3.1 (Assoziativität der Parallelkomposition) Die Parallelkomposition ist assoziativ, d.h. es gilt für beliebige $P_1, P_2, P_3 \subseteq A^* \times \mathbb{P}(A)$

$$(P_1 \parallel P_2) \parallel P_3 = P_1 \parallel (P_2 \parallel P_3)$$

•

Diese Eigenschaft der Parallelkomposition erlaubt es, ein System, das aus mehreren parallelkomponierten Komponenten besteht, eine beliebige Anordnung der Komponenten zuzuordnen. Diese Eigenschaft wird für den in Kapitel 4 entwickelten Teil der Vorgehensweise notwendig sein.

Satz 2.3.2 (Vertauschung von Abstraktion und Parallelkomposition) Für beliebige $P_1 \subseteq A^* \times \mathbb{P}(A), A_1 \subseteq A, P_2 \subseteq (A \backslash A_1)^* \times \mathbb{P}(A \backslash A_1)$ gilt

$$(P_1 \backslash A_1) \parallel P_2 = (P_1 \parallel P_2) \backslash A_1$$

•

Diese Eigenschaft von Parallelkomposition und Abstraktion erlaubt es, die Abstraktion von einer auf mehrere Komponenten auszuweiten, falls die weiteren Komponenten nicht von der Abstraktion betroffen sind. Auch diese Eigenschaft wird für das in dieser Arbeit vorgeschlagene Verfahren von Bedeutung sein, wie sich in Abschnitt 2.6.3 zeigen wird.

2.3.3 Beispiel

Im folgenden Beispiel werden die beiden Kompositionsoperatoren "$\parallel$" und "$\backslash$" am Beispiel der Komposition zweier unbeschränkter Puffer demonstriert.

Beispiel 2.3.1 (Unbeschränkte Puffer) Die Definition der Aktionsmengen und Abbildungen werden wie in Beispiel 2.2.1 verwendet. Damit wird der Puffer P_1 definiert als

$$P_1 \stackrel{\text{def}}{=} (I_1 \cup O_1, F_1)$$

$$\begin{aligned} F_1 = \{(t, R) \mid\ & (\forall s \sqsubseteq t.\, \mathsf{data}^*(O_1 © t) \sqsubseteq \mathsf{data}^*(I_1 © t)) \wedge \\ & (I_1 \cap R = \emptyset) \wedge \\ & (\forall o \in O_1.\, \mathsf{data}^*((O_1 © t) \circ o) \sqsubseteq \mathsf{data}^*(I_1 © t) \Rightarrow o \notin R)\} \end{aligned}$$

Ähnlich wie im Fall der Modellierung mittels Spuren, wie in 2.2.4 beschrieben, bedeutet dies, daß jede ausgegebene Nachricht vorher als Eingabe empfangen worden sein muß. Weiterhin wird gefordert, daß eine Eingabe nie verweigert werden kann; steht noch eine Eingabe zur Ausgabe an, so kann diese nicht verweigert werden. Der Puffer P_2 wird analog beschrieben:

$$P_2 \stackrel{\text{def}}{=} (I_2 \cup O_2, F_2)$$

wobei I_2,O_2 und T_2 entsprechend definiert sind mit $I_2 = O_1$.

Werden nun, ähnlich wie in 2.2.4 und wie in Abbildung 2.1 dargestellt, die beiden Prozesse parallelkomponiert, so ist

$$P_1 \parallel P_2 = (I_1 \cup O_1 \cup O_2, F)$$

wobei

$$\begin{aligned} F = \{(t,R) \mid\ & (\forall s \sqsubseteq t.\ \mathsf{data}^*(O_1 © s) \sqsubseteq \mathsf{data}^*(I_1 © s) \wedge \\ & \qquad \mathsf{data}^*(O_2 © s) \sqsubseteq \mathsf{data}^*(O_1 © s)) \wedge \\ & (I_1 \cap R = \emptyset) \wedge \\ & (\forall o \in O_1.\, \mathsf{data}^*((O_1 © t) \circ o) \sqsubseteq \mathsf{data}^*(I_1 © t) \Rightarrow o \notin R) \wedge \\ & (\forall o \in O_2.\, \mathsf{data}^*((O_2 © t)) \circ o \sqsubseteq \mathsf{data}^*(O_1 © t) \Rightarrow o \notin R)\} \end{aligned}$$

Das Gesamtsystem $(P_1 \parallel P_2)\backslash I_2$, das wie in Abbildung 2.2 dargestellt durch Verbergen des Kanals mit allen Aktionen aus O_1 (bzw. I_2) entsteht, ist dementsprechend definiert als

$$(P_1 \parallel P_2)\backslash I_2 = (I_1 \cup O_2, F)$$

wobei

$$\begin{aligned} F = \{(t,R) \mid\ & (\forall s \sqsubseteq t.\, \mathsf{data}^*(O_2 © s) \sqsubseteq \mathsf{data}^*(I_1 © s)) \\ & (I_1 \cap R = \emptyset) \wedge \\ & (\forall o \in O_2.\, \mathsf{data}^*((O_2 © t) \circ o) \sqsubseteq \mathsf{data}^*(I_1 © t) \Rightarrow o \notin R)\} \end{aligned}$$

Damit liefert auch hier die Komposition zweier unbeschränkter Puffer einen Puffer mit gleichem Verhalten. ◇

2.4 sat-Kalkül

Bei der Systementwicklung in der hier vorgestellten Arbeit erfolgt die Beschreibung des asynchron bzw. synchron kommunizierenden Systems in Form von Spur- bzw. Failureprädikaten. Die in Abschnitt 2.2 bzw. 2.3 angegebenen Semantiken definieren die beiden Kompositionsoperatoren Parallelkomposition und Abstraktion direkt auf der semantischen Ebene. Um den Entwurfsprozeß zu vereinheitlichen, empfiehlt es sich daher, entsprechende Regeln für die Kompositionsoperatoren zu definieren, die es erlauben, Nachweise von Eigenschaften direkt auf der Ebene prädikativer Beschreibungen zu führen. Damit kann dann auf der Ebene dieser Beschreibungen, im folgenden auch oft Spezifikationen genannt, gearbeitet werden, ohne auf die semantische Ebene wechseln zu müssen. Durch Anwendung der Regeln kann dann gezeigt werden, daß aus den Eigenschaften der Komponenten eines Systems die Eigenschaften des Systems hergeleitet werden, oder anders ausgedrückt, die Spezifikationen der Komponenten die Spezifikation des Systems realisiert.

2.4.1 sat-Relation

Die sat-Relation beschreibt den Bezug zwischen den syntaktischen Bezeichnern eines zusammengesetzten Systems und deren semantischen Eigenschaften. Sie stellt damit den

Zusammenhang zwischen der Spezifikation eines zusammengesetzten Systems und der Spezifikation der Komponenten her. Da, je nach Wahl des Modells, eine Spezifikation ein Spur- bzw. ein Failureprädikat darstellt, werden zwei Definitionen benötigt. Die Spurdefinition lautet:

Definition 2.4.1 (Spur-sat**-Relation)** P bezeichne einen Prozeß (I, O, T) und S eine Spezifikation der Art $S : (I \cup O)^\omega \to \mathbb{B}$. Dann *erfüllt P die Spezifikation S*, kurz

$$P \text{ sat } S$$

wenn

$$\forall t \in T.S(t)$$

gilt. ○

Die der Spurform entsprechende Definition für den Failurefall lautet:

Definition 2.4.2 (Failure-sat**-Relation)** P bezeichne einen Prozeß (A, F), und S eine Spezifikation der Art $S : A^* \times \mathbb{P}(A) \to \mathbb{B}$. Dann *erfüllt P die Spezifikation S*, kurz

$$P \text{ sat } S$$

wenn

$$\forall (t, R) \in F.S(t, R)$$

gilt. ○

Die Schreibweise P sat S und damit auch Definition 2.4.2 werden im folgenden auch auf Failuremengen mit Alphabet (A, F) angewendet.

Wie oben besprochen, wird der sat-Kalkül verwendet, um auf syntaktischer Ebene nachzuweisen, daß mit den Anforderungen an die Komponenten eines System die Anforderungen an das System implementiert werden können. Um diese Implementierungsbeziehung ausdrücken zu können, wird die folgende allgemeine Regel eingeführt, die sowohl für Spur- als auch für Failurespezifikationen gilt.

Definition 2.4.3 (Implikationsregel) Sei $S : A^\omega \to \mathbb{B}$ bzw. $S : A^* \times \mathbb{P}(A) \to \mathbb{B}$ eine Spur- bzw. Failurespezifikation. Dann gilt

$$\frac{P \text{ sat } S \qquad S \Rightarrow S'}{P \text{ sat } S'}$$

○

2.4.2 Spurregeln

Um die Definition der Kompositionsoperatoren aus Abschnitt 2.2 auf der Ebene des sat-Kalküls zur Verfügung zu stellten, muß für jeden Operator eine entsprechende Regel eingeführt werden. Diese Regeln erlauben es, aus den Spezifikationen der Komponenten eine Spezifikation für das System herzuleiten. Dabei wird die stärkste Spezifikation hergeleitet, die entsprechend der Spursemantik vom System erfüllt wird.

Parallelkomposition

Die folgende Definition führt eine entsprechende Regel für die Parallelkomposition nach Definition 2.2.3 ein.

Definition 2.4.4 (Spur-Par-Regel) Seien S_1 und S_2 Spurspezifikationen der Art $S_1 : A_1^\omega \to \mathbb{B}$ und $S_2 : A_2^\omega \to \mathbb{B}$. Dann gilt

$$\frac{P_1 \text{ sat } S_1 \quad P_2 \text{ sat } S_2}{P_1 \parallel P_2 \text{ sat } S_1(A_1 \copyright t) \wedge S_2(A_2 \copyright t)}$$

∘

Abstraktion

Die folgende Definition führt eine entsprechende Regel für die Abstraktion nach Definition 2.2.4 ein.

Definition 2.4.5 (Spur-Abs-Regel) Sei S eine Spurspezifikation der Art $S : A^\omega \to \mathbb{B}$. Dann gilt

$$\frac{P \text{ sat } S}{P \backslash A' \text{ sat } \exists t' \in A^\omega . S(t') \wedge (A \backslash A') \copyright t' = t}$$

∘

Korrektheit

Mit den Regeln 2.4.3, 2.4.4 und 2.4.5 lassen sich nun Herleitungen im sat-Kalkül definieren. Eine Aussage ist herleitbar im Spur-sat-Kalkül, wenn sie durch Anwendung der Regeln 2.4.3, 2.4.4 und 2.4.5 hergeleitet werden kann.

Da die Herleitungen im sat-Kalkül auf der Ebene der Prozeßterme und der Spezifikationen und damit ohne direkten Bezug zu den Prozessen des Spurmodells stattfinden, ist es wesentlich, daß die so gewonnenen Aussagen korrekt sind. Eine Herleitung soll dabei als korrekt bezeichnet werden, wenn aus der Herleitung von Systemeigenschaft aus Komponenteneigenschaften im sat-Kalkül auch die Herleitung auf der semantischen Ebene für beliebige Instantiierungen der Prozeßterme folgt. Diese Tatsache drückt der folgende Satz aus.

Satz 2.4.1 (Korrektheit des Spur-sat-Kalküls) Ist zu den Spezifikationen $T_1, \ldots, T_n$ und T zu Prozessen mit den jeweiligen Ein- und Ausgabealphabeten $I_1, \ldots, I_n$ und I bzw. $O_1, \ldots, O_n$ und O die Aussage

$$\frac{\begin{array}{l} P_1 \text{ sat } T_1 \\ \quad \vdots \\ P_n \text{ sat } T_n \end{array}}{P \text{ sat } T}$$

im sat-Kalkül herleitbar, so gilt

$$P_1 \text{ sat } T_1 \wedge \ldots \wedge P_n \text{ sat } T_n \Rightarrow P \text{ sat } T$$

für beliebige Prozesse $P_1, \ldots, P_n$ und P mit den Ein- und Ausgabealphabeten $I_1, \ldots, I_n$ und I bzw. $O_1, \ldots, O_n$ und O. •

Beweis 2.4.1 (Satz 2.4.1) Für die einzelnen Regeln 2.4.3, 2.4.4 und 2.4.5 folgt die Korrektheit der jeweiligen Einzelschrittherleitung unmittelbar aus der Definition 2.4.1 sowie den Definitionen 2.2.3 bzw. 2.2.4. Entsprechend den Regeln der Prädikatenlogik folgt dann aus der Korrektheit der Einzelschritte einer mehrschrittigen Herleitung auch die Korrektheit der mehrschrittigen Herleitung. □

2.4.3 Failureregeln

Analog zum Fall des Spurmodells in Abschnitt 2.4.2 lassen sich entsprechende Regeln für das Failuremodell definieren, um aus den Spezifikation von Komponenten die Spezifikation des System herzuleiten.

Parallelkomposition

Die folgende Definition führt eine entsprechende Regel für die Parallelkomposition nach Definition 2.3.2 ein.

Definition 2.4.6 (Failure-Par-Regel) Seien S_1 und S_2 Failurespezifikationen der Art $S_1 : A_1^* \times \mathbb{P}(A_1) \to \mathbb{B}$ und $S_2 : A_2^* \times \mathbb{P}(A_2) \to \mathbb{B}$ sowie $A = A_1 \cup A_2$. Dann gilt

$$\frac{P_1 \text{ sat } S_1 \quad P_2 \text{ sat } S_2}{P_1 \parallel P_2 \text{ sat } \exists R_1 \subseteq A_1, R_2 \subseteq A_2 . S_1(A_1 ©t, R_1) \wedge S_2(A_2 ©t, R_2) \wedge R = R_1 \cup R_2}$$

∘

Abstraktion

Die folgende Definition führt eine entsprechende Regel für die Abstraktion nach Definition 2.3.3 ein.[16]

Definition 2.4.7 (Failure-Abs-Regel) Sei S eine Failurespezifikation der Art $S : A^* \times \mathbb{P}(A) \to \mathbb{B}$. Dann gilt

$$\frac{P \text{ sat } S}{P \backslash A' \text{ sat } \exists t' \in A^\omega . S(t', R \cup A') \wedge (A \backslash A') ©t' = t}$$

∘

[16] Die entsprechende Regel in [Hoa85] ist auf nichtdivergente Prozesse eingeschränkt. Darauf wird hier der Einfachheit halber verzichtet, da diese Regel in Abschnitt 3.1 diese Regel für die unendliche Failuresemantik eingeführt wird.

Korrektheit

Mit den Regeln 2.4.3, 2.4.6 und 2.4.7 lassen sich nun – entsprechend wie im Spur-sat-Kalkül – Herleitungen im Failure-sat-Kalkül definieren. Auch für diese Herleitungen im Failure-sat-Kalkül läßt ganz entsprechend zu Satz 2.4.1 die Korrektheit nachweisen. Da die Formulierung des Satzes identisch zu Satz 2.4.1 ist und der Beweis entsprechend verläuft, wird auf deren Einführung verzichtet.

2.5 Vergleich der Modelle

In Abschnitt 1.2 wurde bereits die Behauptung aufgestellt, daß asynchron kommunizierende nachrichtenorientierte Systeme gegenüber den synchron kommunizierenden den Vorteil aufweisen, daß sich ihre Eigenschaften einfacher beschreiben und spezifizieren sowie verifizieren lassen. In diesem Abschnitt wird nun diese Behauptung anhand eines Beispiels genauer betrachtet.

Dazu wird ein einfacher reaktiver Prozeß, der Puffer, unter beiden Sichtweisen untersucht.

2.5.1 Informelle Beschreibung

In 1.2 wurde darauf hingewiesen, daß bereits auf der informellen Beschreibungsebene der Aufwand zur Beschreibung eines asynchron kommunizierenden Systems geringer ist als der eines synchron kommunizierenden. Dies wird zuerst an einem Beispiel veranschaulicht. Dazu wird die Beschreibung eines einfachen Systems - eines Puffers - unter den beiden oben genannten Modellierungsannahmen betrachtet. Dieser Puffer könnte beispielsweise eingesetzt werden, um zwei miteinander kommunizierende Systemkomponenten zu verbinden.

Wird ein Puffer auf der Ebene der asynchron kommunizierenden Systeme beschrieben, so wäre eine geeignete Beschreibung hinsichtlich der von der Pufferkomponente empfangenen oder versendeten Nachrichten:

> Nachrichten werden nur in der Reihenfolge weitergegeben, in der sie empfangen werden; alle empfangenen Nachrichten werden auch weitergeleitet.

Im Gegensatz zum asynchronen Puffer muß bei der Beschreibung des synchronen Puffers wesentlich implementierungsnäher vorgegangen werden. Ein Puffer, der jede Aktion verweigert, erfüllt durchaus die obige Beschreibung. Hier ist es nötig, auch das Akzeptieren von Eingaben zu beschreiben. Zudem reicht es nicht aus, lediglich den vollständigen Ablauf eines Puffers zu charakterisieren; statt dessen muß festgelegt werden, wie sich der Puffer nach jedem Teilschritt verhalten soll, also welche Aktionen nach jedem partiellen Ablauf verweigert werden können. Damit müssen beispielsweise Entscheidungen über die Pufferkapazität und damit über die Eingabebereitschaft getroffen werden. Im vorliegenden Fall wird ein Puffer mit einer Mindestkapazität von einem Element beschrieben:

Nachrichten werden nur in der Reihenfolge weitergeleitet, in der sie empfangen werden; sind alle Nachrichten weitergeleitet, so kann der Empfang einer weiteren Nachricht nicht verweigert werden; wurden mehr Nachrichten empfangen als weitergeleitet, so kann die Ausgabe einer Nachricht nicht verweigert werden.

2.5.2 Formale Beschreibung

Bisher wurde gezeigt, daß bereits auf der informellen Ebene der Beschreibungsaufwand für synchron kommunizierende Systeme höher ist als im Fall der asynchron kommunizierenden, da detailliertere Aussagen getroffen werden müssen. Auf der formalen Ebene läßt sich die höhere Komplexität der Beschreibung synchron kommunizierender Systeme gegenüber asynchron kommunizierenden zusätzlich auf die höheren Komplexität des synchronen Modells zurückführen. Dies zeigt sich beim Vergleich entsprechender Modellierungen mittels dieser Modelle (z.B. [DW92] und [DS92a]). Zur Illustration werden dazu im folgenden die informellen Beschreibungen aus Abschnitt 2.5.1 in eine formale Darstellung umgesetzt, indem jede einzelne informelle Anforderung in einen entsprechenden Teil der Spezifikation umgesetzt wird:

$$
\begin{aligned}
P(t) &\stackrel{\text{def}}{=} \forall s \sqsubseteq t.\, \mathsf{data}^*(O©s) \sqsubseteq \mathsf{data}^*(I©s) \\
C(t) &\stackrel{\text{def}}{=} \mathsf{data}^*(I©t) = \mathsf{data}^*(O©t) \\
T(t) &\stackrel{\text{def}}{=} P(t) \wedge C(t)
\end{aligned}
$$

Dabei beschreibt $P(t)$ die Tatsache, daß die Nachrichten nur in der Reihenfolge weitergegeben werden, in der sie empfangen wurden; $C(t)$ drückt aus, daß alle Nachrichten, die empfangen wurden, auch tatsächlich weitergeleitet werden.

Die Formalisierung der Anforderungen im synchronen Fall erfolgt entsprechend wie im asynchronen Fall:

$$
\begin{aligned}
S(t) &\stackrel{\text{def}}{=} \forall s \sqsubseteq t.\, \mathsf{data}^*(O©s) \sqsubseteq \mathsf{data}^*(I©s) \\
F_I(t,R) &\stackrel{\text{def}}{=} \mathsf{data}^*(I©t) = \mathsf{data}^*(O©t) \Rightarrow I \cap R = \emptyset \\
F_O(t,R) &\stackrel{\text{def}}{=} (\mathsf{data}^*(O©t) \sqsubset \mathsf{data}^*(I©t) \wedge \mathsf{data}^*((O©t) \circ o) \sqsubseteq \mathsf{data}^*(I©t)) \Rightarrow o \notin R \\
F(t,R) &\stackrel{\text{def}}{=} S(t) \wedge F_I(t,R) \wedge F_O(t,R)
\end{aligned}
$$

Hier wird durch $F_I(t,R)$ ausgedrückt, daß der leere Puffer stets bereit ist, neue Nachrichten entgegenzunehmen; die Tatsache, daß gespeicherte Nachrichten stets ausgegeben werden können, wird durch $F_O(t,R)$ beschrieben.

2.6 Systementwurf auf der Spurebene

Wie in der Übersicht angegeben, findet in dem in dieser Arbeit beschriebenen Ansatz die erste Phase des Systementwurfs auf der Spurebene statt. Für den hier beschriebenen Ansatz wird davon ausgegangen, daß das Ergebnis dieser Entwurfsphase die Erstellung einer Spezifikation des Gesamtsystems sowie seiner Komponenten ist. Hierzu werden zuerst die Anforderungen an das Gesamtsystem bestimmt und anschließend eine detailliertere Beschreibung des Systems durch Angabe seiner Unterkomponente entwickelt. Diese Beschreibung der Komponenten besteht dabei aus der Beschreibung ihrer syntaktischen Schnittstelle sowie der Beschreibung von entsprechenden Anforderungen an das Verhalten dieser Komponenten. Dabei kann dieser Prozeß für jede Komponente wiederholt werden, jede Komponente kann wieder in Unterkomponenten zerlegt werden.

Mehrere Aspekte dieser Vorgehensweise sind dabei in den folgenden Abschnitten von besonderer Bedeutung, nämlich

- die Beschreibung des Systems und seiner Komponenten mittels partieller und vollständiger Eigenschaften,
- die Beschreibung des *Parallelitätsgrads* des Systems, d.h. der Granularität der Verteilung von Aufgaben auf verschiedene Prozesse, sowie
- die Art der Strukturierung eines Systems, d.h. der Zerlegung in einzelne Komponenten.

Im folgenden werden diese Aspekte näher erläutert und ihre Auswirkungen auf die vorgestellte Vorgehensweise vorgestellt.

2.6.1 Partielle und vollständige Eigenschaften

In Definition 2.2.5 wurde die Aufspaltung von Spureigenschaften in partielle und vollständige Eigenschaften eingeführt. Dabei wurde diese Aufspaltung damit motiviert, daß mit den partiellen Eigenschaften bereits auf der Spurebene die Eigenschaften von Abläufen beschrieben werden, die auch für die Abläufe unter der synchronen Systemsicht gelten sollen. Dies ist möglich, da - wie Kapitel 4 zeigt - die partiellen Eigenschaften beim Wechsel von der asynchronen zur synchronen Systemsicht erhalten bleiben. Implementieren also die partiellen Eigenschaften der Komponenten eines Systems eine partielle Eigenschaft des Systems auf der Spurebene, so überträgt sich diese Aussage auch auf die Failureebene.

Ganz analog wie bei der vergleichbaren Aufspaltung von Spureigenschaften in Sicherheits- und Lebendigkeitseigenschaften ergibt die Komposition von partiellen Eigenschaften mit den in 2.2.3 eingeführten Kompositionsoperatoren ebenfalls partielle Eigenschaften. Damit bietet sich die folgende Vorgehensweise bei der Implementierung von Systemeigenschaften durch Komponenteneigenschaften an:

1. Die partiellen Eigenschaften der Komponenten $K_1, \ldots, K_n$ implementieren die partiellen Eigenschaften des Systems S.
2. Die partiellen und vollständigen Eigenschaften der Komponenten $K_1, \ldots, K_n$ implementieren die vollständigen Eigenschaften des Systems S.

Wird der sat-Kalkül zum Nachweis der Implementierung verwendet, dann bedeutet dies formal, daß für

- die partiellen und vollständigen Eigenschaften $P_1, \ldots, P_n$ und $C_1, \ldots, C_n$ der Komponenten mit Eingabealphabet $I_1, \ldots, I_n$ und Ausgabealphabet $O_1, \ldots, O_n$ sowie
- die partiellen und vollständigen Eigenschaften P und C des Systems mit Eingabealphabet I und Ausgabealphabet O

die Aussage

$$\frac{\begin{array}{l} K_1 \text{ sat } P_1 \\ \quad \vdots \\ K_n \text{ sat } P_n \end{array}}{S \text{ sat } P}$$

sowie die Aussage

$$\frac{\begin{array}{l} K_1 \text{ sat } P_1 \wedge C_n \\ \quad \vdots \\ K_n \text{ sat } P_n \wedge C_n \end{array}}{S \text{ sat } C}$$

gelten. In Kapitel 3 wird dann gezeigt, daß sich die erste Aussage direkt von der Spur- zur Failureebene übertragen laßt. Auch für die zweite Aussage läßt sich mit dem dort beschriebenen Schema zum Wechsel von der Spur- zur Failureebene eine vergleichbare Aussage für das synchrone Modell formulieren.

2.6.2 Parallelität auf der Spurebene

Eine Möglichkeit, Modelle verteilter Systeme zu klassifizieren, besteht darin, zu unterscheiden, wie die Parallelität von Aktionen in diesen Modellen dargestellt wird. Dabei wird unterschieden zwischen solchen Modellen, die nicht zwischen der parallelen Ausführung zweier Aktionen und deren beliebiger Sequentialisierung unterscheiden ("Interleaving"), und solchen, die diesen Unterschied ausdrücken können ("True Concurrency").[17] Die hier getroffene Entscheidung, das Spur- und Failuremodell, und damit Interleavingsemantiken zu verwenden, hat auch Einfluß auf die methodische Vorgehensweise. Dieser Einfluß wird im folgenden kurz erläutert.

[17] Siehe auch Anhang A.4.3 für weitere Erläuterungen und Literaturhinweise.

Explizite Parallelität

Die Unfähigkeit von Interleavingsemantiken, Parallelität in gewünschtem Maße auszudrücken, wird immer wieder hervorgehoben und an spezifischen Beschränkungen festgemacht (vgl. z.B. [BCHK94]). In der hier vorgestellten Vorgehensweise tritt eine weiteren Facette dieser Problematik in den Vordergrund, die speziell auf der verwendeten Implementierungsrelation beruht. Die Unfähigkeit der Spursemantik, den gewünschten Grad der Parallelisierung im Entwicklungsprozeß mittels der Charakterisierung von Spurmengen zu unterstützen, wird im nächsten Beispiel kurz beleuchtet.

Beispiel 2.6.1 (Verteilter Puffer) Im folgenden Beispiel werden für einen Puffer Spezifikationen mit unterschiedlichem Parallelisierungsgrad angeben. Es wird zuerst eine Beschreibung angegeben, bei der der Puffer als Zweikanalpuffer, d.h. als Parallelkomposition zweier Puffer mit unterschiedlichen Alphabeten aufgefaßt wird. Dabei werden entsprechende Alphabete und eine geeignete Hilfsfunktion data^* wie in Beispiel 2.2.4 verwendet. Dieser *verteilte* Puffer T_{12} wird als Komposition von entsprechenden Eigenschaften der beiden Teilpuffer beschrieben:

$$T_{12}(t) \stackrel{\text{def}}{=} P_1(t) \wedge C_1(t) \wedge P_2(t) \wedge T_2(t)$$

wobei

$$\begin{aligned}
P_1(t) &\stackrel{\text{def}}{=} \forall s \sqsubseteq t.\, \mathsf{data}^*(Out_1 © s) \sqsubseteq \mathsf{data}^*(In_1 © s) \\
P_2(t) &\stackrel{\text{def}}{=} \forall s \sqsubseteq t.\, \mathsf{data}^*(Out_2 © s) \sqsubseteq \mathsf{data}^*(In_2 © s) \\
C_1(t) &\stackrel{\text{def}}{=} \mathsf{data}^*(Out_1 © t) = \mathsf{data}^*(In_1 © t) \\
C_2(t) &\stackrel{\text{def}}{=} \mathsf{data}^*(Out_2 © t) = \mathsf{data}^*(In_2 © t)
\end{aligned}$$

Eine graphische Darstellung der beabsichtigten verteilten Implementierung findet sich in Abbildung 2.4.

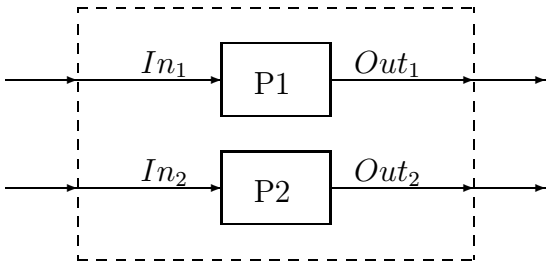

Abbildung 2.4: Parallele Implementierung von T_{12}

Im Unterschied zur parallelen Implementierung wird nun eine Spezifikation eines Puffers betrachtet, der nur aus einer sequentiellen Komponente bestehen und nicht aus der Parallelkomposition zweier Puffer mit unterschiedlichen Alphabeten, wie oben geschehen. Die entsprechende Spezifikation $T(t)$ wird folgendermaßen angegeben:

$$T(t) \stackrel{\text{def}}{=} P(t) \wedge C(t)$$

wobei

$$P(t) \stackrel{\text{def}}{=} \forall s \sqsubseteq t : \mathsf{data}^*((Out_1 \cup Out_2)©s) \sqsubseteq \mathsf{data}^*((In_1 \cup In_2)©s)$$
$$C(t) \stackrel{\text{def}}{=} \mathsf{data}^*((Out_1 \cup Out_2)©t) = \mathsf{data}^*((In_1 \cup In_2)©t)$$

Eine graphische Darstellung dieses Puffers findet sich in Abbildung 2.5. Bei der Verwen-

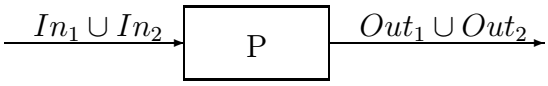

Abbildung 2.5: Serielle Implementierung von T_{12}

dung des sat-Kalküls und damit der Implikation als Implementierungsrelation zeigt sich, daß der durch $T(t)$ beschriebene Puffer eine Implementierung des durch T_{12} beschriebenen Puffers ist, da

$$T(t) \Rightarrow T_{12}(t)$$

gilt. Dies widerspricht jedoch der Intuition der Spezifikation, bei der beim Puffer T_{12} scheinbar explizit eine Verteilung der Nachrichten auf zwei Kanäle gefordert hat, diese jedoch in der durch $T(t)$ beschriebenen Implementierung nicht mehr berücksichtigt wird. Dementsprechend ist auch die Menge der beschriebenen Abläufe unterschiedlich. Während die parallele Implementierung den Ablauf

$$in_1.a \circ in_2.b \circ out_2.b \circ out_1.a$$

mit entsprechenden Nachrichten a und b auf den Kanälen erlaubt, so ist dies bei der sequentiellen Implementierung nicht möglich. Die sequentielle Implementierung erlaubt nicht die Ausgabe von $out_2.b$ vor $out_1.a$, da die Ausgabekanäle nicht parallel bedient werden. ◇

Dieser Mangel hat damit auch wesentlichen Einfluß auf den Wechsel vom Spurmodell zum Failuremodell, wie er in Kapitel 3 beschrieben wird. Der ursprünglich beabsichtigte Grad der Parallelisierung wird in den Anforderungen nicht dokumentiert und geht damit im Laufe des Entwicklungsprozesses verloren. Wie dieser Informationsverlust einfach zu vermeiden ist und was dies für die Begriffe *Komponente* und *Prozeß* bedeutet, wird im Abschluß dieses Abschnitts beschrieben.

Zu beachten ist, daß diese Form der Unterscheidung im Failuremodell getroffen werden kann. So wird im Falle von Beispiel 2.6.1 anhand der Refusalmengen unterschieden, ob - wie im parallelen Fall - auf Out_1 und Out_2 gleichzeitig Nachrichten verfügbar sind $((\exists o \in Out_1.o \notin R) \wedge (\exists o \in Out_2.o \notin R))$, oder - wie im sequentiellen Fall - nur jeweils eine Nachricht aus Out_1 und Out_2 $(\exists o \in Out_1 \cup Out_2.o \notin R)$. Andere, strengere Unterscheidungen sind jedoch auch im Failuremodell aufgrund seines *Interleaving*-Charakters nicht möglich.[18]

[18]Die Eigenschaft "Parallele Ausführung = beliebige Sequentialisierung" ist auch Teil der algebraischen Charakterisierung von CSP.

Syntaktische Festlegung der Parallelität mittels Prozessen und Komponenten

Das Prinzip der syntaktischen Festlegung besteht in der Festschreibung der Verteilung des Systems. Dies bedeutet, daß bereits zum Zeitpunkt der Systembeschreibung Aussagen über die innere Struktur des zu implementierenden Systems getroffen werden.

Bei der Definition des semantischen Modells in Abschnitt 2.2 und 2.3 wurde der Begriff des *Prozesses* als kleinste Beschreibungseinheit eingeführt. Im Gegensatz dazu wurde zu Beginn von Abschnitt 2.6 von *Komponenten* als Bausteine von Systemen gesprochen. Diese Unterscheidung ist dabei gewußt gewählt, da für den Systementwurf auf der Spurebene beide Begriffe verwendet werden sollen. Wurde oben davon gesprochen, daß bereits zum Zeitpunkt der Systembeschreibung Aussagen über den internen Aufbau des Systems getroffen werden sollen, so ist damit nicht der *architektonische* Aufbau des Systems aus Komponenten, sondern der *verhaltensmäßige* Aufbau aus Prozessen zu verstehen.

Soll also ein bestimmter Grad von Parallelität *explizit* für eine Komponente eines Systems gefordert werden, so muß diese Komponente durch *mehrere* Prozesse beschrieben werden. Dabei haben diese Prozesse alle Zugriff auf die Eingaben der Komponente, unterscheiden sich jedoch in den jeweiligen Ausgaben. In Beispiel 2.6.1 muß also die Komponente "Puffer" durch zwei Pufferprozesse realisiert werden.

Diese strukturelle Festlegung des Parallelitätgrads kann dann auch als Anforderung an die Spezifikation eines Systems eingesetzt werden. Damit kann festgelegt werden, daß die Realisierung einer Systemspezifikation den in der Spezifikation festgelegt Grad der Parallelität aufweist. Soll also ein System, beschrieben durch die Parallelkomposition der Prozesse $P_1 \parallel \ldots \parallel P_n$, mit diesem Grad der Parallelität realisiert werden, so ist für jeden der Prozesse eine eigene Realisierung anzugeben. Es sind also Prozesse $Q_{1,1}, \ldots, Q_{m,n_m}$ zu realisieren mit

$$\begin{array}{lcl} P_1 & \stackrel{\text{def}}{=} & (Q_{1,1} \parallel \ldots \parallel Q_{1,n_1}) \backslash H_1 \\ \vdots & \vdots & \vdots \\ P_m & \stackrel{\text{def}}{=} & (Q_{m,1} \parallel \ldots \parallel Q_{m,n_m}) \backslash H_m \end{array}$$

mit $P_1 \parallel \ldots \parallel P_m = (Q_{1,1} \parallel \ldots \parallel Q_{m,n_m}) \backslash (H_1 \cup \ldots \cup H_m)$. Dazu ist es nicht notwendig, daß die Realisierungen der Prozesse P_k nur aus unterschiedlichen Prozessen $Q_{i,j}$ bestehen, die Prozesse $Q_{i,j}$ können auch von mehreren Prozessen P_k genutzt werden. Da die Parallelkomposition der Spurbeschreibungen asynchroner Systeme - wie in Definition 2.2.3 beschrieben - keine Synchronisierung der Ausgabemengen erlaubt, wird so der gewünschte Grad der Parallelität realisiert.

Da, wie oben angesprochen, der Grad der Parallelität in der Failuredarstellung direkt durch die Beschreibung eines Prozesses ausgedrückt werden kann, spielt die Festlegung des Parallelitätgrads beim in Kapitel 3 beschriebenen Wechsel von der Spur- zur Failuresemantik eine wichtige Rolle. In Abschnitt 3.5 wird dazu gezeigt, wie die syntaktische Festlegung des Parallelitätgrads methodisch für die in dieser Arbeit vorgestellte Vorgehensweise eingesetzt werden kann.

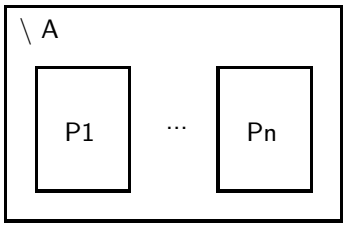

Abbildung 2.6: Struktur eines verteilten Systems

2.6.3 Beschreibung verteilter Systeme

Im Abschnitt 2.1.2 wurden zwei syntaktische Operatoren zur Komposition verteilter Systeme eingeführt, nämlich die Abstraktion und die Parallelkomposition. Es stellt sich daher die Frage, wie diese beiden Operatoren für die Entwicklung verteilter Systeme sinnvoll eingesetzt werden können.

Um diese Frage zu beantworten, wird zuerst nochmals auf die Beschreibung verteilter Systeme näher eingegangen. In [LL94] wird dazu darauf verwiesen, daß die Frage, ob ein System verteilt ist oder nicht, eine Frage des Blickwinkels ist:

> Although one usually speaks of a distributed system, it is more accurate to speak of a distributed *view* of a system.

Dies bedeutet, daß es jeweils vom Grad der Abstraktion abhängig ist, ob ein System bzw. eine Komponente als eine Einheit oder aufgebaut aus mehreren Unterkomponenten angesehen werden soll. Dies gilt im besonderen Maße für die Entwicklung von Anforderungen an ein verteiltes Systeme und seine Komponenten.

Daraus ergibt sich, wie in Abbildung 2.6 dargestellt, für die Entwicklung verteilter Systeme eine besondere syntaktische Gestalt für die Darstellung des zusammengesetzten Systems S. Diese Form des syntaktischen Aufbaus eines Systems und seiner Komponenten ist für die in Abschnitt 4.3 beschriebene Vorgehensweise zur Bestimmung minimaler Anforderungen wesentlich:

- Das Gesamtsystem S ergibt sich durch die Abstraktion des zusammengesetzten Systems S' hinsichtlich interner Aktionen, also

  $$S = (S' \backslash A')$$

- Das zusammengesetzte System schließlich entsteht durch die Parallelkomposition der einzelnen auf dieser Stufe als nicht verteilt betrachteten Komponenten, also

  $$S' = (P_1 \parallel \ldots \parallel P_n)$$

Dies bedeutet für den Spezifikationsentwicklungsprozeß, daß die Anforderungen an das Gesamtsystem durch die Parallelkomposition von Anforderungen an die einzelnen Komponenten der ersten Beschreibungsebene und die Abstraktion von internen Aktionen implemen-

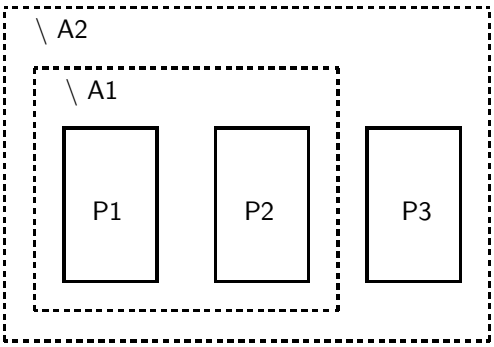

Abbildung 2.7: Vermischung von Parallelkomposition und Abstraktion

tiert werden.[19] Damit liegen sowohl für das Gesamtsystem als auch für die Komponenten explizite Anforderungen vor. Für eine weniger abstrakte Darstellung der Systemstruktur können diese Anforderungen, soweit nötig, wiederum durch entsprechende Detaillierungen realisiert werden. Im folgenden wird daher stets davon ausgegangen, daß das verteilte System durch die sukzessive Anwendung dieses Prinzips beschrieben wird und damit jeweils für die Komponenten bzw. Subkomponenten explizite Beschreibungen vorliegen. Insbesondere sollen hier keine Systeme der Gestalt

$$((P_1 \parallel P_2)\backslash A_1) \parallel P_3)\backslash A_2 \tag{2.8}$$

betrachtet werden, also Systeme, bei denen die Abstraktion von internen Aktionen auf der betrachteten Beschreibungsebene nicht ganz außen, sondern vermischt mit der Parallelkomposition auftritt, wie in Abbildung 2.7 dargestellt. Hier ist, entsprechend der syntaktischen Struktur, das System aus zwei Komponenten aufgebaut, wobei die erste Komponente selbst durch zwei Subkomponenten realisiert ist. Dabei liegen für die erste Komponente keine expliziten Anforderungen vor, sondern nur implizit durch die Anforderungen an die beiden Subkomponenten.

Hierbei ist anzumerken, daß diese Art der Systemstrukturierung keine echte Einschränkung an den Systemaufbau darstellt. Durch entsprechende Umbenennung der internen Aktionen von $(P_1 \parallel P_2)\backslash A_1$, also der Aktionen aus A_1 in Aktionen A_1', kann eine Umstrukturierung des Systems erreicht werden. Dabei sind die Aktionen so umzubenennen, daß $A_1' \cap A_2 = \emptyset$ gilt. Zur Abstraktion der entsprechenden internen Aktionen müssen dementsprechend auch P_1 sowie P_2 zu einem geeigneten P_1' bzw. P_2' angepaßt werden.[20] Dazu werden die Aktionen aus A_1 in P_1 durch entsprechende Aktionen aus A_1' ersetzt. Nach Satz 2.3.2 kann nun das System in 2.8 als

$$(P_1' \parallel P_2' \parallel P_3)\backslash(A_1' \cup A_2)$$

[19]Diese Parallelkomposition mit anschließender Abstraktion entspricht auch der Komposition von Prozessen, wie sie im wesentlichen in der CSP-Version von [Hoa78] verwendet wird.

[20]Die genaue formale Behandlung der Umbenennung von Prozessen wird in [Hoa85] und in Abschnitt A beschrieben.

dargestellt werden und genügt damit wieder dem oben erläuterten Strukturierungsprinzip. Daß darüberhinaus die Komponenten eines Systems als *flach* komponiert aufgefaßt werden[21], hier angedeutet durch die Verwendung der Parallelkomposition ohne Klammerung, ist schließlich auf die in Satz 2.3.1 angeführte Assoziativität der Parallelkomposition zurückzuführen.

2.7 Zusammenfassung

In diesem Kapitel wurden die Grundlagen der semantischen Modelle eingeführt, die für die in dieser Arbeit vorgestellte Vorgehensweise verwendet werden, und hinsichtlich ihrer Darstellungkomplexität verglichen. Aufbauend auf diesen Modellen wurden für die weitere Vorgehensweise dieser Arbeit wichtige Ergebnisse erarbeitet: Da die Beschreibung von System und Komponenten auf der asynchronen Ebene den Ausgangspunkt dieser Vorgehensweise darstellt, wurde gezeigt, wie sich für die spätere Entwicklung wichtige Eigenschaften bereits auf der Spurebene festlegen lassen. Neben der methodischen Behandlung partieller und totaler Eigenschaften wurden Beschreibungsschemata für zwei Klassen sequentieller Prozesse, monotone und unterbrechbare, eingeführt. Weiterhin wurde gezeigt, wie auf der Spurebene der gewünschte Grad der Parallelität unter Verwendung der Begriffe *Komponente* und *Prozeß* beschrieben werden kann. Zusammen mit einem Schema zur Strukturierung von Systemen sind so die Voraussetzungen für die weitere Vorgehensweise geschaffen.

[21] *Flach* bedeutet in diesem Zusammenhang, daß die Parallelkomposition selbst keine Strukturierung des Systems einführt, sondern lediglich die Abstraktion.

Kapitel 3

Modellwechsel

The major achievement of modern science is to demonstrate the links between phenomena at different levels of abstraction and generality, from quarks, particles, atoms and molecules right through to stars, galaxies, and (more conjecturally) the entire universe. On a less grand scale, the computer scientist has to establish such links in every implementation of higher level concepts in terms of lower (...). Such links are formalized as equations or more general predicates, describing the relationships between observations made at different levels of abstraction (...). Their clarification assists in understanding the structure of an entire scientific discipline.

C.A.R. Hoare, [Hoa94]

Wie eingangs in Abschnitt 1.3 erläutert, ist es das Ziel der Arbeit, eine mehrschrittige methodische Vorgehensweise für die Entwicklung nachrichtenorientierter synchron kommunizierender Systeme anzugeben:

- Im ersten Schritt werden System und Komponenten mit Hilfe des Spurmodells - wie in Abschnitt 2.2 beschrieben - spezifiziert. Dazu wird - wie im Spurmodell nötig - die Asynchronität der Kommunikation, also die unbeschränkte Bereitschaft zur Eingabe, sowohl für die Komponenten als auch das System angenommen. Diese Annahme wird in den weiteren Schritten wieder aufgehoben.
- Im zweiten Schritt wird auf ein detaillierteres Modell, das unendliche Failuremodell, gewechselt, das die Beschreibung von Systemen mit synchroner Kommunikation erlaubt. Für diesen Wechsel werden die bisher entwickelten Spurspezifikationen des Systems und der Komponenten in Failurespezifikationen umgesetzt. Dabei werden

Failurespezifikationen verwendet, die ebenfalls asynchrone Systeme beschreiben, also unbeschränkte Eingabebereitschaft garantieren.

Da das Failuremodell im Gegensatz zum Spurmodell die Beschreibung synchroner Kommunikation erlaubt, kann nun auch die eingeschränkte und tatsächlich gewünschte Eingabebereitschaft des Systems spezifiziert werden.

- Erst im dritten Schritt wird die Failurebeschreibung der Komponenten so abgeändert, daß die Forderung nach unbeschränkter Eingabebereitschaft abgeschwächt wird. Dabei werden die abgeschwächten Anforderungen an die Eingabebereitschaft des Systems berücksichtigt. Die abgeschwächten Failurebeschreibungen der Komponenten stellen schließlich die Anforderungen an eine effiziente Realisierung und damit den Schlußpunkt der Entwicklung dar.

Ziel von Kapitel 3 ist es, die Durchführung des zweiten Schritts zu ermöglichen. Dazu werden drei Schwerpunkte behandelt:

Einführung des unendlichen Failuremodells: In Abschnitt 3.1 wird das unendliche Failuremodell als Erweiterung des Failuremodells aus Abschnitt 2.3 eingeführt. Damit können sowohl die Eigenschaften von asynchron als auch synchron kommunizierenden Systemen verträglich wiederzugeben werden und die Voraussetzung für die Transformation der Spurspezifikationen geschaffen.

Transformation der Spurspezifikationen: In Abschnitt 3.2 wird gezeigt, daß sich die partiellen Eigenschaften direkt von der Spur- zur Failurebeschreibung übertragen lassen, während die vollständigen Spureigenschaften für die Failurebeschreibung angepaßt werden müssen. Dadurch werden die zuvor im asynchron orientierten Spurmodell beschriebenen Eigenschaften von System und Komponenten entsprechend im Failuremodell beschrieben, also, kurz gesagt, asynchron kommunizierende Komponenten und System in einem synchron orientierten Modell dargestellt. Dieses allgemeine Schema wird schließlich in Abschnitt 3.4 für Fall der in Abschnitt 2.2.5 beschriebenen sequentiellen Probleme durch ein speziell zugeschnittenes Schema ergänzt.

Abschwächung der Systemanforderungen: In Abschnitt 3.7 wird gezeigt, wie die wegen des Spurmodells geforderte unbeschränkte Eingabebereitschaft des Systems abgeschwächt werden kann. Ebenso wird eine entsprechende Abschwächung der Ausgabebereitschaft vorgestellt.

Weiterhin wird in 3.3 gezeigt, wie synchron und asynchron kommunizierende Komponenten kombiniert werden können. Schließlich werden in 3.6 und 3.8 die methodischen Vorzüge dieses Ansatzes gegenüber anderen Möglichkeiten der Entwicklung synchroner nachrichtenorientierter Systeme sowie weitere methodische Aspekte diskutiert. Die einzelnen Abschnitte werden dabei jeweils an einem kurzen Beispiel erläutert.

3.1 Unendliche Failuresemantik

Überlegungen, die Failuresemantik auch zur Modellierung asynchron kommunizierender

Systeme zu verwenden, sind bereits zuvor - beispielsweise in [JJH90] - angestellt worden. Die in [Hoa85] eingeführte klassische Failuresemantik ist für die in dieser Arbeit betrachtete Vorgehensweise nicht geeignet. Auch die dort beschriebene Erweiterung der Failuresemantik zur Failure/Divergence-Semantik eignet sich nicht. In diesem Abschnitt wird daher eine geeignete Modifikation der ursprünglichen Failuresemantik eingeführt, nämlich die *unendliche Failuresemantik*, die für die in dieser Arbeit verwendete Vorgehensweise geeignet ist. Dazu werden zuerst kurz die Schwächen der klassischen Failuresemantik aufgezeigt, das alternative Modell der endlichen Failuresemantik kurz skizziert und für die beiden Kompositionsoperatoren Parallelkomposition und Abstraktion die entsprechende Interpretation angegeben. Darüberhinaus werden die entsprechenden Regeln für den sat-Kalkül an dieses Modell angepaßt. Eine kurze Diskussion der Unterschiede zwischen klassischer und unendlicher Failuresemantik sowie eine Erläuterung der intuitiven Bedeutung des Modells beenden den Abschnitt. Da die unendliche Failuresemantik einen eigenständigen Komplex der Arbeit darstellt, findet sich eine ausführlichere Behandlung der unendlichen Failuresemantik in Anhang A.

3.1.1 Unverträglichkeit der Failuresemantik

Im folgenden werden nur die Schwächen der endlichen Failuresemantik angesprochen, soweit sie die technischen Aspekte der hier vorgestellten Vorgehensweise betreffen. Tatsächlich weist die unendliche Failuresemantik gegenüber der endlichen Failuresemantik noch weitere Eigenschaften neben der Behebung dieser Schwächen auf, die sie auch unabhängig von dieser Arbeit interessant erscheinen lassen. Diese Aspekte werden daher ausführlicher in Anhang A behandelt. Diese Schwächen wirken sich nicht nur auf den Wechsel der Modelle von der Spur- zur Failuresemantik aus, sondern auch auf die Bestimmung der Residuen, wie sie im Kapitel 4 beschrieben wird. Die Problematik bei der Residuenbestimmung wird in Abschnitt 4.2.3 behandelt.

Die Kernproblematik der Unzulänglichkeit der Failuresemantik für den hier vorgestellten Ansatz liegt in der Behandlung *divergenter* Systeme. Darunter werden solche Systeme verstanden, die eine unbeschränkte Anzahl von internen Aktionen ausführen können, ohne eine von außen beobachtbare Aktion durchzuführen. Während die Spursemantik divergente Systeme genauso behandelt wie nichtdivergente, führen diese Systeme in der Darstellung des Failuremodells zu einer Anomalie:[1] Divergenten Systemen wird durch die Failuresemantik kein Modell zugeordnet, die Semantik ist also hier nur partiell definiert. Auch die üblicherweise verwendete Failure/Divergence-Semantik, die diese Anomalie vermeidet, ist hier nicht einsetzbar. Sie ordnet solchen Systemen nämlich nicht das von außen beobachtbare Verhalten zu. Statt dessen ist bei einem solchen System ab dem Zeitpunkt der Divergenz jedes Verhalten möglich, jede Aktion kann ausgeführt oder abgelehnt werden.[2]

Ziel des ersten Schritts der in dieser Arbeit beschriebenen Vorgehensweise ist es, vom einfacheren Spurmodell ins komplexere Failuremodell zu wechseln, *ohne* dabei das jeweils

[1] Siehe Beispiel A.2.1 in Anhang A.
[2] Siehe Beispiel A.2.2 in Anhang A.

beschriebene Verhalten der asynchron kommunizierenden Systeme und Komponenten zu ändern. Genau dies geschieht aber, wenn dieser Modellwechsel an einem divergenten System vorgenommen wird. Trotzdem kann die Entwicklung divergenter Systeme durchaus von Interesse sein, da diese Divergenz durch die Möglichkeit der unbeschränkten Eingabebereitschaft in asynchron kommunizierenden Systemen entstanden sein kann; eine Beschränkung der nötigen Eingabebereitschaft, wie sie Rahmen der Vorgehensweise vorgenommen wird, kann dann zu einem nichtdivergenten System führen. Beispiel 4.2.1 beschreibt ein solches System.

Daher wird ein semantisches Modell benötigt, das in der Lage ist, nichtdivergente und divergente Systeme einheitlich zu behandeln. Eine Möglichkeit hierfür stellt die im folgenden beschriebene *unendliche Failuresemantik* dar. Dieses Modell wurde gewählt, da es eine sehr einfache Erweiterung des in Kapitel 2 vorgestellten klassischen Failuremodells darstellt und dank seiner Einfachheit einen einfachen Modellwechsel von der Spur- zur Failureebene erlaubt.

3.1.2 Modell

In ihrer ursprünglichen Form beschreibt die Failuresemantik, welche Aktionen ein Prozeß nach einem endlichen Ablauf verweigern kann. Dementsprechend bietet es sich an, als Modell Paare von *endlichen* Abläufen und Mengen von Aktionen zu verwenden. Damit läßt sich das semantische Modell $\mathcal{F}_A$ für Prozesse mit dem Alphabet A definieren als

$$\mathcal{F}_A \subseteq \mathbb{P}(A^* \times \mathbb{P}(A))$$

Für die in dieser Arbeit vorgestellte Vorgehensweise reicht jedoch die Betrachtung ausschließlich endlicher Abläufe aus den in 3.1.1 bzw. Anhang A beschriebenen Gründen nicht aus. Es ist daher notwendig, zusätzlich auch *unendliche* Abläufe zu betrachten. Damit wird als semantisches Modell $\mathcal{I}_A$ für Prozesse mit dem Alphabet A nun

$$\mathcal{I}_A \subseteq \mathbb{P}(A^\omega \times \mathbb{P}(A))$$

verwendet. Ebenso wie in der in 2.3.1 eingeführten Semantik erfüllen die Prozesse der unendlichen Failuremodells noch weitere Eigenschaften. Da diese für die weitere in dieser Arbeit beschriebene Vorgehensweise nicht unmittelbar von Bedeutung sind, wird der interessierte Leser auf den Anhang A verwiesen.

3.1.3 Intuitive Bedeutung

Wie im Abschnitt 3.1.1 besprochen, wurde die unendliche Failuresemantik aus technischen Gründen eingeführt. Zur Beschreibung der einzelnen Komponenten des Systems ist ein tieferes Verständnis der Bedeutung dieser Semantik eigentlich nicht nötig, da die

Prädikate zur Beschreibung dieser Komponenten durch eine schematische Transformation der ursprünglichen Spurkomponenten erhalten werden können. Trotzdem erscheint es sinnvoll, dem Modell auch eine intuitive Bedeutung beizumessen, um dadurch die Akzeptanz des Modells zu erhöhen und auch den Entwicklungsprozeß mit verbessertem Verständnis zu untermauern.

Im Falle der in Abschnitt 2.3 vorgestellten Failuresemantik ist die intuitive Bedeutung eines Failurepaares (t, R) sehr einsichtig. Hier beschreibt R die Menge aller Aktionen, die *nach* der Ausführung der Aktionen aus t vom betrachteten Prozeß abgelehnt werden können. Es ist jedoch ungleich schwieriger, einem Failurepaar mit einer unendlichen Spurkomponente eine intuitive Bedeutung zuzuordnen, da die Vorstellung einer Menge von Aktionen, die *nach* einer *unendlichen* Sequenz von Aktionen abgelehnt werden kann, anscheinend keinen besonderen Sinn macht. Dazu wird die bisherige, für endliche Spuranteile geeignete Vorstellung auf endliche und unendlicher Spuranteile t anwendbare Interpretation ausgeweitet. In diesem Fall drückt das Paar (t, R) aus, daß alle Aktionen aus R nach dem Eintreten eines Teils der Aktionen aus t *nicht ununterbrochen neben* den restlichen Aktionen aus t angeboten wurden. Befindet sich ein Failurepaar (t, R) nicht im Modell eines Prozesses, so bedeutet dies, daß nach einem Teil des Ablaufs t ein Teil (oder auch alle) Aktionen aus dieser Menge R parallel neben den jeweiligen Aktionen aus dem restlichen Ablauf angeboten wurden. In Anhang A.4 wird diese Intuition anhand zweier Beispiele deutlich gemacht.

3.1.4 Anpassung

Die Änderung des semantischen Modells zieht natürlich auch eine Änderung der auf diesem Modell definierten Operatoren bzw. der auf diesem Modell basierenden Kalkülregeln nach sich. Die dazu nötigen Anpassungen der Definitionen von Kompositionsoperatoren und sat-Kalkülregeln sind dabei jedoch selbsterklärend und werden hier lediglich aus Gründen der Vollständigkeit angegeben.

Kompositionsoperatoren

Die Definition der Parallelkomposition stellt eine triviale Anpassung der ursprünglichen Definition 2.3.2 an die unendliche Failuresemantik dar.

Definition 3.1.1 (Parallelkomposition) Seien $P_1 = (A_1, F_1)$ und $P_2 = (A_2, F_2)$ mit $F_1 \in \mathcal{I}_{A_1}$ bzw. $F_2 \in \mathcal{I}_{A_2}$ zwei Prozesse. Dann wird die Parallelkomposition von P_1 und P_2 definiert als

$$(P_1 \parallel P_2) \stackrel{\text{def}}{=} (A_1 \cup A_2, F)$$

wobei

$$F = \{(t, R_1 \cup R_2) \mid \ ((I_1 \cup O_1)\copyright t, R_1) \in F_1 \wedge ((I_2 \cup O_2)\copyright t, R_2) \in F_2\}$$

$\circ$

Entsprechend wird auch die Definition der Abstraktion 2.3.3 an die Abstraktion im unendlichen Failuremodell angepaßt.

Definition 3.1.2 (Abstraktion) Sei $P = (A, F)$ mit $F \in \mathcal{I}_A$ ein Prozeß. Dann wird die Abstraktion von P hinsichtlich von Aktionen aus A' definiert als

$$(P \backslash A') \stackrel{\text{def}}{=} (A \backslash A', F')$$

wobei

$$F' = \{((A \backslash A') \copyright t, R \backslash A') \mid (t, R \cup A') \in F\}$$

∘

sat**-Kalkül**

Wie bei der Anpassung der Definitionen von Parallelkomposition und der Abstraktion werden auf ganz ähnliche Weise auch die sat-Kalkülregeln an die unendliche Failuresemantik angepaßt. Die Implikationsregel lautet dabei

Definition 3.1.3 (Implikationsregel) Sei $S : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$ eine Failurespezifikation. Dann gilt

$$\frac{\begin{array}{l} P \text{ sat } S \\ S \Rightarrow S' \end{array}}{P \text{ sat } S'}$$

∘

Dementsprechend ergibt sich die ursprüngliche Regel 2.4.6 für die Parallelkomposition zu

Definition 3.1.4 (Par-Regel) Seien S_1 und S_2 Failurespezifikationen der Art $S_1 : A_1^\omega \times \mathbb{P}(A_1) \to \mathbb{B}$ und $S_2 : A_2^\omega \times \mathbb{P}(A_2) \to \mathbb{B}$ sowie $A = A_1 \cup A_2$. Dann gilt

$$\frac{\begin{array}{l} P_1 \text{ sat } S_1 \\ P_2 \text{ sat } S_2 \end{array}}{P_1 \parallel P_2 \text{ sat } \exists R_1 \subseteq A_1, A_2 \subseteq A_2 . S_1(A_1 \copyright t, R_1) \wedge S_2(A_2 \copyright t, R_2) \wedge R = R_1 \cup R_2}$$

∘

Die Regel 2.4.7 wird dementsprechend angepaßt zu

Definition 3.1.5 (Abs-Regel) Sei S eine Failurespezifikation der Art $S : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$. Dann gilt

$$\frac{P \text{ sat } S}{P \backslash A' \text{ sat } \exists t' \in A^\omega . S(t', R \cup A') \wedge (A \backslash A') \copyright t' = t}$$

∘

3.2 Transformation

Wie zu Beginn des Kapitel erläutert, wurden im ersten Schritt der Entwicklung im Spurmodell Spezifikationen für ein System und seine Komponenten entwickelt, sowie die Implementierung des Systems durch die Komponenten gezeigt. Da jedoch die Behandlung synchroner Kommunikation im Spurmodell nicht möglich ist, muß die Entwicklung in dem feineren Modell der Failuresemantik fortgesetzt werden. Dazu müssen die für System und Komponenten vorliegenden Spurbeschreibungen in Failurebeschreibungen umgesetzt werden. Damit diese Umsetzung für den in dieser Arbeit beschriebenen Ansatz geeignet ist, müssen folgende Anforderungen erhoben werden:

- Die Failureeigenschaft läßt sich durch ein einfaches Schema aus der Spureigenschaft entwickeln.
- Erfüllen die Komponenten eines Systems die Failureeigenschaften, die aus den Spureigenschaften der Komponenten entwickelt wurden, so erfüllt auch das System die Failureeigenschaften, die aus der Spureigenschaft des Systems entwickelt wurde.

Diese Anforderungen sind für die in dieser Arbeit beschriebene Vorgehensweise wesentlich, da die zweiphasige Entwicklung zur Vereinfachung des Entwurfs synchroner Systeme dienen soll. Kann der Übergang von der Spurbeschreibung zur Failurebeschreibung nicht einfach schematisch erfolgen, so ist besteht kein methodischer Vorteil gegenüber einer Entwicklung, die direkt auf der synchronen Ebene ansetzt. Weiterhin sollen die mit Hilfe der asynchronen Systemsicht vorgenommenen Entwicklungsschritte auch unter der synchronen Systemsicht zur Verfügung stehen. Ist dies nicht der Fall, so geht die dort vorgenommene Entwicklungsarbeit verloren.

Vor der formalen Definition dieser Umsetzung wird nochmals auf die Abschnitte 2.2 bzw. 2.3 zurückgegriffen und erläutert, welche Vorstellungen hinter den beiden hier verwendeten semantischen Modellen steckt:

- Im Spurmodell werden ausgabevollständige Abläufe eines Prozesses dargestellt; nach jedem Ablauf wird einen Ruhezustand des Prozesses erreicht, bei dem alle durch die beobachteten Eingaben ausgelösten Ausgaben bereits aufgetreten sind.
- Im Failuremodell werden partielle Abläufe einschließlich der Mengen der nach diesen Abläufen vom Prozeß ablehnbaren Aktionen dargestellt; jeder Ablauf beschreibt einen Zustand des Prozesses, bei dem eventuell - in Abhängigkeit der Refusalmenge - noch weitere Ausgaben (oder auch Eingaben) möglich sind.

In Abschnitt 2.2.5 wurde bereits angesprochen, daß Spureigenschaften eines Prozesses oft als Kombination von den Eigenschaften der partiellen und vollständigen Abläufe dargestellt werden. Auf der anderen Seite charakterisieren Spuranteile einer Failurebeschreibung die partiellen Eigenschaften. Damit liegt es nahe, die partiellen und vollständigen Spureigenschaften getrennt voneinander umzusetzen. In Abschnitt 3.2.1 wird gezeigt, daß sich die partiellen Eigenschaften ohne zusätzliche Umsetzung direkt von der Spur- auf die Failurebeschreibung übertragen lassen. Weiterhin wird gezeigt, daß die partiellen Eigenschaft

auch die zweite der obigen Anforderungen erfüllen. Damit ergibt sich für diese Vorgehensweise die Motivation, die auf der synchronen Ebene gewünschten partiellen Eigenschaften bereits auf der asynchronen Ebene als partielle Spureigenschaften zu verwenden. In Abschnitt 3.2.2 wird anschließend auch eine Umsetzung für die verbleibenden vollständigen Eigenschaften angegeben. Auch diese Umsetzung erfüllt die beiden obigen Anforderungen.

3.2.1 Verträglichkeit der partiellen Eigenschaften

Wie in Kapitel 2 erläutert, unterscheidet sich eine Failureeigenschaft von einer Spureigenschaft nur durch die zusätzliche Refusalkomponente. Somit stellt eine Spureigenschaft auch eine Failureeigenschaft dar. Da aber für das endliche Failuremodell und das unendliche Failuremodell die in Definition 2.3.1 bzw. A.3.1 beschriebenen Nebenbedingungen für Prozesse erfüllt sein müssen, stellen nicht alle Spureigenschaften sinnvolle Failureeigenschaften dar. Nur wenn die Spureigenschaft präfixabgeschlossen ist, kann sie als Failureeigenschaft zur Charakterisierung eines Prozesses verwendet werden.

Da die in Definition 2.2.5 eingeführten partiellen Eigenschaften präfixabgeschlossen sind, können sie - wie in Abschnitt 2.2.5 erwähnt - direkt in entsprechende partielle Eigenschaften des Failuremodells umgesetzt werden. Darüberhinaus wurde in Abschnitt 2.6 bereits erwähnt, daß sich die Implementierungsbeziehung für partiellen Eigenschaften von der Spur- auf die Failureebene übertragen läßt: garantieren die partiellen Spureigenschaften der Komponenten die partiellen Spureigenschaften des Systems, so gilt dies auch für die partiellen Failureeigenschaften. Diese Behauptung zeigen die beiden folgenden Sätze für die beiden in Kapitel 2 eingeführten Kompositionsoperatoren.

Satz 3.2.1 (Partielle Verträglichkeit 1) Zur Spezifikation zweier asynchroner Prozesse P_1 und P_2 sowie zweier synchroner Prozesse Q_1 und Q_2 mit

- Eingabealphabeten I_1 und I_2 sowie Ausgabealphabeten O_1 und O_2 für P_1 bzw. P_2,
- Alphabet $I_1 \cup O_1$ und $I_2 \cup O_2$ für Q_1 bzw. Q_2

seien die Spureigenschaften $T_1 : (I_1 \cup O_1)^\omega \to \mathbb{B}$, $T_2 : (I_1 \cup O_2)^\omega \to \mathbb{B}$ und $T : (I_1 \cup I_2 \cup O_1 \cup O_2)^\omega \to \mathbb{B}$ gegeben, wobei

$$\frac{P_1 \text{ sat } T_1(t) \quad P_2 \text{ sat } T_2(t)}{P_1 \parallel P_2 \text{ sat } T(t)} \tag{3.1}$$

im Spur-sat-Kalkül herleitbar ist. Dann ist auch

$$\frac{Q_1 \text{ sat } T_1(t) \quad Q_2 \text{ sat } T_2(t)}{Q_1 \parallel Q_2 \text{ sat } T(t)} \tag{3.2}$$

im Failure-sat-Kalkül herleitbar. •

Beweis 3.2.1 (Satz 3.2.1) Ist Aussage 3.1 im Spur-sat-Kalkül herleitbar, so gilt entsprechend Satz 2.4.1 sowie Definition 2.2.3

$$(\forall t.P_1(t) \Rightarrow T_1(t)) \wedge (\forall t.P_2(t) \Rightarrow T_2(t)) \Rightarrow \\ (\forall t.P_1((I \cup O_1)©t) \wedge P_2((I_1 \cup O_2)©t) \Rightarrow T(t))$$

wobei $P_1 : (I_1 \cup O_1)^\omega \rightarrow \mathbb{B}$ und $P_2 : (I_1 \cup O_2)^\omega \rightarrow \mathbb{B}$ die Spurmengen beliebiger Prozesse P_1 und P_2 charakterisieren. Bei Wahl von T_1 für P_1 und T_2 für P_2 folgt somit

$$\forall t \in (I_1 \cup I_2 \cup O_1 \cup O_2)^\omega.\ T_1((I_1 \cup O_1)©t) \wedge T_2((I_2 \cup O_2)©t) \Rightarrow T(t)$$

Damit ist entsprechend Definition 3.1.4 und 3.1.3 auch Aussage 3.2 im Failure-sat-Kalkül herleitbar. □

Die entsprechende Aussage gilt auch für die Abstraktion.

Satz 3.2.2 (Partielle Verträglichkeit 2) Zur Spezifikation eines asynchronen Prozesses P und eines synchronen Prozesses Q mit

- Eingabealphabet I sowie Ausgabealphabet O für P,
- Alphabet $I \cup O$ für Q

und $H \subseteq O$ seien die Spureigenschaften $T : (I \cup O)^\omega \rightarrow \mathbb{B}$ und $T' : (I \cup O \backslash H)^\omega \rightarrow \mathbb{B}$ gegeben, wobei

$$\frac{P \text{ sat } T(t)}{T \backslash H \text{ sat } T'(t)} \tag{3.3}$$

im Spur-sat-Kalkül herleitbar ist. Dann ist auch

$$\frac{Q \text{ sat } T(t)}{Q \backslash H \text{ sat } T'(t)} \tag{3.4}$$

im Failure-sat-Kalkül herleitbar. •

Beweis 3.2.2 (Satz 3.2.2) Ist Aussage 3.3 im Spur-sat-Kalkül herleitbar, so gilt entsprechend Satz 2.4.1 sowie Definition 2.2.3

$$(\forall t.P(t) \Rightarrow T(t)) \Rightarrow (\forall t.(\exists t'.((I \cup O) \backslash H)©t' = t \wedge P(t)) \Rightarrow T'(t))$$

wobei $P : (I \cup O)^\omega \rightarrow \mathbb{B}$ die Spurmenge eines Prozesses P charakterisiert. Bei Wahl von T für P folgt somit

$$\forall t \in ((I \cup O) \backslash H)^\omega.(\exists t' \in (I \cup O)^\omega.((I \cup O) \backslash H)©t' = t \wedge T(t)) \Rightarrow T'(t)$$

Damit ist entsprechend Definition 3.1.5 und 3.1.3 auch Aussage 3.4 im Failure-sat-Kalkül herleitbar. □

Hierbei ist zu beachten, daß weder in Satz 3.2.1 noch in Satz 3.2.2 von der Eigenschaft Gebrauch gemacht wurde, daß die Spureigenschaften partielle Eigenschaften sind. Zwar gelten auch für nicht präfixabgeschlossene Spureigenschaften die Herleitbarkeitsaussagen von Satz 3.2.1 und 3.2.2, jedoch ist hier die Aussage trivial, da dann kein synchroner Prozeß existiert, der diese Eigenschaft erfüllt.

3.2.2 Implizierte Failureeigenschaften

In Abschnitt 3.2.1 wurde gezeigt, daß partielle Eigenschaften der Spurbeschreibung direkt in eine Failurebeschreibung umgesetzt werden können. Damit bleibt also die Aufgabe, auch für die vollständigen Eigenschaften eine passende Umsetzung anzugeben. Wie in Kapitel 2 erläutert, beschreibt das Spurmodell ausgabevollständige Abläufe; das Failuremodell dagegen beschreibt partielle Abläufe sowie die Mengen der nach diesem Ablauf ablehnbaren Aktionen. Durch die schematische Umsetzung muß also die Charakterisierung einer Menge ausgabevollständiger Abläufe in eine Charakterisierung einer Menge partieller Abläufe und Refusalmengen umgesetzt werden. Da die vollständigen Eigenschaften der Spurbeschreibung im Gegensatz zu den partiellen Ausgaben nicht für jeden Ablauf sondern nur die ausgabevollständigen Abläufe gelten sollen, läßt sich folgende Anforderung für eine Umsetzung der Spurbeschreibung aufstellen:

- Ist der Prozeß nach einem Ablauf nicht mehr ausgabebereit, können also alle Ausgaben abgelehnt werden, so ist dieser Ablauf ausgabevollständig und erfüllt damit auch die vollständige Eigenschaft der Spurbeschreibung.

Da darüberhinaus Spurbeschreibungen von einer asynchronen Systemsicht mit unbeschränkter Eingabebereitschaft ausgehen, läßt sich eine weitere Anforderung an die Umsetzung der Spurbeschreibung in eine Failurebeschreibung aufstellen:

- Ein asynchron kommunizierender Prozeß darf "nie" eine Eingabe ablehnen.[3]

Für diese beiden Anforderungen ist nun eine schematische Umsetzung einer vollständigen Spureigenschaft in eine Failureeigenschaft anzugeben. Die folgende Definition gibt eine formale Beschreibung dieser Umsetzung.

Definition 3.2.1 (Implizierte Failureeigenschaft) Sei $A = I \cup O$ das Alphabet eines asynchronen Prozesses mit Eingabealphabet I und Ausgabealphabet O, sowie $T : A^\omega \to \mathbb{B}$ eine Spureigenschaft. Dann heißt $F : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$ mit

$$F^T(t, R) \stackrel{\text{def}}{=} F_T^I(t, R) \wedge F_T^O(t, R)$$

wobei

$$F_T^I(t, R) \stackrel{\text{def}}{=} I \cap R = \emptyset \tag{3.5}$$

$$F_T^O(t, R) \stackrel{\text{def}}{=} O \subseteq R \Rightarrow T(t) \tag{3.6}$$

die *durch T implizierte Failureseigenschaft.* ∘

Offensichtlich erfüllt dieses Umsetzung einer vollständigen Spureigenschaft in eine durch sie implizierte Failureeigenschaft die zu Beginn von Abschnitt 3.2 aufgestellte erste Forderung: die Failureeigenschaft läßt sich durch ein einfaches Schema aus der Spureigenschaft

[3]Diese Anforderung, die auch in anderen Ansätzen (vgl. z.B. [LT89], [Jos92]) eine wesentliche Rolle spielt, wird im allgemeinen als *input-enabledness* bezeichnet. Dort wird die input-enabledness zu jedem *endlichen* Zeitpunkt eines Ablaufs gefordert; hier - siehe Definition 3.2.1 und 3.4.2 - werden statt dessen stärkere Anforderungen gestellt.

herleiten. Um für die in dieser Arbeit vorgestellte Vorgehensweise geeignet zu sein, muß nun aber auch noch die zweite Anforderung sichergestellt werden: aus der Umsetzung der Komponenteneigenschaften muß sich die Umsetzung der Systemeigenschaften ableiten lassen. Daher wird im folgenden die Verträglichkeit dieser Eigenschaft mit den Kompositionsoperatoren untersucht.

3.2.3 Verträglichkeit der Transformation

Wie zu Beginn dieses Kapitels erwähnt, stellen die aus den Spurbeschreibungen umgesetzten Failurebeschreibungen den Ausgangspunkt für den nächsten Entwicklungsschritt dar, nämlich die Abschwächung der Failurebeschreibungen und insbesondere der unbeschränkten Eingabebereitschaft. Dazu ist es nötig, daß die auf der Spurebene hergestellte Implementierungsbeziehung zwischen den Eigenschaften der Komponenten eines Systems und den Eigenschaften eines Systems auf die Failureebene übertragen werden kann.

Wie bereits im Fall der partiellen Eigenschaften wird nun auch die Verträglichkeit der Umsetzung mit den in Abschnitt 2.1.2 eingeführten Kompositionsoperatoren bezüglich der Implementierungsbeziehung gezeigt. Die Verträglichkeit der Umsetzung stellt sicher, daß sich der Nachweis der Implementierungsbeziehung von der Spurebene auf die Failureebene übertragen läßt. Ähnlich wie im Fall der partiellen Eigenschaften ist dazu zu zeigen, daß jeder Beweisschritt im Spur-sat-Kalkül durch entsprechende Beweisschritte im Failure-sat-Kalkül nachvollzogen werden kann.

Verträglichkeit hinsichtlich der Parallelkomposition

Zunächst wird die Kompositionalität der in 3.2.1 beschriebenen Umformung von Spurprädikaten zu Failureprädikaten für die Parallelkomposition nachgewiesen. Dazu werden die Anforderung T_1 und T_2 an zwei asynchrone Prozesse P_1 und P_2 betrachtet; im Spur-sat-Kalkül gilt damit

$$\frac{P_1 \text{ sat } T_1 \qquad P_2 \text{ sat } T_2}{P_1 \parallel P_2 \text{ sat } T_1(A_1 \copyright t) \wedge T_2(A_2 \copyright t)}$$

Werden nun zwei Failuremengen Q_1 und Q_2 betrachtet, deren Anforderungen durch Umformung der Anforderungen an die asynchronen Prozesse hervorgehen, so gilt: die Failuremenge $Q_1 \parallel Q_2$ erfüllt die Anforderungen, die durch die Umsetzung der Anforderung $T_1(A_1 \copyright t) \wedge T_2(A_2 \copyright t)$ entsteht. Diese Feststellung drückt der nachfolgende Satz aus.

Satz 3.2.3 (Verträglichkeit 1) Für die Spezifikation asynchroner Prozesse seien mit

- $T_1 : (I_1 \cup O_1)^\omega \to \mathbb{B}$
- $T_2 : (I_2 \cup O_2)^\omega \to \mathbb{B}$

- $T : (I_1 \cup I_2 \cup O_1 \cup O_2)^\omega \to \mathbb{B}$

Spureigenschaften asynchroner Prozesse mit den Ein- und Ausgabealphabeten I_1 und O_1, I_2 und O_2, bzw. $(I_1 \cup I_2)\backslash(O_1 \cup O_2)$ und $O_1 \cup O_2$ gegeben. Weiterhin sei

$$\frac{P_1 \text{ sat } T_1(t) \quad P_2 \text{ sat } T_2(t)}{P_1 \parallel P_2 \text{ sat } T(t)} \tag{3.7}$$

im Spur-sat-Kalkül herleitbar. Es seien

- $F_1 : (I_1 \cup O_1)^\omega \times \mathbb{P}(I_1 \cup O_1) \to \mathbb{B}$
- $F_2 : (I_2 \cup O_2)^\omega \times \mathbb{P}(I_1 \cup O_1) \to \mathbb{B}$
- $F : (I_1 \cup I_2 \cup O_1 \cup O_2)^\omega \times \mathbb{P}(I_1 \cup I_2 \cup O_1 \cup O_2) \to \mathbb{B}$

die implizierten Failureeigenschaften der entsprechenden Spurprädikate T_1, T_2 bzw. T. Dann ist auch die Aussage

$$\frac{Q_1 \text{ sat } F_1(t, R) \quad Q_2 \text{ sat } F_2(t, R)}{Q_1 \parallel Q_2 \text{ sat } F(t, R)} \tag{3.8}$$

im Failure-sat-Kalkül herleitbar. •

Beweis 3.2.3 (Satz 3.2.3) Für den Nachweis der mengentheoretischen Aussagen in den nachfolgenden Beweisen werden im wesentlichen Theoreme aus [KM68] verwendet. Weiterhin seien im folgenden

- $A_1 = I_1 \cup O_1$
- $A_2 = I_2 \cup O_2$
- $A = A_1 \cup A_2$

Um nun die Herleitbarkeit der Aussage 3.8 im Failure-sat-Kalkül zu zeigen, reicht es entsprechend den Definitionen 3.1.4 und 3.1.3 aus,

$$\begin{aligned}&\forall t \in A^\omega, R_1 \subseteq A_1, R_2 \subseteq A_2.\\ &\quad F_1(A_1 ©t, R_1) \wedge F_2(A_2 ©t, R_2) \Rightarrow F(t, R_1 \cup R_2)\end{aligned}$$

zu zeigen. Damit wird zunächst die Aussage

$$F_1(A_1 ©t, R_1) \wedge F_2(A_2 ©t, R_2) \wedge O \subseteq R_1 \cup R_2 \Rightarrow T_1(A_1 ©t) \wedge T_2(A_2 ©t) \tag{3.9}$$

gezeigt:

$$\begin{aligned}&F_1(A_1 ©t, R_1) \wedge F_2(A_2 ©t, R_2) \wedge O \subseteq R_1 \cup R_2\\ \Longleftrightarrow\ &[O = O_1 \cup O_2]\\ &F_1(A_1 ©t, R_1) \wedge F_2(A_2 ©t, R_2) \wedge O_1 \cup O_2 \subseteq R_1 \cup R_2\\ \Longleftrightarrow\ &[\text{Def. } F_1(t, R), F_2(t, R)]\end{aligned}$$

$$
\begin{array}{ll}
 & I_1 \cap R_1 = \emptyset \wedge (O_1 \subseteq R_1 \Rightarrow T_1(A_1 ©t)) \wedge \\
 & I_2 \cap R_2 = \emptyset \wedge (O_2 \subseteq R_2 \Rightarrow T_2(A_2 ©t)) \wedge \\
 & O_1 \cup O_2 \subseteq R_1 \cup R_2 \\
\Longrightarrow & [O_1 \cap O_2 = \emptyset, I_i \cap R_i = \emptyset, R_i \subseteq I_i \cup O_i, *] \\
 & O_1 \subseteq R_1 \Rightarrow T_1(A_1 ©t) \wedge \\
 & O_2 \subseteq R_2 \Rightarrow T_2(A_2 ©t) \wedge \\
 & O_1 \subseteq R_1 \wedge O_2 \subseteq R_2 \\
\Longrightarrow & [\text{Aussagenlogik}] \\
 & T_1(A_1 ©t) \wedge T_2(A_2 ©t)
\end{array}
$$

Für den Schluß $(*)$ wird exemplarisch die Disjunktheit von R_2 und O_1 gezeigt:

$$
\begin{array}{ll}
 & R_2 \cap O_1 \\
= & [R_2 \subseteq A_2 \Rightarrow R_2 \cap A_2 = R_2] \\
 & (A_2 \cap R_2) \cap O_1 \\
= & [\text{Def. } A_2] \\
 & ((I_2 \cup O_2) \cap R_2) \cap O_1 \\
= & [\text{Assoziativität von } \cap] \\
 & (I_2 \cup O_2) \cap (R_2 \cap O_1) \\
= & [\text{Distributivität von } \cup \text{ und } \cap] \\
 & (I_2 \cap (R_2 \cap O_1)) \cup (O_2 \cap (R_2 \cap O_1)) \\
= & [\text{Assoziativität von } \cap] \\
 & ((I_2 \cap R_2) \cap O_1) \cup (O_2 \cap (R_2 \cap O_1)) \\
= & [I_2 \cap R_2 = \emptyset] \\
 & (\emptyset \cap O_1) \cup (O_2 \cap (R_2 \cap O_1)) \\
= & [\emptyset \cap O_1 = \emptyset] \\
 & \emptyset \cup (O_2 \cap (R_2 \cap O_1)) \\
= & [\emptyset \cup M = M] \\
 & O_2 \cap (R_2 \cap O_1) \\
= & [\text{Kommutativität von } \cap] \\
 & O_2 \cap (O_1 \cap R_2) \\
= & [\text{Assoziativität von } \cap] \\
 & (O_2 \cap O_1) \cap R_2) \\
= & [\text{Kommutativität von } \cap] \\
 & (O_1 \cap O_2) \cap R_2) \\
= & [O_1 \cap O_2 = \emptyset] \\
 & \emptyset \cap R_2 \\
= & [\emptyset \cap R_2 = \emptyset] \\
 & \emptyset
\end{array}
$$

Der Nachweis der Disjunktheit von R_1 und O_2 wird entsprechend gezeigt.

Wegen der Herleitbarkeit von 3.7 im Spur-sat-Kalkül folgt – wie bereits in Beweis 3.2.1 von Satz 3.2.1 gezeigt –

$$\forall t \in A^{\omega}.T_1(A_1 ©t) \wedge T_2(A_2 ©t) \Rightarrow T(t)$$

Damit folgt aus 3.9 auch

$$F_1(A_1 \copyright t, R_1) \wedge F_2(A_2 \copyright t, R_2) \wedge O \subseteq R_1 \cup R_2 \Rightarrow T(t) \tag{3.10}$$

Es bleibt der Nachweis der Aussage

$$I \cap (R_1 \cup R_2) = \emptyset \tag{3.11}$$

Dies folgt direkt aus der Eigenschaft der Alphabete:

$$\begin{array}{ll} & I_1 \cap R_1 = \emptyset \wedge I_2 \cap R_2 = \emptyset \\ \Longrightarrow & [O_1 \cap O_2 = \emptyset, I_i \cap O_i = \emptyset, R_i \subseteq I_i \cup O_i] \\ & ((I_1 \cup I_2) \backslash (O_1 \cup O_2)) \cap (R_1 \cup R_2) = \emptyset \\ \Longleftrightarrow & [\text{Def. } I] \\ & (I \cap (R_1 \cup R_2) = \emptyset \end{array}$$

Aus den Aussagen 3.10 und 3.11 folgt direkt die Behauptung. □

Verträglichkeit hinsichtlich der Abstraktion

Ebenso wie die Aussage der Verträglichkeit für die Parallelkomposition gilt, kann auch die entsprechende Aussage für die Abstraktion gezeigt werden. Dazu werden die Anforderungen T an einen asynchronen Prozeß P betrachtet; diese Anforderungen reichen aus, um die Anforderungen an das asynchrone System $P \backslash H$ zu realisieren. Wird nun eine Failuremenge Q betrachtet, deren Anforderungen durch Umformung der Anforderungen an den asynchronen Prozeß hervorgehen, so gilt: die Failuremenge $Q \backslash H$ erfüllt die Anforderungen, die durch Umformung der Anforderungen an das asynchrone System $P \backslash H$ hervorgehen. Diese Feststellung drückt der nachfolgende Satz aus.

Satz 3.2.4 (Verträglichkeit 2) Für die Spezifikation asynchroner Prozesse seien mit

- $T : (I \cup O)^\omega \rightarrow \mathbb{B}$
- $T' : (I \cup O \backslash H)^\omega \rightarrow \mathbb{B}$

Spurprädikate asynchroner Prozesse mit Ein- und Ausgabealphabeten I und O bzw. I und $O \backslash H$ gegeben, wobei $H \subseteq O$. Weiterhin sei

$$\frac{P \text{ sat } T(t)}{P \backslash H \text{ sat } T'(t)} \tag{3.12}$$

im Spur-sat-Kalkül herleitbar. Es seien

- $F : (I \cup O)^\omega \times \mathbb{P}(I \cup O) \rightarrow \mathbb{B}$
- $F' : (I \cup O \backslash H)^\omega \times \mathbb{P}(I \cup O \backslash H) \rightarrow \mathbb{B}$

die implizierten Failureeigenschaften der entsprechenden Spurprädikate T bzw. T'. Dann ist auch die Aussage

$$\frac{Q \text{ sat } F(t, R)}{Q \backslash H \text{ sat } F'(t, R)} \tag{3.13}$$

im Failure-sat-Kalkül herleitbar. •

Beweis 3.2.4 (Satz 3.2.4) Im folgenden sei nun

- $A = I \cup O$
- $A' = A \backslash H$

Um die Herleitbarkeit der Aussage 3.13 zu zeigen, reicht es laut 3.1.5 und 3.1.3

$$\forall t \in A^{\omega}, R \subseteq A.F(t, R \cup H) \Rightarrow F'(A' \copyright t, R \backslash H) \tag{3.14}$$

zu zeigen. Wegen

$$\begin{array}{ll} & \exists t'.A' \copyright t' = t \wedge F(t', R \cup H) \\ \Longrightarrow & [\text{Aussage 3.14}] \\ & \exists t'.A' \copyright t' = t \wedge F'(A' \copyright t', R \backslash H) \\ \Longrightarrow & [\text{Prädikatenlogik}] \\ & \exists t'.A' \copyright t' = t \wedge F'(t, R \backslash H) \\ \Longrightarrow & [\text{Prädikatenlogik}] \\ & F'(t, R \backslash H) \end{array}$$

folgt damit die Herleitbarkeit von 3.13.

Wegen der Herleitbarkeit von 3.12 im sat-Kalkül gilt – wie in Beweis 3.2.2 von Satz 3.2.2 gezeigt –

$$\forall t \in A'^{\omega}.(\exists t' \in A^{\omega}.(A' \copyright t' = t \wedge T(t)) \Rightarrow T'(t) \tag{3.15}$$

Wegen

$$\begin{array}{ll} & T(t) \wedge t' = A' \copyright t \\ \Longrightarrow & [\text{Aussage 3.15}] \\ & T'(t') \wedge t' = A' \copyright t \end{array}$$

gilt auch

$$T(t) \Rightarrow T'(A' \copyright t) \tag{3.16}$$

Damit folgt

$$\begin{array}{ll} & F(t, R \cup H) \\ \Longleftrightarrow & [\text{Def. } F(t,R)] \\ & I \cap (R \cup H) = \emptyset \wedge (O \subseteq (R \cup H) \Rightarrow T(t)) \\ \Longrightarrow & [I \cap H = \emptyset] \\ & I \cap R = \emptyset \wedge (O \subseteq (R \cup H) \Rightarrow T(t)) \\ \Longrightarrow & [\text{Prädikatenlogik}] \\ & I \cap R = \emptyset \wedge ((O \backslash H) \subseteq (R \backslash H) \Rightarrow T(t)) \\ \Longrightarrow & [\text{Def. } O'] \\ & I \cap R = \emptyset \wedge (O' \subseteq (R \backslash H) \Rightarrow T(t)) \\ \Longrightarrow & [3.16] \\ & I \cap R = \emptyset \wedge (O' \subseteq (R \backslash H) \Rightarrow T'(A' \copyright t)) \\ \Longleftrightarrow & [\text{Def. } F'(t,R)] \\ & F'(A' \copyright t, R \backslash H) \end{array}$$

und die Gültigkeit der Aussage 3.14 und damit, wie oben gezeigt, die Behauptung. □

3.3 Kombinierte Systeme

Bei der Definition der implizierten Failureeigenschaften wurde für die Eingabebereitschaft $F_T^I(t, R)$ eine sehr strenge Anforderung gestellt, nämlich die Forderung nach *unbeschränkter* Eingabebereitschaft:

$$F_T^I(t, R) \equiv I \cap R = \emptyset \tag{3.17}$$

Die Forderung, daß eine Eingabe *nie* abgewiesen werden darf, ist jedoch zu stark, um von einem Prozeß im Sinne von Definition A.3.1 erfüllt zu werden. Eine wichtige Eigenschaft der Prozesse des unendlichen Failuremodells besteht in der Terminierungseigenschaft (Definition A.3.1, 5). Damit eignet sich die Forderung 3.17 *nicht* für die Realisierung durch einen Prozeß des unendlichen Failuremodells.

Die ist aber für die in dieser Arbeit vorgeschlagenen Vorgehensweise unproblematisch, da diese zum Ziel hat, die Eingabebereitschaft an *alle* Komponenten eines Systems soweit abzuschwächen, bis eine effiziente und damit im allgemeinen endliche Pufferung von Nachrichten ausreicht, und damit für jede Komponente diese Anforderung aufgehoben wird. Da also die Forderung nach unbeschränkter Eingabebereitschaft nur den Ausgangspunkt für die anschließende Abschwächung darstellt, ist deren Realisierbarkeit für diese Vorgehensweise nicht von Bedeutung.

Trotzdem kann es auch wünschenwert sein, asynchron kommunizierende Komponenten mit synchron kommunizierenden in einem System gemeinsam einzusetzen, also im Sinne der Pufferung kombinierte Systeme zu konstruieren. Für diese asynchron kommunizierenden Komponenten ist dann eine Bestimmung der minimalen Anforderungen an die Eingabebereitschaft nicht nötig oder nur überflüssig. Die Eingliederung von Prozessen, bei denen die Forderung 3.17 erhoben wird, ist jedoch - wie oben erwähnt - aus semantischen Gründen nicht möglich.

Um dies schematisch zu ermöglichen, können Abschwächungen von Forderung 3.17 definiert werden, die einerseits eine Beschreibung asynchroner Komponenten als Prozesse erlauben, und andererseits keine Bestimmung minimaler Anforderungen an die Eingabebereitschaft erfordern. Dabei existieren unterschiedliche Möglichkeiten, wie die Forderung nach unbeschränkter Eingabebereitschaft eingeschränkt werden kann. Im folgenden wird die *aktionsweise* Parallelität der Eingabe eingeführt. Darüberhinaus wird eine weitere Form kurz diskutiert.

3.3.1 Aktionsweise Parallelität der Eingabe

Die aktionsweise Eingabebereitschaft einer Komponente oder eines Prozesses ist die schwächste Form der Abschwächung der unbeschränkten Eingabebereitschaft, die durch einen Prozeß im Sinne von A.3.1 beschreibbar ist. Bei dieser Form wird gefordert, daß

ein Prozeß stets bereit sein muß, eine Nachricht bzw. Aktion entgegenzunehmen, solange diese Aktion bisher nur *endlich oft* entgegengenommen wurde. Formal läßt sich diese Anforderung definieren als

$$\forall a \in I.(a©t \in A^* \Rightarrow a \notin R) \tag{3.18}$$

Daß diese Anforderung tatsächlich eine Abschwächung von 3.17 ist, folgt sofort, wenn 3.17 in der Form

$$\forall a \in I.a \notin R$$

dargestellt wird. Diese schwächere Anforderung an die Eingabebereitschaft ist im unendlichen Failuremodell prinzipiell durchaus realisierbar, wie das Beispiel A.4.1 im Anhang A zeigt.

Diese Form der Abschwächung erfüllt neben ihrer Realisierbarkeit auch eine weitere wichtige Eigenschaft: Wird in der in 3.2.1 beschriebenen Transformation die unbeschränkte Eingabebereitschaft durch die aktionsweise Parallelität ersetzt, so ist diese Transformation ebenfalls verträglich mit der Implikation hinsichtlich Parallelkomposition und Abstraktion. Der Nachweis dieser Verträglichkeit folgt aus dem Axiom 6 der Definition A.3.1 für Prozesse der unbeschränkten Failuresemantik:

$$(t, R) \in I \wedge a©t \notin A^* \Rightarrow (t, R \cup \{a\}) \in I$$

Dieses Axiom der Serialität von Aktionen drückt aus, daß eine Aktion, die in einem Ablauf eines Prozesses unbeschränkt oft auftritt, von diesem Prozeß abgelehnt werden kann.

Definition 3.3.1 (Implizierte Failureeigenschaft 2) Sei $A = I \cup O$ das Alphabet eines asynchronen Prozesses mit Eingabealphabet I und Ausgabealphabet O, sowie $T : A^\omega \to \mathbb{B}$ eine Spureigenschaft. Dann heißt $F : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$ mit

$$F_T(t, R) \stackrel{\text{def}}{=} F^I(t, R) \wedge F_T^O(t, R)$$

wobei

$$\begin{aligned} F^I(t, R) &\stackrel{\text{def}}{=} \forall i \in I.(i©t \in A^* \Rightarrow i \notin R) \\ F_T^O(t, R) &\stackrel{\text{def}}{=} O \subseteq R \Rightarrow T(t) \end{aligned}$$

die *durch T implizierte Failureseigenschaft mit aktionsweiser Parallelität.* ∘

Da die Verträglichkeit im Fall der implizierten Failureeigenschaften mit aktionsweiser Parallelität im wesentlichen identisch ist mit der in Satz 3.2.3 gegebenen Formulierung, wird auf eine entsprechende Neuformulierung dieses Satzes verzichtet. Statt dessen wird im folgenden Satz die auf die Kernaussage reduzierte Fassung wiedergegeben.

Satz 3.3.1 (Verträglichkeit) Die Transformation von Spureigenschaften mittels der implizierten Failureeigenschaften mit aktionsweiser Parallelität ist verträglich mit der Parallelkomposition hinsichtlich der Implikation. •

Beweis 3.3.1 (Satz 3.3.1) Der Nachweis der Verträglichkeit der Anforderung an die Eingabebereitschaft erfolgt entsprechend wie in Beweis 3.2.3. Für den Nachweis dieses Satzes reicht es daher, die Verträglichkeit der Ausgabebereitschaft nachzuweisen. Sei dazu im folgenden

$$F_1(A_1 © t, R_1) \wedge F_2(A_2 © t, R_2) \wedge O_1 \cup O_2 \subseteq R_1 \cup R_2$$

Damit folgt

$$\begin{array}{ll} & O_1 \subseteq R_1 \cup R_2 \\ \Longrightarrow & \text{[Prädikatenlogik]} \\ & O_1 \backslash R_1 \subseteq R_2 \\ \Longrightarrow & \text{[Definition 3.3.1]} \\ & \forall o \in O_1 \backslash R_1 . o © t \notin A^* \\ \Longrightarrow & \text{[Definition A.3.1, Axiom 6]} \\ & F_1(A_1 © t, R_1 \cup (O_1 \backslash R_1) \\ \Longrightarrow & \text{[Definition “\”]} \\ & F_1(A_1 © t, O_1 \cup R_1) \\ \Longrightarrow & \text{[Definition A.3.1, Axiom 3]} \\ & F_1(A_1 © t, O_1) \end{array}$$

Eine entsprechende Behandlung von F_2 führt analog zu $F_2(A_2 © t, O_2)$. Damit gilt

$$F_1(A_1 © t, O_1) \wedge F_2(A_2 © t, O_2)$$

Entsprechend wie in Beweis 3.2.3 läßt damit auch $T(t)$ und insgesamt die Behauptung herleiten. □

Wie für den Fall der Parallelkomposition läßt sich die entsprechende Aussage auch für den Fall der Abstraktion formulieren:

Satz 3.3.2 (Verträglichkeit) Die Transformation von Spureigenschaften mittels der implizierten Failureeigenschaften mit aktionsweiser Parallelität ist verträglich mit der Abstraktion hinsichtlich der Implikation. •

Der Beweis erfolgt entsprechend wie im Fall des Beweises 3.2.4.

Schließlich zeigt die Verträglichkeit dieser Form der implizierten Failureeigenschaften, daß auch die Anforderung nach unbeschränkter Eingabebereitschaft ein geeigneter Ausgangspunkt für die Kapitel 4 beschriebene Vorgehensweise zur Abschwächung der Eingabebereitschaft darstellt. Da die Variante der implizierten Failureanforderungen mit der schwächere Anforderung 3.18 ebenfalls hinsichtlich der Parallelkomposition und Abstraktion verträglich ist, kann auch diese Beschreibung als Ausgangspunkt für die Abschwächung der Eingabebereitschaft der Komponenten verwendet werden.

3.3.2 Weitere Form der Abschwächung

Nachrichtenorientierte Kommunikation wird im allgemeinen kanalorientiert realisiert - wie beispielweise bei den in Abschnitt 1.1 erwähnten Ansätzen. Wie in Abschnitt 4.5 detailliert

besprochen wird, ist dabei auf einem Kanal jeweils nur sequentielle Übertragung möglich. Damit liegt es für diese Art von Systemen nahe, die Forderung nach unbeschränkter Eingabebereitschaft an diese vorgegebene Zerlegung der Eingabealphabete anzupassen und so eine Abschwächungen der aktionsweisen Parallelität zu erhalten.

Die aktionsweise Parallelität ist, wie oben beschrieben, im unendlichen Failuremodell realisierbar und verträglich, und erlaubt es, synchron und asynchron kommunizierende Komponenten in einem System zu kombinieren. Trotzdem ist sie intuitiv noch immer unbefriedigend. Jede Aktion wird unabhängig von den anderen Eingabeaktionen betrachtet. Im kanalorientierten Ansatz interpretiert, entspricht dies einem System, bei dem jeder Aktion ein eigener Kanal zugeordnet wird.

Innerhalb eines Systems von kommunizierenden Komponenten treten Aktionen jedoch im allgemeinen nicht alle parallel zueinander auf. Sie werden auch zum Teil sequentiell erzeugt, insbesondere wenn sequentielle Prozesse untereinander Nachrichten austauschen. In diesem Fall lassen sich die Aktionsmengen der Prozesse zu Kanälen bündeln. Wie Abschnitt 4.5.2 zeigt, sind dazu die Ausgabealphabete der sequentiellen Prozesse sowie alle feineren Zerlegungen davon geeignete Kanalalphabete. Statt 3.18 ist es dann ausreichend, nur noch die Eingabebereitschaft für einen Kanal nach einer endlichen Anzahl von Eingaben auf diesem Kanal zu fordern.

3.4 Transformation sequentieller Prozesse

Wie in Abschnitt 3.2.2 bereits angesprochen, stellen die implizierten Failureeigenschaften keine explizite Beschreibung der Failuremenge eines zur Spurbeschreibung äquivalenten Prozesses dar. Dies liegt dran, daß erst die implizierte Failurebeschreibung zusammen mit den Eigenschaften der Prozesse der unendlichen Failuresemantik den Prozeß charakterisiert.

Die Bestimmung der minimal benötigten Eingabebereitschaft einer Komponente eines Gesamtsystems - wie in Kapitel 4 beschrieben wird - erfolgt aber in Abhängigkeit der Verhaltensbeschreibung der Systemkomponenten. Nur eine detaillierte Beschreibung des Ausgabeverhaltens der einzelnen Systemkomponenten liefert mit dem dort beschriebenen Verfahren die gewünschte minimale Eingabebereitschaft. Da aber die Prozesse der unendlichen Failuresemantik gegenüber den in 3.2.1 beschriebenen implizierten Failureeigenschaften zusätzlich eingeschränkt sind, ist eine schärfere Darstellung der Failureeigenschaften nötig.

Daher ist es sinnvoll, die bisher in den implizierten Failureeigenschaften implizit beschriebene Verhaltensweise explizit zu formulieren. Dazu ist es nötig, zusätzliche Anforderungen an die Spurbeschreibungen zu charakterisieren und mit dieser das allgemeine Schema der implizierten Failureeigenschaft zu verschärfen. In Abschnitt 2.2.5 wurden zwei Klassen von Spurbeschreibungen asynchroner sequentieller Prozesse, nämlich die monotonen und

die unterbrechbaren Prozesse, eingeführt. Für diese beiden Prozeßarten wird in diesem Abschnitt eine Transformation in eine explizite Failuredarstellung angegeben. Diese explizite Failuredarstellung erfüllt nicht nur die Eigenschaften der implizierte Failurebeschreibung. Weiterhin ist sie für die in Kapitel 4 angesprochene Vorgehensweise ausreichend detailliert, da sie die Ausgabebereitschaft nicht nur für das gesamte Ausgabealphabet pauschal, sondern für jede Ausgabe einzeln beschreibt.

In den folgenden Abschnitten werden explizite Darstellungen der implizierten Failureeigenschaften für monotone und unterbrechbare Prozesse eingeführt. Für diese Darstellungen wird die Bedeutung im Entwicklungsprozeß kurz erläutert, und deren Anwendung jeweils in einem Beispiel gezeigt.

3.4.1 Failuredarstellung monotoner Prozesse

Ist die Beschreibung der Eigenschaft eines asynchronen Systems oder Komponente auf der Spurebene durch eine monotone relationale Beschreibung (P, C) gegeben, so läßt sich für diese Eigenschaft eine Darstellung der implizierten Failureeigenschaft angeben, die die partiellen Abläufe und die implizierte Ausgabebereitschaft auf der Ebene der Failuredarstellung explizit charakterisiert. Dazu wird in Definition 3.2.1 die Minimalanforderung an die Ausgabebereitschaft 3.6 durch eine detailliertere Beschreibung der Ausgabebereitschaft sowie durch eine explizite Beschreibung der Eigenschaften der partiellen Abläufe ersetzt. Für die Neufassung der Definition 3.2.1 ist daher zunächst die Definition dieser Charakterisierung der Ausgabebereitschaft und der partiellen Abläufe von sequentiellen Prozessen mit monotoner, sequentieller Beschreibung nötig.

Definition 3.4.1 (Partielle Abläufe und Ausgabebereitschaft) Sei (P, C) eine relationale Spurdarstellung eines asynchronen Prozesses mit Eingabalphabet I und Ausgabealphabet O. Dann heißen $S_{P,C} : I \cup O)^\omega \to \mathbb{B}$ und $F^O_{P,C} : (I \cup O)^\omega \times \mathbb{P}(I \cup o) \to \mathbb{B}$ mit

$$\begin{array}{lcl} S_{P,C}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.P(I©s, O©s) \\ F^O_{P,c}(t, R) & \stackrel{\text{def}}{=} & \neg C(I©t, O©) \Rightarrow \exists o \in O.o \notin R \wedge P(I©t, (O©t) \circ o) \end{array}$$

die Charakterisierung der *partiellen Abläufe* bzw. der *Ausgabebereitschaft* der Failuredarstellung von (P, C). ∘

Die Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft der Failuredarstellung von (P, C) liefert eine Verschärfung der Ausgabebereitschaft der implizierten Failureeigenschaft der dazugehörigen Spurdarstellung von (P, C):

Satz 3.4.1 (Charakterisierung) Sei (P, C) die monotone relationale Darstellung eines Prozesses mit Eingabealphabet I und Ausgabealphabet O, sowie T die Spurmenge von (P, C). Sei F^O_T die Charakterisierung der Ausgabebereitschaft der durch T implizierten Failureeigenschaft, sowie $S_{P,C}$ und $F^O_{P,C}$ die Charakterisierung der partiellen Abläufe und

der Ausgabebereitschaft der Failuredarstellung von (P, C). Dann gilt für alle Prozesse Q der unendlichen Failuresemantik mit Alphabet $I \cup O$

$$Q \text{ sat } S_{P,C}(t) \wedge F^O_{P,C}(t, R) \Rightarrow Q \text{ sat } F^O_T(t, R) \tag{3.19}$$

•

Beweis 3.4.1 (Satz 3.4.1) Aussage 3.19 läßt sich nicht nur für Prozesse der unendlichen Failuresemantik, sondern ganz allgemein für Failuremengen mit Alphabet $I \cup O$ zeigen. Es gilt

$$\begin{array}{ll} & S_{P,C}(t) \wedge F^O_{P,C}(t, R) \wedge O \subseteq R \\ \Longrightarrow & \text{[Definition 3.4.1]} \\ & (\forall s \sqsubseteq t.P(I \copyright s, O \copyright s)) \wedge \\ & (\neg C(I \copyright t, O \copyright t) \Rightarrow \exists o \in O.o \notin R \wedge P(I \copyright t, (O \copyright t) \circ o)) \wedge \\ & O \subseteq R \\ \Longrightarrow & \text{[Aussagenlogik, Definition } \subseteq\text{]} \\ & (\forall s \sqsubseteq t.P(I \copyright s, O \copyright s)) \wedge C(I \copyright t, O \copyright t) \\ \Longrightarrow & \text{[Definition 2.2.6]} \\ & T(t) \end{array}$$

also insgesamt

$$S_{P,C}(t) \wedge F^O_{P,C}(t, R) \wedge O \subseteq R \Rightarrow T(t) \tag{3.20}$$

Mit Definition 3.2.1 folgt aus 3.20

$$S_{P,C}(t) \wedge F^O_{P,C}(t, R) \Rightarrow F^O_T(t, R)$$

und damit die Behauptung. □

Satz 3.4.1 drückt also aus, daß die Charakterisierungen der partiellen Abläufe $S_{P,C}$ und der Ausgabebereitschaft $F^O_{P,C}$ der Failuredarstellung von (P, C) hinreichend sind, um die Ausgabebereitschaft der implizierten Failureeigenschaft F^O_T sicherzustellen. Für die Bestimmung der *minimalen* Eingabebereitschaft ist es jedoch darüberhinaus wesentlich, daß $F^O_{P,C}$ das Verhalten ausreichend detailiert beschreibt. Diese Aussage ist im folgenden Satz formuliert. Dort wird festgestellt, daß $F^O_{P,C}$ nicht schwächer formuliert werden kann, ohne die Ausgabebereitschaft F^O_T zu verletzen. Damit ist $F^O_{P,C}$ für die sequentiellen Prozesse dargestellt durch die monotone Relation (P, C) nicht nur eine hinreichende, sondern auch notwendige Anforderung an die Ausgabebereitschaft. Von den Prozessen der unendlichen Failuresemantik werden also wirklich nur solche Ausgaben ausgeführt, die auch laut der expliziten implizierten Failureeigenschaft ausgeführt werden können.

Satz 3.4.2 (Partielle Abläufe und Ausgabebereitschaft) Sei (P, C) die monotone relationale Darstellung eines Prozesses mit Eingabealphabet I und Ausgabealphabet O, sowie T die Spurmenge von (P, C). Sei F^O_T die Charakterisierung der Ausgabebereitschaft der durch T implizierten Failureeigenschaft, sowie $S_{P,C}$ und $F^O_{P,C}$ die Charakterisierung

der partiellen Abläufe und der Ausgabebereitschaft der Failuredarstellung von (P, C). Dann gilt für alle Prozesse Q der unendlichen Failuresemantik mit Alphabet $I \cup O$

$$Q \text{ sat } F_T^O \wedge S_{P,C} \Rightarrow Q \text{ sat } F_{P,C}^O$$

•

Für den Beweis wird neben den Definitionen der Spurmenge, der implizierten Failureeigenschaft und der Charakterisierungen auch die Eigenschaften eines Prozesses wie in A.3.1 beschrieben, benötigt.

Beweis 3.4.2 (Satz 3.4.2) Der Beweis wird per Widerspruch geführt, also unter der Annahme

$$Q \text{ sat } F_T^O \wedge S_{P,C} \wedge \neg(Q \text{ sat } F_{P,C}^O)$$

Dazu wird die Existenz von t und R angenommen mit $\neg F_{P,C}^O(t, R)$, also laut Definition 3.4.1

$$\neg C(I©t, O©t) \wedge \forall o \in O.(o \in R \vee \neg P(I©t, (O©t) \circ o)) \tag{3.21}$$

Wegen Q sat $S_{P,C}$ folgt aus 3.21

$$\neg C(I©t, O©t) \wedge P(I©t, O©t) \tag{3.22}$$

Wegen der Eigenschaft 2.4 der relationalen Darstellung folgt aus 3.22

$$O©t \in O^* \tag{3.23}$$

Wegen der Stetigkeit von "©" folgt aus 3.23 die Existenz eines r mit

$$r \in A^* \wedge r \sqsubseteq t \wedge O©r = O©t \tag{3.24}$$

Wegen der Monotonieeigenschaft 2.6 der relationalen Darstellung folgt aus 3.21 und 3.24 mit der Endlichkeit von O und dem Axiom 2 der Definition A.3.1 der unendlichen Failuresemantik

$$\forall s \in A^*.r \circ s \sqsubseteq t \Rightarrow \forall o \in O.(o \in R \vee \neg P(I©(r \circ s), (O©(r \circ s)) \circ o)) \tag{3.25}$$

Aus dem Axiom 7 der Definition A.3.1 der unendlichen Failuresemantik folgt mit 3.25

$$(t, R \cup O) \in Q \tag{3.26}$$

Wegen Q sat $F_T^O(t, R)$ folgt aus 3.26, der Eigenschaft 3.6 der Definition 3.2.1 der implizierten Failureeigenschaft sowie der Definition 2.2.6 der Spurmenge der relationalen Beschreibung

$$C(I©t, O©t) \tag{3.27}$$

Aus dem Widerspruch zwischen 3.22 und 3.27 folgt die Behauptung. □

Mit Satz 3.4.2 wurde gezeigt, daß $F^O_{P,C}$ nur das Ausgabeverhalten eines Prozesses des unendlichen Failuremodells charakterisiert, das für die Sicherstellung des Ausgabeverhaltens eines durch die monotone relationale Darstellung (P, C) beschriebenen Prozesses des Spurmodells gefordert wird. Insgesamt folgt also damit:

- $S_{P,C}$ charakterisiert genau die partiellen Abläufe von (P, C)
- F^I charakterisiert eine realisierbare Anforderung an die Eingabebereitschaft
- $F^O_{P,C}$ charakterisiert genau die Ausgaben, die vom durch (P, C) beschriebenen Prozeß ausgeführt werden können.

Um die Charakterisierung der ausführbaren Ausgaben mittels $F^O_{P,C}$ nochmals zu veranschaulichen, wird im nächsten Satz gezeigt, welche Ausgaben entsprechend $F^O_{P,C}$ stets zurückgewiesen werden können:

Satz 3.4.3 (Refusalabschluß) Sei $F^O_{P,C}$ die Charakterisierung der Ausgabebereitschaft der Failuredarstellung der relationalen Beschreibung (P, C) eines sequentiellen Prozesses mit Eingabealphabet I und Ausgabealphabet O. Dann gilt

$$F^O_{P,C}(t, R) \wedge \neg P(I©t, (O©t) \circ o) \Rightarrow F^O_{P,C}(t, R \cup \{o\})$$

•

Beweis 3.4.3 (Satz 3.4.3) Der Beweis folgt direkt aus der Anwendung von Definition 3.4.1 durch Kontraposition im zweiten Glied der Prämisse:

$$\begin{array}{ll} & F^O_{P,C}(t, R) \wedge \neg F^O_{P,C}(t, R \cup \{o\}) \\ \Longrightarrow & \text{[Definition 3.4.1, Aussagenlogik]} \\ & F^O_{P,C}(t, R) \wedge \neg F^O_{P,C}(t, R \cup \{o\}) \wedge \neg C(I©t, O©t) \\ \Longrightarrow & \text{[Definition 3.4.1]} \\ & (\exists o \in O.o \notin R \wedge P(I©t, (O©t) \circ o)) \wedge \\ & (\forall o' \in O.o' \notin R \cup \{o\} \Rightarrow \neg P(I©t, (O©t) \circ o')) \\ \Longrightarrow & \text{[Prädikatenlogik]} \\ & P(I©t, (O©t) \circ o) \end{array}$$

□

Satz 3.4.3 zeigt, daß nur solche Ausgaben o nicht nach einem Ablauf t abgewiesen werden können, die entsprechend der Charakterisierung der partiellen Abläufe $P(I©t, (O©t) \circ o)$ auch mögliche Ausgaben sind.

Da nun zu einer Spurbeschreibung eine geeignete explizite Failurebeschreibung vorliegt, rechtfertigt das die Einführung der expliziten implizierten Failuredarstellung in der folgenden Definition. Durch die genannten Eigenschaften ist diese Form der Failuredarstellung ein geeigneter Ausgangspunkt für die in Kapitel 4 beschriebene Vorgehensweise.

Definition 3.4.2 (Explizite implizierte Failureeigenschaft) Sei (P, C) die monotone relationale Darstellung eines Prozesses mit Eingabealphabet I und Ausgabealphabet

O. Sei $S_{P,C}$ und $F^O_{P,C}$ die Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft der Failuredarstellung von (P, C). Dann heißt $F_{P,C}$ mit

$$F_{P,C}(t, R) \stackrel{\text{def}}{=} S_{P,C}(t) \wedge F^I(t, R) \wedge F^O_{P,C}(T, R)$$

die *explizite durch* (P, C) *implizierte Failureeigenschaft mit aktionsweiser Parallelität.* ∘

Beispiel 3.4.1 (Empfänger) Im folgenden wird für die Empfängerkomponente aus Beispiel 2.2.3 die Failuredarstellung bestimmt. Insbesondere wird auch gezeigt, wie sich diese von einer allgemeineren Transformation hinsichtlich der Ausgabeparallelität unterscheidet.

Mit den Definition der Alphabete und Hilfsfunktionen aus Beispiel 2.2.3 wurde dort das Verhalten des Empfängers mit der relationalen Beschreibung der partiellen und totalen Ausgaben definiert. Dabei wurde der Empfänger durch zwei Prozesse dargestellt, jeweils einen für jeden Ausgabekanal. Damit ergeben sich die folgenden partiellen und vollständigen Abläufe:

$$\begin{array}{lcl}
P_{\mathsf{Out}}(i, o) & \stackrel{\text{def}}{=} & \mathsf{data}^*(o) \sqsubseteq \mathsf{data}^*(\mathsf{dup}(i)) \\
P_{\mathsf{SAck}}(i, o) & \stackrel{\text{def}}{=} & \mathsf{bit}^*(o) \sqsubseteq \mathsf{bit}^*(i) \\
C_{\mathsf{Out}}(i, o) & \stackrel{\text{def}}{=} & \mathsf{data}^*(o) = \mathsf{data}^*(\mathsf{dup}(i)) \\
C_{\mathsf{SAck}}(i, o) & \stackrel{\text{def}}{=} & \mathsf{bit}^*(o) = \mathsf{bit}^*(i)
\end{array}$$

Da die Empfängerkomponente als Parallelkomposition dargestellt wurde, wird auch die Failuredarstellung als Parallelkomposition der Prozesse $F_{P_{\mathsf{Out}},C_{\mathsf{Out}}}$ und $F_{P_{\mathsf{SAck}},C_{\mathsf{SAck}}}$ dargestellt. Die zugehörige Failuredarstellung des Empfängers ist dabei - entsprechend Definition 3.4.1 sowie vereinfachender Umformungen -

$$\begin{array}{lcl}
S_{P_{\mathsf{Out}},C_{\mathsf{Out}}}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.\, \mathsf{data}^*(Out©s) \sqsubseteq \mathsf{data}^*(\mathsf{dup}(RData©s)) \\
S_{P_{\mathsf{SAck}},C_{\mathsf{SAck}}}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.\, \mathsf{bit}^*(SAck©s) \sqsubseteq \mathsf{bit}^*(RData©s) \\
F^{Out}_{P_{\mathsf{Out}},C_{\mathsf{Out}}}(t, R) & \stackrel{\text{def}}{=} & \mathsf{data}^*(Out©t) \sqsubset \mathsf{data}^*(\mathsf{dup}(RData©t)) \Rightarrow \\
 & & \quad \exists o \in Out. \\
 & & \quad\quad (o \notin R \wedge \mathsf{data}^*((Out©t) \circ o) \sqsubseteq \mathsf{data}^*(\mathsf{dup}(RData©t))) \\
F^{SAck}_{P_{\mathsf{Out}},C_{\mathsf{Out}}}(t, R) & \stackrel{\text{def}}{=} & \mathsf{bit}^*(SAck©t) \sqsubset \mathsf{bit}^*(RData©t) \Rightarrow \\
 & & \quad \exists o \in SAck.(o \notin R \wedge \mathsf{bit}^*((SAck©t) \circ b) \sqsubseteq \mathsf{bit}^*(RData©t)) \\
F_{P_{\mathsf{Out}},C_{\mathsf{Out}}}(t, R) & \stackrel{\text{def}}{=} & S_{P_{\mathsf{Out}},C_{\mathsf{Out}}}(t) \wedge F^{RData}_{P_{\mathsf{Out}},C_{\mathsf{Out}}}(t, R) \wedge F^{Out}_{P_{\mathsf{Out}},C_{\mathsf{Out}}}(t, R) \\
F_{P_{\mathsf{SAck}},C_{\mathsf{SAck}}}(t, R) & \stackrel{\text{def}}{=} & S_{P_{\mathsf{SAck}},C_{\mathsf{SAck}}}(t) \wedge F^{RData}_{P_{\mathsf{SAck}},C_{\mathsf{SAck}}}(t, R) \wedge F^{SAck}_{P_{\mathsf{SAck}},C_{\mathsf{SAck}}}(t, R)
\end{array}$$

◇

3.4.2 Failuredarstellung unterbrechbarer Prozesse

Ähnlich wie im Fall der monotonen Prozesse muß eine Transformation in eine explizite Failuredarstellung auch für unterbrechbare Prozesse definiert werden. Hier zeigt sich, daß

dazu die gleiche Transformation wie im Falle monotoner relationaler Prozeßbeschreibungen verwendet werden kann: die Eigenschaften der für monotone relationale Darstellungen (P, C) definierten Charakterisierungen der partiellen Abläufen und der Ausgabebereitschaft der Failuredarstellung von (P, C) lassen sich direkt auf die relationale Darstellung unterbrechbarer Prozesse übertragen. Die Definition 3.4.1 der Charakterisierungen der partielle Abläufe und Ausgabebereitschaft der Failuredarstellung von (P, C) ist bereits unabhängig von der Monotonieeigenschaft der relationalen Darstellung formuliert, und daher auch direkt für unterbrechbare Prozesse anwendbar. Damit lassen sich die Sätze 3.4.1 und 3.4.2 auch für unterbrechbare Prozesse formulieren.

Satz 3.4.4 (Charakterisierung) Sei (P, C) die relationale Darstellung eines unterbrechbaren Prozesses mit Eingabealphabet I und Ausgabealphabet O, sowie T die Spurmenge von (P, C). Sei F_T^O die Charakterisierung der Ausgabebereitschaft der durch T implizierten Failureeigenschaft, sowie $S_{P,C}$ und $F_{P,C}^O$ die Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft der Failuredarstellung von (P, C). Dann gilt für alle Prozesse Q der unendlichen Failuresemantik mit Alphabet $I \cup O$

$$Q \text{ sat } S_{P,C}(t) \wedge F_{P,C}^O(t, R) \Rightarrow Q \text{ sat } F_T^O(t, R)$$

•

Beweis 3.4.4 (Satz 3.4.4) Der Beweis 3.4.1 von Satz 3.4.1 kann hier unmittelbar verwendet werden, da nur mit Definition 2.2.6 nur eine allgemeine Eigenschaft relationaler Darstellungen, aber keine der Eigenschaften monotoner Darstellungen ausgenutzt wurde. □

Satz 3.4.5 (Partielle Abläufe und Ausgabebereitschaft) Sei (P, C) die relationale Darstellung eines unterbrechbaren Prozesses mit Eingabealphabet I und Ausgabealphabet O, sowie T die Spurmenge von (P, C). Sei F_T^O die Charakterisierung der Ausgabebereitschaft der durch T implizierten Failureeigenschaft, sowie $S_{P,C}$ und $F_{P,C}^O$ die Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft der Failuredarstellung von (P, C). Dann gilt für alle Prozesse Q der unendlichen Failuresemantik mit Alphabet $I \cup O$

$$Q \text{ sat } F_T^O \wedge S_{P,C} \Rightarrow Q \text{ sat } F_{P,C}^O$$

•

Beweis 3.4.5 (Satz 3.4.5) Der Beweis von Satz 3.4.2 kann durch Verwendung von 2.7 statt 2.6 eingesetzt werden. Wegen 3.23 bleibt hier der Schluß 3.25 weiterhin gültig. □

Da also für die relationale Darstellung unterbrechbarer Prozesse die entsprechenden Aussagen wie im Falle monotoner Prozesse gelten, kann Definition 3.4.2 auf unterbrechbare Prozesse erweitert werden:

Definition 3.4.3 (Explizite implizierte Failureeigenschaft) Sei (P, C) die relationale Darstellung eines unterbrechbaren Prozesses mit Eingabealphabet I und Ausgabealphabet O. Sei $S_{P,C}$ und $F_{P,C}^O$ die Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft der Failuredarstellung von (P, C). Dann heißt $F_{P,C}$ mit

$$F_{P,C}(t, R) \stackrel{\text{def}}{=} S_{P,C}(t) \wedge F^I(t, R) \wedge F_{P,C}^O(T, R)$$

die *explizite durch* (P, C) *implizierte Failureeigenschaft mit aktionsweiser Parallelität.* $\circ$

Das folgende Beispiel zeigt die Anwendung der expliziten implizierten Failureeigenschaften von Definition 3.4.3 auf einen unterbrechbaren Prozeß.

Beispiel 3.4.2 (Sender) Die Anwendung der expliziten Transformation auf die in Beispiel 2.2.4 angegebene Definition der relationalen Prozeßbeschreibung eines unterbrechbaren Prozesses liefert - unter Verwendung einiger prädikatenlogischer Umformungen -

$$
\begin{array}{lcl}
S_{P,C}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.(\mathsf{data}^*(\mathsf{dup}(SData©s)) \sqsubseteq \mathsf{data}^*(In©s) \wedge \\
& & \quad \#\,\mathsf{dup}(SData©s) \leq \#\,\mathsf{dup}(RAck©s) + 1 \wedge \\
& & \quad (\#\,\mathsf{dup}(SData©s) > \#\,\mathsf{dup}(RAck©s) \vee \\
& & \quad SData©s \in SData^*)) \\
F^{SData}_{P,C}(t, R) & \stackrel{\text{def}}{=} & (\#In©t > \#\,\mathsf{dup}(SData©t) \wedge SData©t \in SData^*) \Rightarrow \\
& & \exists o \in O.(\mathsf{data}^*(\mathsf{dup}((SData©t) \circ o)) \sqsubseteq \mathsf{data}^*(In©t) \wedge \\
& & \quad \#\,\mathsf{dup}((SData©t) \circ o) \leq \#\,\mathsf{dup}(RAck©t) + 1 \wedge o \not\subseteq R) \\
F_{P,C}(t, R) & \stackrel{\text{def}}{=} & S_{P,C}(t) \wedge F^{In}_{P,C}(t, R) \wedge F^{SData}_{P,C}(t, R)
\end{array}
$$

$\diamond$

Parallele Prozesse

In den beiden vorherigen Abschnitten wurde gezeigt, wie zur Spurbeschreibung sequentiell ablaufender Prozesse die entsprechende Failuredarstellung gewonnen werden kann. Dies ist für den Wechsel von der Spur- zur Failurebeschreibung für sequentiell ablaufende Systemkomponenten ausreichend. Für parallel ablaufende Komponenten ist diese Transformation jedoch nicht ausreichend, da die Transformation die Failurebeschreibung eines sequentiellen Prozesses zum Ziel hat. Dies wird insbesondere durch die Anforderung an die partiellen Abläufe ausgedrückt.

Wie zu Beginn von Abschnitt 3.4 besprochen, sollten jedoch solche Komponenten bereits auf der Spurebene entsprechend der in 2.6.2 Weise in parallel ablaufende, sequentielle Prozesse aufgespalten werden. Dann wird die Transformation der Spurbeschreibung in die Failurebeschreibung auf die einzelnen sequentiellen Prozesse angewendet. Die Parallelkomposition der Failurebeschreibungen liefert dann eine Failurebeschreibung der gesamten Komponente. In Beispiel 3.4.1 wurde diese Art der Transformation bereits auf den aus zwei Prozessen bestehenden Empfänger des Alternierenden-Bit-Protokolls angewendet. Im folgenden Abschnitt 3.5 wird dieser Aspekt nochmals ausführlich bei der methodischen Behandlung der Transformation aufgegriffen.

3.5 Methodischer Einsatz der Transformation

In den Abschnitten 3.2.2 und 3.4 wurden zwei unterschiedliche Schemata für den Wechsel von der Spurbeschreibung zur Failurebeschreibung angegeben.

Das erste Schema, die implizierten Failureeigenschaften aus Definition 3.2.1, beschreibt das gewünschte Verhalten auf der Ebene der Failureebene nur indirekt. Wie bereits zuvor diskutiert, stellt es ohne entsprechende explizite Charakterisierung des gewünschten Verhaltens keinen geeigneten Ausgangspunkt für das weitere Vorgehen dar - weder für die in Kapitel 4 beschriebene Bestimmung der minimalen Eingabebereitschaft, noch für eine in dieser Arbeit nicht angesprochene Umsetzung in eine Implementierung. Jedoch ist die Transformation - wie in Abschnitt 3.2.3 gezeigt - verträglich mit der Abstraktion und der Parallelkomposition.

Das zweite Schema, definiert für die Klasse monotoner und unterbrechbarer Prozesse, ist - wie in Abschnitt 3.4.1 und 3.4.2 gezeigt - eine Spezialisierung des ersten Schemas. Gleichzeitig liefert es eine explizite Darstellung der gewünschten Failureeigenschaften. Damit stellt es einen geeigneten Ausgangspunkt für die Bestimmung der minimalen Eingabebereitschaft dar, wie in Abschnitt 3.4 argumentiert, und darüberhinaus dank seiner Detailliertheit auch für eine weitere Implementierung.

Aus diesen bisher abgeleiteten Eigenschaften läßt sich damit ein methodisches Vorgehen für die zu Beginn dieses Kapitels beschriebenen ersten beiden Schritte des Entwurfsprozesses angeben. In den folgenden beiden Abschnitte wird erläutert, wie der Systementwurf auf der Spurebene und der Wechsel von der Spur- zur Failureebene methodisch sinnvoll für den Entwurf synchroner Systeme eingesetzt werden.

3.5.1 Entwurf auf der Spurebene

Wie zu Beginn des Kapitels erwähnt, stellt die Beschreibung des asynchronen Systems den Ausgangspunkt der in dieser Arbeit vorgestellten Vorgehensweise dar. Diese Beschreibung besteht dabei aus den Spurbeschreibungen der Eigenschaften eines Systems und seiner Komponenten. Dabei werden auf der Spurebene bereits alle gewünschten Eigenschaften des Systems und der Komponenten festgelegt, die den auf der asynchronen Ebene modellierten Datenfluß betreffen. Dies sind insbesondere die *partiellen* und *vollständigen Eigenschaften* des Systems und seiner Komponenten wie in 2.6.1 beschrieben. Dabei werden zur Beschreibung der partiellen Eigenschaften jene Eigenschaften herangezogen, die in der synchronen Systemsicht gelten sollen. Die vollständigen Eigenschaften beschreiben das gewünschte Ausgabeverhalten unter der idealisierten Annahme der asynchronen Kommunikation.

Da das Ziel der Entwicklung die Beschreibung eines synchronen Systems ist, wird bereits auf der Spurebene eine weitere Eigenschaften berücksichtigt, die für die spätere synchrone Realisierung entscheidend ist. Für das Nichtblockieren eines synchronen Systems ist der *Grad der Parallelität*, besonders hinsichtlich des Ausgabeverhaltens, von zentraler Bedeutung. Ein höherer Grad von Parallelität kann bei synchroner Kommunikation zu einer schwächeren benötigten Eingabebereitschaft führen. Daher ist es also sinnvoll, bereits auf der Ebene der Spurbeschreibung des Systems den gewünschten Grad der Parallelität festzulegen. In Abschnitt 2.6 wurde gezeigt, wie Prozesse als kleinste Beschreibungseinheiten eingesetzt werden können, um bereits auf der Ebene der Spurbeschreibungen den Grad

der Parallelität in einem System zu beschreiben. Dies geschieht mit der expliziten Beschreibung des Systems - oder, je nach Stufe der Entwicklung, einer Komponente - durch die Parallelkomposition sequentieller Prozesse wie in Abschnitt 2.6.2 beschrieben. Dabei werden die einzelnen sequentiellen Prozesse mit der in Abschnitt 2.2.5 eingeführten relationalen Darstellung monotoner oder unterbrechbarer Prozesse beschrieben. Beispielsweise wurden für die Modellierung des Empfängers des Alternierenden-Bit-Protokolls in Beispiel 2.2.3 zwei Prozesse benötigt, nämlich der Ausgabeprozeß mit

$$T_{\mathsf{Out}}(t) = (\forall s \sqsubseteq t.\, \mathsf{data}^*(Out©s) \sqsubseteq \mathsf{data}^*(\mathsf{dup}(RData©s))) \wedge \\ \mathsf{data}^*(Out©t) = \mathsf{data}^*(\mathsf{dup}(RData©t))$$

sowie der Quittungsprozeß mit

$$T_{\mathsf{Out}}(t) = (\forall s \sqsubseteq t.\, \mathsf{bit}^*(Out©s) \sqsubseteq \mathsf{bit}^*(\mathsf{dup}(RData©s))) \wedge \\ \mathsf{bit}^*(Out©t) = \mathsf{bit}^*(\mathsf{dup}(RData©t))$$

In der in dieser Arbeit beschriebene Vorgehensweise wird die asynchrone Systemsicht eingesetzt, um beim Entwurf der datenflußorientierten Eigenschaften von Synchronisierungsdetails zu abstrahieren. Dies erlaubt es, ein möglichst einfaches Modell zu verwenden, um die Implementierung der gewünschten Systemeigenschaften durch die entwickelten Komponenteneigenschaften nachzuweisen. Dabei wird, wie in 2.6.2 erläutert, jeder Prozeß der Systembeschreibung durch die Parallelkomposition von Komponentenprozessen und das Verbergen interner Kommunikationsaktionen implementiert. Da die einzelnen Prozesse durch ihre partiellen und vollständigen Eigenschaften beschrieben sind, wird das 2.6.1 beschriebene Verfahren zum Nachweis der Implementierungsrelation verwendet: die partiellen Eigenschaften der Komponentenprozesse implementieren die partiellen Eigenschaften der Systemprozesse; die partiellen und vollständigen Eigenschaften der Komponentenprozesse implementieren die vollständigen Eigenschaften der Systemprozesse. Mit dem Nachweis der Implementierung schließt der Entwurf auf der Spurebene. Im Fall des Alternierenden-Bit-Protokolls ist dazu zu zeigen, daß die in den Beispielen 2.2.3 und 2.2.4 gezeigten partiellen und vollständigen Eigenschaften des Senders und des Empfängers die partiellen und vollständigen Eigenschaften des Systems, nämlich

$$\begin{aligned} P(t) &= \forall s \sqsubseteq t. Out©s \sqsubseteq In©s \\ C(t) &= Out©t = In©t \end{aligned} \tag{3.28}$$

implementieren.

3.5.2 Wechsel zur Failureebene

Mit dem Wechsel von der asynchronen zur synchronen Systemsicht wird die weitere Entwicklung hin zu einem synchron kommunizierenden System vorbereitet. Dazu werden die

bisher entwickelten Spurbeschreibungen der Prozesse, sowohl die der Komponenten als auch die des Systems, in die implizierten Failureeigenschaften umgesetzt.

Entsprechend der Vorgehensweise liegen dabei die Spurbeschreibungen des Systems und der Komponenten als Parallelkomposition sequentieller Prozesse vor. Für die Umsetzung jeder Spurbeschreibung gegeben durch die relationale Darstellung (P, C) kann also die explizite Form der Failureeigenschaften, wie in Definition 3.4.2 beschrieben, verwendet werden. Die Charakterisierung der partiellen Abläufe $S_{P,C}$ entspricht dabei genau den partiellen Eigenschaften der Spurmenge der relationalen Darstellung. Dies entspricht genau der Absicht, die die partiellen Eigenschaften der Prozesse direkt von der Spur- auf die Failuredarstellung übertragen. Im Fall des Alternierenden-Bit-Protokolls lassen sich die partiellen und vollständigen Eigenschaften der in 3.28 beschriebenen Anforderungen darstellen als

$$\begin{aligned} P(i,o) &= o \sqsubseteq i \\ C(i,o) &= o = i \end{aligned}$$

Die Charakterisierung der partiellen Abläufe $S_{P,C}$ der Failuredarstellung für das System ist entsprechend

$$S_{P,C}(t) = \forall s \sqsubseteq t.Out©s \sqsubseteq In©s \tag{3.29}$$

Wie in den Sätzen 3.2.1 und 3.2.2 gezeigt, überträgt sich damit auch die Implementierung der partiellen Eigenschaften der Systemprozesse durch die partiellen Eigenschaften der Komponentenprozesse von der Spur- auf die Failureebene. Im Fall des Alternierenden-Bit-Protokolls läßt sich also die in 3.29 gezeigte Anforderung $S_{P,C}$ durch die partiellen Eigenschaften des Senders und des Empfängers implementieren.

Für die weiteren Schritte ist damit nur noch sicherzustellen, daß sich die Implementierungsbeziehung auch auf die Ein- und Ausgabebereitschaft der expliziten Form der implizierten Failureeigenschaft übertragen läßt. Laut Satz 3.4.1 erfüllen alle Komponentenprozesse mit der expliziten Failureeigenschaft auch die implizierte Eigenschaft entsprechend Definition 3.2.1. Da die implizierte Eigenschaft aber mit den Kompositionsoperatoren verträglich ist, erfüllen auch alle Systemprozesse die durch ihre Spureigenschaften implizierten Failureeigenschaften. Da sich die partiellen Eigenschaften der sequentiellen Systemprozesse von der Spur- auf die Failureebene übertragen ließen, folgt mit Satz 3.4.2 auch die Gültigkeit der expliziten durch die vollständigen Spureigenschaften implizierten Failureeigenschaften der Systemprozesse. Damit überträgt sich, wie gewünscht, die Implementierungsbeziehung von der Spur- zur Failureebene. Im Fall des Alternierenden-Bit-Protokolls lassen sich also aus den Anforderungen an die partiellen Abläufe und die Ein- und Ausgabebereitschaft des Senders und des Empfängers wie in den Beispielen 3.4.1 und 3.4.2 die Anforderungen

$$\begin{aligned} F^{In}_{P,C}(t,R) &= \forall i \in In.(i©t \in In^* \Rightarrow i \notin R) \\ F^{Out}_{P,C}(t,R) &= Out©t \neq In©t \Rightarrow \exists o \in Out.o \notin R \wedge (Out©t) \circ o \sqsubseteq In©t \end{aligned}$$

an das Gesamtsystem herleiten.

Die so erhaltenen Failurebeschreibungen der System- und Komponentenprozesse stellen damit einen geeigneten Ausgangspunkt für die weitere Entwicklung des synchronen Systems dar. Vor der Bestimmung der minimalen Eingabebereitschaft der Komponenten können dazu noch die Beschreibungen der Prozesse angepaßt werden: die Systembeschreibung durch Abschwächung der Anforderungen, die Komponenten durch eine Verschärfung der Anforderungen. Beide Möglichkeiten werden in den folgenden Abschnitten beschrieben.

3.6 Abstraktion und Verfeinerung

Wie in Abschnitt 1.2 besprochen, wird der Verzicht auf die Modellierung der engen Koppelung der kommunizierenden Prozesse als Abstraktionsmöglichkeit eingesetzt; diese erlaubt es, im ersten Entwicklungsschritt ausschließlich die Eigenschaften zu berücksichtigen, die für den am Datenfluß orientierten Entwurf relevant sind. Erst im zweiten Schritt werden diejenigen Eigenschaften betrachtet, die sich aus der engen Koppelung mittels synchroner Kommunikation ergeben. Diese Vorgehensweise liegt vielen Methoden zur schrittweisen Systementwicklung ("stepwise refinement", vgl. z.B. [Dij76]) zugrunde. Dabei spielt naturgemäß der Begriff der Abstraktion bzw. ihres Gegenstücks, der Verfeinerung, eine zentrale Rolle.

Die hier vorgestellte Vorgehensweise entspricht jedoch nicht der schrittweisen Verfeinerung, wie sie im allgemeinen zu finden ist. Ziel dieses Abschnitts ist es daher, zu erläutern, warum gerade die hier vorgestellte Vorgehensweise gewählt wurde. Dazu wird in Abschnitt 3.6.1 kurz auf die wesentlichen Techniken zur schrittweisen Verfeinerung eingegangen, und deren Modularität als wesentliche Eigenschaft herausgestellt. In Abschnitt 3.6.2 wird anschließend gezeigt, warum sich der Übergang von asynchroner zu synchroner Kommunikation wesentlich von diesen Aspekten unterscheidet und was dies für eine gezielte Vorgehensweise bedeutet.

3.6.1 Verfeinerung und Unterspezifikation

In der oben skizzierten Vorgehensweise wird die Verfeinerung eingesetzt, um die in den ersten Entwicklungsschritten verwendeten Abstraktionen zurückzunehmen; diese erlauben zwar eine elegantere Systemsicht und entsprechend eine einfachere Verifikation der Entwurfsschritte, sind aber für die weitere Entwicklung der hier betrachteten nachrichtenorientierten Systeme nicht ausreichend implementierungsnah. Nun sollen auch Eigenschaften spezifiziert werden, die bisher nicht berücksichtigt wurden. Diese prinzipielle Vorgehensweise, also das Hinzunehmen von zusätzlichen bzw. bisher nicht berücksichtigten Anforderungen, kann somit als Elimination von Unterspezifikation gesehen werden. Entsprechende Sichtweisen finden sich in vielen unterschiedlichen Verfeinerungstechniken wieder; die wichtigsten sind:

Verhaltensverfeinerung: Bei dieser Form der Verfeinerung, wie sie z.B. in [Bro94], [Hoa93a] oder [Bac93] behandelt wird, wird der Nichtdeterminismus, der aufgrund

einer lockeren Spezifikation möglich ist, eingeschränkt; dies ist beispielsweise der Fall, wenn zu einer Spezifikation, die das Verhalten bei fehlerhafter Benutzerinteraktion offen läßt, in einem zweiten Schritt eine explizite Fehlerbehandlung hinzugenommen wird.

Strukturverfeinerung: Diese Form der Verfeinerung, siehe z.B. [Bro94], [Hoa93a], oder [Bac93], bei der ein bisher unstrukturiertes System in Komponenten zerlegt wird und dadurch die interne Struktur sichtbar gemacht wird, spielt gerade bei der Modellierung verteilter Systeme eine große Rolle; eine einfaches Beispiel hierfür ist ein Transportmedium, das zuerst als elementare Einheit aufgefaßt wird, dann jedoch als System aus zwei Diensterbringern und einem einfacheren Transportmedium gesehen wird.[4]

Schnittstellen- oder Aktionsverfeinerung: Bei diesen beiden Formen der Verfeinerung, vergleiche beispielsweise [Bro94], [Ace92], oder [ZCdR92], werden Aktionen oder Nachrichten, die zuvor als elementar angesehen wurden, aufgespalten und selbst als strukturierte Einheiten aufgefaßt; beispielsweise werden Bytenachrichten als Bitoktetts interpretiert, oder ein Buchungsvorgang in das Ausstellen der Buchung und deren Bestätigung zerlegt.

3.6.2 Asynchronität als Abstraktion

Nach der Einführung der obigen Verfeinerungstechniken wird nun erläutert, daß die Rücknahme der Asynchronität als Abstraktion *nicht* als Verfeinerung im obigen Sinne zu verstehen ist. Dabei wird hier mit der Rücknahme der Asynchronität nicht der Wechsel vom auf asynchroner Kommunikation basierenden Modell zum auf synchroner Kommunikation basierenden Modell bezeichnet, also beispielsweise im hier vorliegenden Fall der Wechsel vom Spurmodell zum Failuremodell wie in den vorderen Abschnitten erläutert. Vielmehr wird mit dieser Rücknahme die Einschränkung der unbeschränkten Eingabebereitschaft verstanden, die bei den Prozessen vorgenommen wird, um effizient implementierbare Prozeßbeschreibungen zu erhalten.

Eine naheliegende Auffassung, asynchron kommunizierende Systeme als Abstraktion synchron kommunizierender zu verstehen, beruht auf der *Aktionsverfeinerung*. Dieser Ansatz wird in [Stø94] gewählt. Hierbei wird der einfache Nachrichtenaustausch im asynchron kommunizierenden System als Abstraktion von zwei Aktionen verstanden, nämlich dem Senden der Nachricht sowie der Bestätigung der Annahme oder deren Verweigerung durch den Empfänger. Diese Form der Verfeinerung entspricht im hier vorgestellten Ansatz dem Wechsel von Spur- zum Failuremodell. In der vollen Allgemeinheit ist dieser Schritt jedoch nur möglich, wenn der Empfänger nie die Annahme einer Nachricht verweigert. Anderenfalls sind jeweils entsprechende Verträglichkeitsanforderungen zwischen zwei so verfeinerten Komponenten nachzuweisen, um neben der Verfeinerung der Aktionen auch eine Verfeinerung des gesamten Systemverhaltens sicherzustellen. In [Stø94] wird dieser

[4]Siehe hierzu z.B. Varianten des alternierenden Bitprotokolls wie [DW92] oder [DS92b].

Übergang in Verfeinerungsregeln für Komponenten mit Nebenbedingungen für die Verträglichkeit zusammengefaßt. In dem in dieser Arbeit vorgestellten Ansatz wird dieser Übergang aufgespalten in die Schritte *Modellwechsel*, *Abschwächung der Anforderung* und *Herleitung* der minimalen Komponentenanforderungen. Dadurch wird im Gegensatz zu [Stø94] eine herleitungsorientierte Vorgehensweise sowie die Bestimmung minimaler Verträglichkeitsanforderungen ermöglicht.

Wie oben diskutiert, ist die Frage, ob Asynchronität als Abstraktion angesehen werden kann, eng mit der Frage der Verhaltensverfeinerung verknüpft. Anhand des folgenden Beispiels wird gezeigt, daß die Einschränkung der unbeschränkten Eingabebereitschaft *keine* Verhaltensverfeinerung, wie in Abschnitt 3.6.1 beschrieben, darstellt. Vielmehr ist die Rücknahme dieser Beschränkung eine *Vergröberung*. Dies wird auch im weiteren, insbesondere in Abschnitt 3.7, von Bedeutung sein. Trotzdem liefert die Asynchronität als Abstraktion im methodischen Sinn die Basis für die in dieser Arbeit beschriebene entwicklungstechnisch günstige Vorgehensweise, wie in Abschnitt 3.6.3 erläutert. Im folgenden Beispiel wird anhand eines sehr einfachen Prozesses, des puffernden Transportmediums, die Beziehung zwischen unbeschränkter und beschränkter Eingabebereitschaft eines Prozesses demonstriert.

Beispiel 3.6.1 (Puffernde Medium) Seien I und O zwei nichtleere disjunkte Mengen von Nachrichten, und $B_1, B_2 : ((I \cup O)^\omega \times \mathbb{P}(I \cup O)) \to \mathbb{B}$ zwei Prozesse mit

$$\begin{array}{lcl} B_1(t,R) & = & S(t,R) \wedge F_1^I(t,R) \wedge F^O(t,R) \\ S(t,R) & = & \forall s \sqsubseteq t.\, \mathsf{data}^*(O \copyright s) \sqsubseteq \mathsf{data}^*(I \copyright s) \\ F_1^I(t,R) & = & I \copyright t \in I^* \Rightarrow I \cap R = \emptyset \\ F^O(t,R) & = & \mathsf{data}^*(O \copyright t) \sqsubset \mathsf{data}^*(I \copyright t) \Rightarrow \\ & & \quad \exists o.(o \notin R \wedge (\mathsf{data}^*((O \copyright t) \circ o) \sqsubseteq \mathsf{data}^*(I \copyright t)) \end{array}$$

und

$$\begin{array}{lcl} B_2(t,R) & = & S(t,R) \wedge F_2^I(t,R) \wedge F^O(t,R) \\ F_2^I(t,R) & = & (I \copyright t \in I^* \wedge I \copyright t = O \copyright t) \Rightarrow I \cap R = \emptyset \end{array}$$

Dabei beschreibt B_1 ein Medium, das wegen F_1^I zur unbeschränkten Pufferung fähig ist; B_2 ist ein Medium, bei dem wegen F_2^I lediglich die Pufferung eines Elements sichergestellt ist. Damit kann B_1 als die, im obigen Sinne, abstraktere Variante von B_2 angesehen werden, da B_1 den Aspekt der synchronen Kommunikation zugunsten einer asynchronen Kommunikation vernachlässigt, während B_2 diesen Aspekt berücksichtigt.

Es gilt aber auch

$$\forall t \in (I \cup O)^\omega, R \subseteq (I \cup R).F_1^I(t,R) \Rightarrow F_2^I(t,R)$$

und damit auch

$$\forall t \in (I \cup O)^\omega, R \subseteq (I \cup O).B_1(t,R) \Rightarrow B_2(t,R)$$

Damit ist jedoch B_1 die stärker determinierte Variante von B_2; insgesamt stehen also die Asynchronität als Abstraktion und die Verfeinerung als Elimination von Unterspezifikation im Widerspruch. ◇

Das Beispiel zeigt, daß der Wechsel von einer asynchronen zu einer synchronen Sichtweise nicht als Verfeinerung im üblichen Sinn verstanden werden kann. Dies ist auch intuitiv leicht nachzuvollziehen: in der asynchronen Modellierung wird explizit gefordert, daß der beschriebene Prozeß zur unbeschränkten Pufferung fähig ist; die Charakterisierung des Prozesses mit beschränkter Pufferung wird damit trivialerweise erfüllt. Damit ist definitionsgemäß jeder unbeschränkt puffernde Prozeß auch ein beschränkt puffernder Prozeß.

3.6.3 Asynchronität als methodischer Aspekt

Wie in Abschnitt 3.6.2 erläutert, kann der Wechsel von asynchroner zu synchroner Systemsicht nicht als einfacher Verfeinerungschritt im üblichen formalen Sinne verstanden werden. Insbesondere läßt dieser Wechsel eine wesentliche Eigenschaft vermissen, die für eine im formalen Entwicklungsprozeß praktikable Verfeinerungsbeziehung unerläßlich ist, nämlich die Kompositionalität. Dies wird am folgenden einfachen Beispiel verdeutlicht.

Beispiel 3.6.2 (Master/Slave-System) Das Master/Slave-System, siehe Abbildung 3.1, besteht aus einer Master-Komponente sowie zwei Slave-Komponenten. Aufträge, die von der Umgebung des Systems an die Master-Komponente zugestellt werden, werden von dieser nach einem zufälligen Schema an die beiden Slavekomponenten zur Bearbeitung weitergegeben. Die bearbeiteten Aufträge werden dann wieder vom Master entgegengenommen, und an die Umgebung weitergeleitet. Im folgenden werden - mit entsprechenden

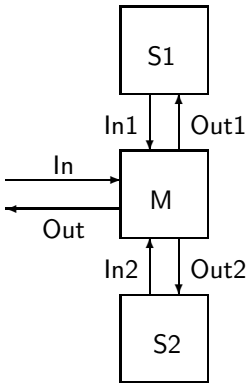

Abbildung 3.1: Das Master/Slave-System

Alphabeten und Konstruktionen wie in Beispiel 2.2.3 - die Kanäle In, In_1, In_2, Out, Out_1 und Out_2 verwendet, einschließlich entsprechender Kanalalphabete sowie eine Funktion data^* zur Selektion der Nachrichteninhalte. Für den Fall asynchroner Kommunikation läßt sich der Master als Prozeß mit Eingabealphabet $I_M = In \cup In_1 \cup In_2$ und Ausgabealphabet $O_M = Out \cup Out_1 \cup Out_2$ und der Charakterisierung $M : (I_M \cup O_M)^\omega \times \mathbb{P}(I_M \cup O_M) \to \mathbb{B}$ beschreiben mit $In_{1,2} = In_1 \cup In_2$ und $Out_{1,2} = Out_1 \cup Out_2$, wobei

$$M(t, R) \quad \stackrel{\text{def}}{=} \quad S_M(t) \wedge F_M^I(t, R) \wedge F_M^O(t, R)$$

$$
\begin{array}{lcl}
S_M(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.\ \mathsf{data}^*(Out_{1,2}©s) \sqsubseteq \mathsf{data}^*(In©s) \wedge \\
 & & \quad \mathsf{data}^*(Out©s) \sqsubseteq \mathsf{data}^*(In_{1,2}©s) \\
F_M^I(t,R) & \stackrel{\text{def}}{=} & F_M^{In}(t,R) \wedge F_M^{In_1}(t,R) \wedge F_M^{In_2}(t,R) \\
F_M^{In}(t,R) & \stackrel{\text{def}}{=} & In©t \in In^* \Rightarrow In \cap R = \emptyset \\
F_M^{In_1}(t,R) & \stackrel{\text{def}}{=} & In_1©t \in In_1^* \Rightarrow In_1 \cap R = \emptyset \\
F_M^{In_2}(t,R) & \stackrel{\text{def}}{=} & In_2©t \in In_2^* \Rightarrow In_2 \cap R = \emptyset \\
F_M^O(t,R) & \stackrel{\text{def}}{=} & (\mathsf{data}^*(Out_{1,2}©t) \sqsubset \mathsf{data}^*(In©t) \Rightarrow \\
 & & \quad \exists o \in Out_{1,2}.(o \notin R \wedge \mathsf{data}^*((Out_{1,2}©t) \circ o) \sqsubseteq \mathsf{data}^*(In©t))) \wedge \\
 & & \quad (\mathsf{data}^*(Out©t) \sqsubset \mathsf{data}^*(In_{1,2}©t) \Rightarrow \\
 & & \quad \exists o \in Out.(o \notin R \wedge \mathsf{data}^*((Out©t) \circ o) \sqsubseteq \mathsf{data}^*(In_{1,2}©t)))
\end{array}
$$

Eine Slavekomponente, beispielsweise S_1, läßt sich im Fall der asynchronen Kommunikation als Prozeß mit Eingabealphabet Out_1 und Ausgabealphabet In_1, sowie der Failuredarstellung $S_1 : (Out_1 \cup In_1)^\omega \times \mathbb{P}(Out_1 \cup In_1) \to \mathbb{B}$ beschreiben mit

$$
\begin{array}{lcl}
S_1(t,R) & \stackrel{\text{def}}{=} & S_{S_1}(t) \wedge F_{S_1}^{Out_1}(t,R) \wedge F_{S_1}^{In_1}(t,R) \\
S_{S_1}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.f^*(\mathsf{data}^*(In_1©s) \sqsubseteq \mathsf{data}^*(Out1©s) \\
F_{S_1}^{Out_1}(t,R) & \stackrel{\text{def}}{=} & Out_1©t \in Out_1^* \Rightarrow Out_1 \cap R = \emptyset \\
F_{S_1}^{In_1}(t,R) & \stackrel{\text{def}}{=} & f^*(\mathsf{data}^*(In_1©t)) \sqsubset \mathsf{data}^*(Out_1©t) \Rightarrow \\
 & & \quad \exists o \in In_1.f^*(\mathsf{data}^*((In_1©t) \circ o)) \sqsubseteq \mathsf{data}^*(Out_1©t) \wedge o \notin R
\end{array}
$$

wobei f eine Funktion zur Bearbeitung der Aufträge darstellt. Die zweite Komponente S_2 läßt sich entsprechend definieren.

Einen möglicher Übergang von der asynchronen zur synchronen Kommunikation liefert die folgende Beschränkung der Eingabebereitschaft des Masters:

$$
\begin{array}{lcl}
F_M'^I(t,R) & \stackrel{\text{def}}{=} & F_M'^{In}(t,R) \wedge F_M'^{In_1}(t,R) \wedge F_M'^{In_2}(t,R) \\
F_M'^{In}(t,R) & \stackrel{\text{def}}{=} & (In©t \in In^* \wedge In©t = (Out_1 \cup Out_2)©t) \Rightarrow In \cap R = \emptyset \\
F_M'^{In_1}(t,R) & \stackrel{\text{def}}{=} & (In_1©t \in In_1^* \wedge Out©t = (In_1 \cup In_2)©t) \Rightarrow In_1 \cap R = \emptyset \\
F_M'^{In_2}(t,R) & \stackrel{\text{def}}{=} & (In_2©t \in In_2^* \wedge Out©t = (In_1 \cup In_2)©t) \Rightarrow In_2 \cap R = \emptyset
\end{array}
$$

Damit ist der Master nur für Eingaben bereit, wenn alle bisher empfangenen Nachrichten von der Umgebung und den Slavekomponenten bereits weitergeleitet wurden. Eine mögliche Beschränkung für die Slavekomponenten ist:

$$
F_{S_1}'^{Out_1} \stackrel{\text{def}}{=} (Out©t \in Out_1^* \wedge Out_1©t = In_1©t) \Rightarrow Out_1 \cap R = \emptyset
$$

Eine Slavekomponente bearbeitet somit nur noch eine Nachricht gleichzeitig. Offensichtlich bleiben die Spuranteile der Prozeßbeschreibungen von Master- und Slavekomponenten unverändert. Dementsprechend bleiben auch die Spuranteile des Gesamtsystems durch die Beschränkung der Eingabebereitschaft unbeeinflußt. Auch die Beschreibung der Anforderungen an die Ausgabebereitschaft bleibt unverändert. Die Failurebeschreibung des

Gesamtsystem können sich aber nun nicht nur in den Eingaben sondern auch in den Ausgaben der Refusalanteile unterscheiden. Beispielsweise ist das Failurepaar

$$(in.c \circ out1.c \circ in.d, I_M \cup O_M)$$

ein mögliches Failurepaar des Systems bestehend aus der Parallelkomposition der beschränkten Komponenten. Denn es gilt zwar nicht

$$F_{S_1}^{In_1}(out1.c, \{in1.f(c)\}$$

aber wegen der Beschränkung der Eingabebereitschaft

$$F_M'^{In_1}(in.c \circ out1.c \circ in.d, In_1)$$

und auch

$$F_{S_1}^{Out_1}(out1.c, Out_1)$$

Auf der anderen Seite ist obiger Failurepaar kein mögliches Failurepaar des Systems bestehend aus den unbeschränkten Versionen, da die beiden letzteren Eigenschaften dort nicht gelten. Anschaulich ist darauf zurückzuführen, daß die Einschränkung der Eingabebereitschaften zu einer Verklemmung der Komponenten führt und so durch die Veränderung der Eingabebereitschaft der Komponenten auch die Ausgabebereitschaft des Gesamtsystems geändert wurde. ◇

Verfügt eine Verfeinerung über die Eigenschaft, daß eine Verfeinerung der Komponenten eines System auch die Verfeinerung des aus diesen Komponenten aufgebauten System bedingt, so spricht man von einer *kompositionalen* Verfeinerungsbeziehung. Tatsächlich ist diese Form der Verfeinerung für den Entwicklungsprozeß von großem Vorteil, da eine *modulare* Entwicklung vorgenommen werden kann. Dies bedeutet, daß jede Komponente ohne Rücksicht auf andere Komponenten des Systems verfeinert werden kann. Die Komposition solcher verfeinerter Komponenten führt zu einer Verfeinerung des Gesamtsystems.

Wie obiges Beispiel zeigt, genügt der Wechsel von der asynchronen zur synchronen Systemsicht mit beschränkter Eingabebereitschaft *nicht* dieser Kompositionalität. Auf der anderen Seite ist dieser Wechsel, wie eingangs erläutert, als methodischer Schritt äußerst wünschenswert. Daher stellt sich die Frage, wie trotzdem eine methodische Vorgehensweise für diesen Wechsel angegeben werden kann. Diese Vorgehensweise muß es erlauben, Systeme mit beschränkten Kapazitäten durch gezielte Beschränkung der Kapazitäten der Komponenten zu entwickeln. Gleichzeitig muß, wie es entsprechend die Kompositionalität garantiert, durch die methodische Vorgehensweise auch sichergestellt werden, daß die zuvor auf der abstrakteren Ebene behandelten Eigenschaften beibehalten werden. Eine solche methodische Vorgehensweise anzugeben ist daher das Ziel des in dieser Arbeit beschriebenen Ansatzes.

Wie in Beispiel 3.6.2 gezeigt, bleiben beim Wechsel von der asynchron zur synchron orientierten Systemsicht zwar die Eigenschaften der partiellen Abläufe erhalten. Dies gilt aber im allgemeinen nicht für die Ein- und Ausgabebereitschaft. Damit müssen diese Eigenschaften beim Wechsel der Systemsichtweise besonders behandelt werden. Entsprechend lassen

sich zwei mögliche Vorgehensweisen für die Entwicklung synchroner Systembeschreibungen beginnend mit einer asynchronen Systemsicht unterscheiden:

1. Auf der Ebene der asynchron kommunizierenden Systeme werden ausschließlich partielle Abläufe berücksichtigt. Die weiteren Eigenschaften, insbesondere die Ein- und Ausgabebereitschaft von System und Komponenten wird erst auf der Ebene der synchron kommunizierenden Systeme hinzugenommen.
2. Es werden auf der Ebene der asynchron kommunizierenden Systeme sowohl Anforderungen an die partiellen Abläufe als auch die Eigenschaften vollständiger Abläufe entwickelt. Auf der synchronen Ebene werden die Eigenschaften abgeschwächt, die die Eingabebereitschaft des Systems charakterisieren.

In der hier vorgestellten Arbeit wird der zweite Ansatz gewählt, da für diesen Ansatz mehrere Gründe sprechen:

1. Im ersten Fall unterscheidet sich die Vorgehensweise nicht von einem Entwurf, der ausschließlich auf der synchronen Ebene vorgenommen wird. Auch dort werden zuerst ausschließlich die partiellen Abläufe entwickelt, und erst anschließend Eingabe- und Ausgabebereitschaft.
2. Im Fall der asynchron kommunizierenden Systeme beschreiben partielle Abläufe, welche Eigenschaften die Eingabe aufweisen muß, falls eine bestimmte Ausgabe produziert worden ist. Damit wird auf die Möglichkeit verzichtet, schon auf der Ebene der asynchron kommunizierenden Systeme und damit ohne Rücksicht auf Frage der Synchronisierung der Ein- und Ausgaben festzulegen, welche Ausgabe zu einer bestimmten Eingabe produziert werden soll.
3. Dieser Verzicht erzwingt es, die Frage der Produktion der Ausgabe immer im Zusammenhang mit der Synchronisierung von Ein- und Ausgabe zu behandeln. Diese Vermischung der Aufgaben führt zu einer Verkomplizierung des Entwurfs. Wird die Frage der Produktion der Ausgabe schon auf der Ebene der asynchron kommunizierenden Systeme behandelt, bleibt auf der Ebene der komplizierter zu behandelnden Systeme nur noch die Behandlung der Eingabebereitschaft. Diese Trennung der Aufgaben führt zu einer Vereinfachung des Entwurfs.

Somit stellt die Asynchronität zwar nicht im formalen, aber im methodischen Sinn eine für die hier vorgestellte Vorgehensweise geeignete Abstraktion dar. Abschnitt 3.7 und Kapitel 4 beschäftigen sich daher mit den formalen und methodischen Fragen, die sich durch den Einsatz der Asynchronität als Abstraktion ergeben.

3.7 Abschwächung der Systemanforderung

Wie in Kapitel 1 besprochen, besteht die ursprüngliche Abstraktion der hier vorgestellten Vorgehensweise darin, den ersten Schritt der Systementwicklung unter der Annahme

der asynchronen Kommunikation vorzunehmen. Damit wird naturgemäß auch für die ursprüngliche Anforderungsspezifikation an das zu entwickelnde System die Anforderung gestellt, zur unbeschränkten Pufferung in der Lage zu sein. Bei der Rücknahme der Abstraktion wird jedoch im allgemeinen kein Interesse mehr an einem echt asynchron kommunizierenden System bestehen, zumal die Realisierung eines solchen Systems - also eines unbeschränkt gepufferten Systems - mit beschränkten Ressourcen nicht möglich ist. Daher ist im Falle der hier beschriebenen Vorgehensweise vor der Bestimmung der zu implementierenden Anforderungen der Systemkomponenten zuerst eine Anpassung der Anforderungen an das System insgesamt nötig.

3.7.1 Abschwächung

Der Verzicht auf die asynchron orientierte Sichtweise bringt es mit sich, von der Forderung abzuweichen, daß das System sich in allen beliebigen Umgebungen wie das ursprünglich spezifizierte, asynchron kommunizierende System verhält. Statt dessen wird es im allgemeinen ausreichend sein, daß es in einer Umgebung, die zusätzlichen Einschränkungen unterliegt, das gleiche Verhalten aufweist, wie das entsprechende asynchron kommunizierende System. Der nächste Schritt der Systementwicklung wird also in der Einschränkung bzw. *der Verschärfung des Umgebungsverhaltens* bestehen.

In der bisherigen Entwicklung wurde die Umgebung nur indirekt in der Beschreibung der Systemanforderungen berücksichtigt. Um diese Sichtweise durch den ganzen Entwicklungsprozeß durchgängig zur Verfügung zu stellen, sollte auch die Aufhebung der Abstraktion direkt durch die Anpassung der Systembeschreibung anstatt des Umgebungsverhaltens vorgenommen werden. Statt also eine Verschärfung des Umgebungsverhaltens zu beschreiben, wird hier die dazu duale Sichtweise der *Abschwächung des Systemverhaltens* vorgenommen.

Abschwächungsbegriff

Wie bereits im letzten Absatz angesprochen, stellt die Abschwächung der Anforderung an das Systemverhalten die Umkehrung der Verfeinerung der Anforderung an das Umgebungsverhalten. Im Failuremodell, wie auch in anderen Modellen zur Beschreibung verteilter Systeme (vgl. z.B. [BDD$^+$93]), wird als Verfeinerungsrelation für Spezifikationen die einfachste Relation, nämlich die Implikation verwendet. Daher wird die Abschwächung als umgekehrte Implikation definiert.

Definition 3.7.1 (Abschwächung) Seien P und P' zwei Failureeigenschaften der Art

$$P, P' : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$$

Dann heißt P' *eine Abschwächung von* P, wenn

$$\forall t \in A^\omega, R \subseteq A.P(t, R) \Rightarrow P'(t, R)$$

gilt. ∘

Damit ist die Abschwächung die Umkehrrelation zur Verfeinerung.

Methodischer Einsatz der Abschwächung

Das Konzept der allgemeinen Abschwächung definiert beliebige Abschwächungen von Failurebeschreibungen. Für den in dieser Arbeit vorgestellten Ansatz wird dieses Konzept nicht in seiner Allgemeinheit benötigt. Beim methodischen Übergang von asynchronen zu synchronen Systembeschreibungen werden die beim Wechsel von der Spur- zur Failuredarstellung gewonnenen Prozeßbeschreibungen im allgemeinen nicht beliebig abgeschwächt. Wie in für die in 3.4 beschriebenen expliziten Transformationsschemata gezeigt, ergeben sich bei der Failuredarstellung der asynchron kommunizierenden Prozesse durch die Transformation der Spurbeschreibung folgende drei Teilbeschreibungen:

- Anforderungen an die partiellen Abläufe
- Anforderungen an die Eingabebereitschaft
- Anforderungen an die Ausgabebereitschaft

Bei der Abschwächung der Systembeschreibung besteht daher die Möglichkeit, jede einzelne dieser drei Anforderungen abzuschwächen. Für die Anforderungen an die partiellen Abläufe ist dies im allgemeinen nicht erforderlich. Denn die Anforderungen an die partiellen Abläufe werden - wegen ihres Charakters als partielle Korrektheitseigenschaften von Systemen - unabhängig von der Beschreibung asynchron oder synchron kommunizierender Systeme erstellt und sollten daher auch nach der Rücknahme der Abstraktion ihre Gültigkeit behalten.

Anders verhält es sich mit den Anforderungen an die Ausgabebereitschaft. Dies ist besonders dann der Fall, wenn auf der Spurebene die Anforderungen an die Ausgabebereitschaft unter Verwendung des in Abschnitt 2.6.2 beschriebenen Schemas mit einem hohen Grad an Parallelität spezifiziert worden sind. Da diese auf Spurebene beschriebene vollständige Entkoppelung unter Umständen die unbeschränkte Eingabebereitschaft im System voraussetzt, kann es auch hier wünschenswert sein, diese Anforderungen abzuschwächen.

Im Regelfall wird auch die durch die Abstraktion bisher erhobene Forderung nach unbeschränkter Pufferung der Eingabe in dieser Form nicht erhalten bleiben. Somit wird es im allgemeinen nötig sein, neben den Anforderungen an die Ausgabebereitschaft insbesondere die Anforderungen an die Eingabebereitschaft entsprechend anzupassen.

Die Abschwächung der Anforderungen an die Ein- oder Ausgabebereitschaft wird vorgenommen, indem Einschränkungen an die Umgebung vorgenommen werden. Die Ein- bzw. Ausgabebereitschaft auf wird also nur noch auf einem Ausschnitt aller bisher möglichen Abläufe gefordert. Dies kann schematisch geschehen, indem eine Einschränkung $U(t)$ des Umgebungsverhaltens $U(t)$ festgelegt wird. Dann wird die Eingabebereitschaft $F^I(t,R)$ bzw. Ausgabebereitschaft $F^O(t,R)$ durch

$$F'^I(t,R) \stackrel{\text{def}}{=} U(t) \Rightarrow F^I(t,R)$$

beziehungsweise

$$F'^O(t,R) \stackrel{\text{def}}{=} U(t) \Rightarrow F^O(t,R)$$

ersetzt. Daß diese modifizierten Anforderungen auch Abschwächungen im Sinne von Definition 3.7.1 darstellen, folgt unmittelbar.

3.7.2 Beispiel

Am folgenden Beispiel wird die Abschwächung der Systemanforderungen im Entwicklungsprozeß demonstriert. Dazu wird wiederum der oben eingeführte Puffer verwendet. Die Spezifikation $B(t, R)$ beschreibt einen Puffer mit unbeschränkter Kapazität:

$$\begin{array}{lcl} B(t,R) & \stackrel{\text{def}}{=} & S(t) \wedge F^I(t,R) \wedge F^O(t,R) \\ S(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.\, \mathsf{data}^*(O©s) \sqsubseteq \mathsf{data}^*(I©s) \\ F^I(t,R) & \stackrel{\text{def}}{=} & \forall i \in I.i©t \in I^* \Rightarrow i \notin R \\ F^O(t,R) & \stackrel{\text{def}}{=} & \mathsf{data}^*(O©t) \sqsubset \mathsf{data}^*(I©t) \Rightarrow \\ & & \quad \exists o \in O.(o \notin R \wedge \mathsf{data}^*((O©t) \circ o) \sqsubseteq \mathsf{data}^*(I©t)) \end{array}$$

Im Gegensatz zu $B(t, R)$ wird mittels $B'(t, R)$ ein Prozeß beschrieben, der einen Puffer der Kapazität Eins darstellt:

$$\begin{array}{lcl} B'(t,R) & \stackrel{\text{def}}{=} & S(t) \wedge F'^I(t,R) \wedge F^O(t,R) \\ F'^I(t,R) & \stackrel{\text{def}}{=} & (I©t \in I^* \wedge O©t = I©t) \Rightarrow I \cap R = \emptyset \end{array}$$

Es zeigt sich leicht, daß $B'(t, R)$ tatsächlich eine Abschwächung von $B(t, R)$ im Sinne von 3.7.1 darstellt. Da ersteres durch Ersetzen von $F^I(t, R)$ durch $F'^I(t, R)$ in letzterem entsteht, folgt aus der Tatsache

$$F^I(t,R) \Rightarrow F'^I(t,R)$$

sofort die Behauptung. Wie oben besprochen, wurde hier in der Tat lediglich die Anforderung an die Eingabebereitschaft $F^I(t, R)$ abgeschwächt, während die Anforderungen an die partiellen Abläufe $S(t)$ und die Ausgabebereitschaft $F^O(t, R)$ unverändert blieben. Die zugehörigen Anforderung an die Umgebung $U(t)$ ist dabei

$$U(t) \stackrel{\text{def}}{=} O©t = I©t$$

3.8 Verschärfung der Anforderungen

In Abschnitt 3.7 wurde gezeigt, wie die Abschwächung von Systemanforderungen eingesetzt werden kann, um den Abstraktionsschritt zurückzunehmen, der durch die asynchrone Kommunikation eingeführt wurde. Diese Abschwächung wird typischerweise beim Wechsel

von der asynchronen zur synchronen Sichtweise zur Anpassung der Anforderungen eingesetzt. Dabei werden die Anforderungen an die Eingabebereitschaft – und unter Umständen auch an die Ausgabebereitschaft – des zu entwickelnden Systems angepaßt.

Alternativ zur Anpassung des Kontrollflusses durch die Änderung des Eingabeverhaltens von Komponenten kann der Kontrollfluß auch durch Änderungen des Ausgabeverhaltens angepaßt werden. Dazu wird im nächsten Abschnitt anhand eines Beispiels gezeigt, warum und in welcher Form diese zusätzlichen Anforderungen auf der Ebene der synchronen Kommunikation zum Tragen kommen. Im folgenden Abschnitt wird dann gezeigt, wie diese zusätzlichen Anforderungen in den Entwurfsprozeß eingebettet werden können.

3.8.1 Nichtblockierendes Senden

Bei der Beschreibung asynchroner Systeme spielt die Überprüfung der Empfangsbereitschaft von Systemkomponenten keine wesentliche Rolle. Bei solchen Systemen soll gerade von solchen Eigenschaften abstrahiert werden. Damit ist das Versenden einer Nachricht immer möglich, und muß nicht von der Annahmebereitschaft des Empfänger anhängig gemacht werden.

Im Gegensatz dazu spielt bei der Behandlung synchron kommunizierender Systeme die Frage der Eingabebereitschaft eine wesentliche Rolle. Insbesondere kann es dabei wünschenswert sein, Ausgaben davon abhängig zu machen, ob der Empfänger zur Annahme der Nachricht bereit ist. Im folgenden Beispiel wird dazu das unterschiedliche Verhalten einer Verteilerkomponente in ihrer Darstellung als asynchron bzw. synchron kommunizierende Komponente gezeigt.

Beispiel 3.8.1 (Verteiler) Ein Verteiler Dist mit Eingabekanal In und Ausgabekanälen Int1 und Int2 leitet auf dem Eingabekanal empfangene Daten über die Ausgabedaten an die verarbeitenden Einheiten Pro1 und Pro2 weiter. Im asynchronen Fall läßt sich das Verhalten des Verteilers mit den Aktionsmengen In, $Int1$ und $Int2$ definiert wie in Beispiel 2.2.4 beschreiben als

$$\begin{aligned} T_{Dist}(t) &\stackrel{\text{def}}{=} P_{Dist}(t) \wedge C_{Dist}(t) \\ P_{Dist}(t) &\stackrel{\text{def}}{=} \forall s \sqsubseteq t.\, \mathsf{data}^*((Int_1 \cup Int_2)©s) \sqsubseteq \mathsf{data}^*(In©s) \\ C_{Dist}(t) &\stackrel{\text{def}}{=} \mathsf{data}^*((Int_1 \cup Int_2)©t) = \mathsf{data}^*(In©t) \end{aligned}$$

wobei data^* entsprechend wie in Beispiel 2.2.4 definiert ist.

Da - wie zu Beginn der Arbeit erläutert - im asynchronen Modell nur der Datenfluß, nicht aber der durch Synchronisierung hervorgerufene Kontrollfluß modelliert werden soll, spielt hier die Frage keine Rolle, ob die verarbeitenden Einheit bereits beschäftigt ist, an die die nächste Nachricht zugestellt wird. Dies spielt bei dieser Sichtweise auch keine wesentliche Rolle, da hier jede Komponente bereits ist, weitere Nachrichten entgegenzunehmen. Die Zuteilung der nächsten Nachricht kann *erratisch*, also ohne zusätzliche Information nichtdeterministisch erfolgen. Falls die Zuteilung an ausschließlich freie verarbeitende

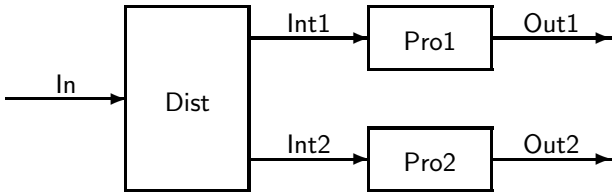

Abbildung 3.2: Das Verteilersystem

Einheiten erfolgen, so muß ein explizites Protokoll zur Flußkontrolle mit Rückmeldungen von den Einheiten an den Verteiler eingeführt werden.

Im Fall der synchronen Kommunikation spielt der durch die Synchronisierung hervorgerufene Kontrollfluß eine wesentliche Rolle. Ist hier eine der verarbeitenden Einheiten belegt, so wird eine stärkere Eingabebereitschaft benötigt, falls die Nachricht dieser Komponente zugeteilt wird anstelle der freien verarbeitenden Einheit. Falls auch hier eine erratische Zuteilung erfolgen kann, so reicht es,

$$
\begin{array}{lcl}
F^{O}(t,R) & \stackrel{\text{def}}{=} & F^{Int_1}(t,R) \vee F^{Int_2}(t,R) \\
F^{Int_1}(t,R) & \stackrel{\text{def}}{=} & \mathsf{data}^*((Int_1 \cup Int_2)©t) \sqsubset \mathsf{data}^*(In©t) \Rightarrow \\
 & & \quad \exists o \in Int_1.(o \notin R \wedge \mathsf{data}^*(((Int_1 \cup Int_2)©t) \circ o) \sqsubseteq \mathsf{data}^*(In©t)) \\
F^{Int_2}(t,R) & \stackrel{\text{def}}{=} & \mathsf{data}^*((Int_1 \cup Int_2)©t) \sqsubset \mathsf{data}^*(In©t) \Rightarrow \\
 & & \quad \exists o \in Int_2.(o \notin R \wedge \mathsf{data}^*(((Int_1 \cup Int_2)©t) \circ o) \sqsubseteq \mathsf{data}^*(In©t))
\end{array}
$$

zur Beschreibung der Ausgabebereitschaft zu verwenden. Falls jedoch eine angelische Zuteilung, also in Abhängigkeit der Bereitschaft der empfangenden Einheit erfolgen soll, so muß statt dessen die Anforderung

$$F^{O}(t,R) \stackrel{\text{def}}{=} F^{Int_1}(t,R) \wedge F^{Int_2}(t,R)$$

verwendet werden. Damit wird sichergestellt, daß die nächste anstehende Nachricht beiden verarbeitenden Einheiten angeboten wird. Ist mindestens eine davon zur Entgegennahme bereit, wird sie einer beliebigen freien Einheit zugeteilt. ◇

Anhand des Beispiels wird klar, daß die Anpassung des Ausgabeverhaltens von Komponenten auch Einfluß auf das Eingabeverhalten des Gesamtsystems haben kann. Wird die angelische Zuteilung verwendet, so hat das Gesamtsystem als Kapazität die Summe der Einzelkapazitäten der Komponenten. Hat also jede verarbeitende Komponente sowie der Verteiler die Kapazität *Eins*, so hat damit das Gesamtsystem die Kapazität *Drei*. Anderenfalls ist die garantierte Gesamtkapazität lediglich nur die Summe der Kapazität des Verteilers sowie der verarbeitenden Einheit mit der geringsten Kapazität. Für die erratische Zuteilung ergibt sich im obigen Fall die Kapazität *Zwei*.

Durch die in den Abschnitten 3.2 und 3.4 beschriebenen Transformationen wird auf der Ebene der Failurebeschreibung stets eine erratische Zuteilung von Nachrichten als Umsetzung für eine nichtdeterministische Spurbeschreibung verwendet. Alternativ ließe sich

auch eine auf der angelischen Zuteilungsstrategie basierende Transformation definieren.[5] In asynchronen Systemen ist jedoch das Auftreten von angelischem Nichtdeterminismus hinsichtlich der Zuteilung der Ausgabe nicht von erratischem Nichtdeterminismus zu unterscheiden (vgl. [dBKPR91]). Daher wird bei der Spurspezifikation der Nichtdeterminismus hinsichtlich der Ausgabe im allgemeinen als Unterspezifikation der Beschreibung und damit erratisch interpretiert. Für den hier gewählten Ansatz, bei dem die asynchrone Sichtweise als Abstraktion der synchronen Implementierungssicht gesehen wird, ist deshalb die Forderungen nach einer generellen angelischen Realisierung des auftretenden Nichtdeterminismus methodisch nicht gerechtfertigt. Daher wird in dem in dieser Arbeit vorgestellten Ansatz die hier angemessenere erratische Zuteilung verwendet. Falls eine angelische Auswahl benötigt wird, kann diese explizit als Verfeinerungsschritt nach dem Wechsel von der Spur- zur Failuredarstellung eingeführt werden.

3.8.2 Allgemeine Verfeinerungsschritte

In Abschnitt 3.8.1 wurde gezeigt, wie eine besondere Art der Verfeinerung nach dem Wechsel von der asynchronen zur synchronen Sichtweise im Entwicklungsprozeß eingesetzt werden kann. Darüberhinaus kann natürlich die Verfeinerung auch allgemein an dieser Stelle zur Elimination der Unterspezifikation des *Ausgabeverhaltens* eingesetzt werden. Diese Verfeinerung ist auch kompositional, und kann jederzeit *vor* oder *nach* dem Wechsel von der asynchronen zur synchronen Sichtweise durchgeführt werden.

3.9 Zusammenfassung

Für die in dieser Arbeit beschriebene systematische Entwicklung synchron kommunizierender Systeme spielt der Wechsel von der asynchron orientierten zur synchron orientierten Systemsicht eine entscheidende Rolle. In diesem Kapitel wurde gezeigt, wie im Zuge dieses Wechsels aus den Spuranforderungen der asynchronen Systeme systematisch entsprechende Failureanforderungen entwickelt werden können, die als Ausgangspunkt für die weitere Herleitung dienen können. Dabei wurde besonders der für die Vorgehensweise wesentliche Fall sequentieller Prozesse behandelt. Weiterhin wurde erläutert, warum die hier vorgeschlagene Vorgehensweise vorteilhaft gegenüber einem Ansatz ist, der nur die partiellen Eigenschaften der asynchronen Systeme berücksichtigt. Abschließend wurde demonstriert, wie Verhaltensaspekte, die erst auf der Ebene der synchron kommunizierenden Systeme eine Rolle spielen, systematisch in den Entwicklungsprozeß integriert werden können.

[5] In dem in Abschnitt 1.2.4 erwähnten Ansatz von [Old91] wird eine angelische Zuteilung bei der Umsetzung von Spur- in Readinessanforderungen verwendet.

Kapitel 4

Effiziente Realisierung

I hold the opinion that the construction of computer programs is a mathematical activity like the solution of differential equations, that programs can be derived from their specifications through mathematical insight, calculation, and proof, using algebraic laws as simple and elegant as those of elementary arithmetic. Such methods of program construction promise benefits in specifications, systems software, safety-critical programs, silicon design, and standards.

C.A.R. Hoare, [Hoa93b]

Gemäß der in den vorangegangenen Kapiteln bisher beschriebenen Vorgehensweise werden bei der Systementwicklung die folgenden Schritte durchlaufen:

- Im ersten Schritt wird die Asynchronität eingesetzt, um mit dieser, gegenüber einer synchronen Modellierung abstrakteren Systemsicht eine einfacheren Entwicklung der System- und Komponentenbeschreibungen zu erlauben. Dabei wird in diesem Schritt die Implementierung der Anforderungen an das System durch die Anforderungen durch an die Komponenten sichergestellt.
- Im zweiten, in Kapitel 3 beschriebenen Schritt werden diese Anforderungen an das System und seine Komponenten von der Spurdarstellung und der damit verbundenen asynchronen Sichtweise in eine entsprechende Failuredarstellung übertragen. Mit der Übertragung der Darstellung von System und Komponenten wird auch die Implementierungsbeziehung zwischen diesen übertragen. Dabei werden die Spurdarstellungen so übertragen, daß auch die Failuredarstellungen von System und Komponenten asynchrones Verhalten modellieren.

- Im dritten, ebenfalls in Kapitel 3 beschriebenen Schritt wird die durch die Übertragung entstandene Failurebeschreibung des Systems abgeändert, um das durch die Umsetzung implizierte asynchrone Verhalten abzuschwächen.

Mit der Ausführung des zweiten Schritts liegen nun die Anforderungen an die Komponenten in Failuredarstellung, aber noch immer mit asynchronem Verhalten vor. Durch die Ausführung des dritten Schritts wurde nur die Systembeschreibung zu einem abgeschwächtem synchronem Verhalten überführt. Das Ziel der Vorgehensweise ist jedoch eine effiziente Realisierung des Systems und damit auch die synchrone Realisierung der Komponenten. Insbesondere sind dazu die Anforderungen an die Eingabebereitschaft der Komponenten abzuschwächen. Es stellt sich daher die Frage, wie diese, an die Komponenten eines Systems gestellten *minimalen* Anforderungen, die oft auch als "Residuen" (vgl. z.B. [Hoa94]) bezeichnet werden, auf möglichst gezielte Art und Weise bestimmt werden können.

Damit ergeben sich die folgenden Schwerpunkte dieses Kapitels:

Einführung des Residuenbegriffs: Zunächst wird in Abschnitt 4.1 der Begriff des "Residuums" allgemein eingeführt und dann in Abschnitt 4.2 für den Spezialfall der Entwicklung verteilter Systeme formuliert. Schließlich wird in 4.2.3 gezeigt, warum die endliche Failuresemantik für die Residuenbestimmung nicht geeignet ist.

Methodischer Einsatz des Residuums: In Abschnitt 4.3 werden dann die eingeführten Residuen für die Bestimmung minimaler Anforderungen an die Systemkomponenten eingesetzt. Dazu wird ein schrittweises Bestimmungsverfahren eingeführt, das sich am strukturellen Aufbau des Systems orientiert.

Bestimmung von Residuen: In Abschnitt 4.4 wird schließlich eine direkte Darstellung der Residuen für die Abstraktion und die Parallelkomposition angegeben, die die eingeführten Eigenschaften aufweist.

Für die letztendliche programmiersprachliche Realisierung der entwickelten minimalen Anforderungen an die Systemkomponenten, beispielsweise mit den in in Abschnitt 1.1 angesprochenen Ansätzen, werden jedoch kanalorientierte Beschreibungen der Komponenten benötigt. Daher wird in Abschnitt 4.5 gezeigt, wie aufbauend auf diesen Anforderungen für die Realisierung geeignete Kanäle identifiziert werden können.

Insgesamt werden die eingeführten Begriffe und Techniken an einzelnen kleineren Beispielen, sowie in Abschnitt 4.6 durchgehend von der Spurdarstellung bis hin zur kanalorientierten Realisierung an einem Beispiel demonstriert.

4.1 Residuen

Das wesentliche Prinzip des Systementwurfs ist, wie oben besprochen, die Modularität. Das System wird bei dieser Vorgehensweise mittels unterschiedlicher Kompositionsoperatoren aus Komponenten aufgebaut. Bei der Verwendung formaler Beschreibungen bedeutet

dies, daß aus den Anforderungen an die Komponenten die Anforderungen an das Systems abgeleitet werden können. Die Verwendung dieser Beschreibungen erlaubt aber auch den umgekehrten Weg. Hierbei werden aus den Anforderungen an das System und einige seiner Komponenten Anforderungen an den Systemrest abgeleitet:

> Ist eine Anforderung R an das Gesamtsystem gegeben und soll dieses mittels eines Kompositionsoperators $\bullet$ aus der gegebenen Komponente mit der Spezifikation Q und einer weiteren, zu bestimmenden Komponente mit der Eigenschaft P konstruiert werden, wie muß dann P beschaffen sein, daß
>
> $$P \bullet Q \Rightarrow R$$
>
> gilt?

Hier sind beliebige Kompositionsoperatoren möglich. Für diese Arbeit sind insbesondere die Parallelkomposition und die Abstraktion von Bedeutung, aber auch die einfache Konjunktion - wie in Abschnitt 4.3.2 gezeigt - wird verwendet. Das folgende Beispiel zeigt die Bestimmung der Anforderungen für den Fall der sehr einfache Komposition mittels der Konjunktion.

Beispiel 4.1.1 (Komposition mittels $\wedge$) Im folgenden wird, für beliebige spezifizierte Objekte $s \in S$, der allgemeine Fall der Bestimmung einer hinreichenden Eigenschaft $P : \mathcal{S} \to \mathbb{B}$ zur Realisierung von $R : \mathcal{S} \to \mathbb{B}$ mittels der Konjunktion von $Q : \mathcal{S} \to \mathbb{B}$ durch $\wedge$ betrachtet. Gesucht ist ein P mit

$$\forall s \in \mathcal{S}.P(s) \wedge Q(s) \Rightarrow R(s) \tag{4.1}$$

Eine mögliche Lösung dafür ist jedes P mit

$$\forall s \in \mathcal{S}.P(s) \Rightarrow P'(s) \tag{4.2}$$

wobei

$$P'(s) \Leftrightarrow (Q(s) \Rightarrow R(s)) \tag{4.3}$$

Daß jedes solche P tatsächlich die Anforderung 4.1 erfüllt, zeigt

$$\begin{array}{ll} & P(s) \wedge Q(s) \\ \Longrightarrow & [4.2] \\ & P'(s) \wedge Q(s) \\ \Longleftrightarrow & [4.3] \\ & (Q(s) \Rightarrow R(s)) \wedge Q(s) \\ \Longrightarrow & [\text{Aussagenlogik}] \\ & R(s) \end{array}$$

$\diamond$

Bei der Bestimmung des zu implementierenden Systemrests P ist im allgemeinen keine beliebige hinreichende Lösung gesucht. Der so eingeschränkte Systemrest kann strenge Anforderungen aufweisen, die eine Implementierung dieses Rests schwierig oder sogar unmöglich

machen. Werden nämlich beliebige hinreichende Anforderungen an diesen Systemrest zugelassen, so ist im obigen Beispiel auch immer *False* eine Lösung. Eine solche Anforderung kann aber nicht realisiert werden.

Die eigentlich gesuchte Anforderung an den zu bestimmenden Systemrest ist die *einfachste* zu realisierende Anforderung, also mit anderen Worten die Anforderung, die hinsichtlich aller möglichen Realisierungen die liberalste ist. Deshalb muß die obige Fragestellung umformuliert werden zu:

> Ist eine Anforderung R an das Gesamtsystem gegeben und soll dieses mittels eines Kompositionsoperators $\bullet$ aus der gegebenen Komponente mit der Spezifikation Q und einer weiteren, zu bestimmenden Komponente mit der Eigenschaft P konstruiert werden, wie muß dann P beschaffen sein, daß
>
> $$P \bullet Q \Rightarrow R$$
>
> und
>
> $$\neg P \bullet Q \Rightarrow \neg R$$
>
> gelten?

Gesucht wird also nicht nur ein Systemrest, dessen Anforderung hinsichtlich der Komposition $\bullet$ mit Q *hinreichend* für die Realisierung der Gesamtanforderung R ist, sondern für diese auch *notwendig* ist. Diese *hinreichende* und *notwendige* Eigenschaft wird der *minimale zu realisierende Systemrest* oder kurz *Residuum*[1] genannt.

Das folgende Beispiel zeigt das Residuum für den Fall der Konjunktion als Komposition.

Beispiel 4.1.2 (Residuum hinsichtlich $\wedge$) Analog zum Beispiel 4.1.1 wird nun der allgemeine Fall der Bestimmung des Residuums $P : \mathcal{S} \to \mathbb{B}$ zur Realisierung von $R : \mathcal{S} \to \mathbb{B}$ mittels der Konjunktion von $Q : \mathcal{S} \to \mathbb{B}$ durch $\wedge$ betrachtet. Gesucht ist ein P mit

$$\forall s \in \mathcal{S}.P(s) \wedge Q(s) \Rightarrow R(s) \tag{4.4}$$

und

$$\forall s \in \mathcal{S}.\neg P(s) \wedge Q(s) \Rightarrow \neg R(s) \tag{4.5}$$

Das entsprechende Residuum P ergibt sich damit zu

$$P(s) \Leftrightarrow (Q(s) \Rightarrow R(s)) \tag{4.6}$$

Der Nachweis, daß P wirklich die Anforderung 4.4 erfüllt, wurde bereits in Beispiel 4.1.1 geführt. Den Nachweis von 4.5 zeigt

$$\begin{array}{ll} & \neg P(s) \wedge Q(s) \\ \Longleftrightarrow & [4.6] \\ & \neg(Q(s) \Rightarrow R(s)) \wedge Q(s) \end{array}$$

[1] **residuum** *lat.* Rest

$\Longrightarrow$ [Aussagenlogik]
$Q(s) \wedge \neg R(s) \wedge Q(s)$
$\Longrightarrow$ [Aussagenlogik]
$\neg R(s)$

$\diamond$

Damit ist das Residuum für die Konjunktion bestimmt. In Abschnitt 4.3.2 wird gezeigt, wie das Residuum der Konjunktion in der in dieser Arbeit vorgestellten Vorgehensweise eingesetzt wird. Weiterhin müssen offensichtlich die Residuen für die Operatoren zur Systemkonstruktion, also die Parallelkomposition und die Abstraktion, definiert werden.

4.2 Residuen verteilter Systeme

Wie zu Beginn des Kapitels beschrieben, besteht der nächste Schritt der in dieser Arbeit vorgestellten Vorgehensweise in der Abschwächung der Anforderungen an die Komponenten eines Systems. Durch die bisherigen Entwicklungsschritte wurden Anforderungen an die Komponenten entwickelt, die die Komponenten als asynchrone Prozesse beschreiben. Das Ziel der Entwicklung ist aber - wie in Abschnitt 1.2.2 erläutert - eine speichereffiziente synchrone Realisierung und damit eine Abschwächung der Eingabebereitschaft der Komponenten.

Diese Abschwächung soll dabei zwei Eigenschaften erfüllen:

- Die abgeschwächte Form der Eingabebereitschaft der Komponenten zusammen mit den Anforderungen an die partiellen Abläufe und der Ausgabebereitschaft der Komponenten muß auch weiterhin die Anforderungen Ein- und Ausgabebereitschaft des Gesamtsystems garantieren.
- Die Anforderungen an die Eingabebereitschaft der Komponenten soll dabei so schwach wie möglich sein, um so die maximale Speichereffizienz zu garantieren.

Dies bedeutet, daß die in Abschnitt 4.1 beschriebenen *hinreichenden* und *notwendigen* Anforderungen an die einzelnen Komponenten gesucht sind. Dazu muß der bisher allgemein eingeführte Residuenbegriff auf die beiden Operatoren zur Systemkonstruktion, die Parallelkomposition und die Abstraktion, angewendet werden.

4.2.1 Residuen im Entwicklungsprozeß

Wie zuvor beschrieben erlaubt die Residuenbestimmung eine *zielgerichtete* Herleitung von minimalen Anforderungen an einzelne Komponenten eines Systems, um dessen Gesamtverhalten sicherzustellen. Damit stellt sich die Frage, warum Residuen erst bei der Bestimmung der minimalen Ein- bzw. Ausgabebereitschaft eingesetzt werden, nicht jedoch vorher im Entwicklungsprozeß.

Die Antwort darauf liegt in der Tatsache, daß durch die Residuenbestimmung nicht immer implementierbare minimale Anforderungen bestimmt werden können. Dies ist besonders bei der Residuenbestimmung von mehrstelligen Operatoren, wie beispielweise der zweistelligen Parallelkomposition[2] von Bedeutung. Hier wird die Residuenbestimmung eingesetzt, um die minimalen Anforderungen an eine Komponente S_2 zu bestimmen, die ausreichen, um zusammen mit einer weiteren Komponente S_1 die Anforderungen an das System $S_1 \parallel S_2$ zu implementieren. Dabei ist es jedoch möglich, daß keine Anforderung an S_2 ausreicht, um die Anforderungen an $S_1 \parallel S_2$ zu implementieren. In diesem Fall wird die triviale minimale Anforderung $\mathbf{F}$ für S_2 ermittelt. Dann muß eine andere Komponente S_1' gewählt werden, um die Anforderungen an das System zu implementieren. Damit ist keine *zielgerichtete* Vorgehensweise möglich.

Hier kommt nun die in den vorherigen Kapiteln beschriebene Vorgehensweise zum Tragen:

- Wie in Abschnitt 2.6 beschrieben, wird auf der Spurebene ausgehend von den Anforderungen an das zu realisierende System eine Implementierung des Systems durch Systemkomponenten entwickelt. Dabei wird die Implementierungsbeziehung zwischen den Anforderungen an das System sowie den Anforderungen an die Komponenten sichergestellt. Die Anforderungen an die Komponenten sind also *hinreichende* Anforderungen zur Implementierung der Systemanforderungen. Für eine sinnvolle Weiterentwicklung sollte bereits auf der Spurebene sichergestellt werden, daß keine trivialen, also nicht realisierbaren Anforderungen verwendet werden.
- Im zweiten Schritt, dem in Kapitel 3 beschriebenen Modellwechsel wird nun die Beschreibung der Anforderungen an System und Komponenten von der Spur- auf die Failuredarstellung übertragen. Mit der Übertragung der Eigenschaften wird dabei auch die Implementierungsbeziehung zwischen System- und Komponentenanforderungen übertragen, wie sie auf der Spurebene nachgewiesen wurde. Damit implementieren die Failureanforderungen der Komponenten die des Systems.
- Schließlich werden, wie in Abschnitt 3.7 beschrieben, die Anforderungen an das System abgeschwächt. Wie in diesem Abschnitt gezeigt wurde, stellt die Abschwächung die Umkehrung der Implementierungsbeziehung dar. Damit folgt aus der Implementierung der ursprünglichen Anforderungen an das System durch die Anforderungen an die Komponenten auch die Implementierung der abgeschwächten Anforderungen durch die Komponentenanforderungen.

Wie bereits auf der Spurebene festgestellt, sind damit die Komponentenanforderungen *hinreichende* Anforderungen zur Implementierung der abgeschwächten Systemanforderungen.

Im letzten Entwicklungsschritt der hier beschriebenen Vorgehensweise werden nun die hinreichenden Anforderungen der Komponenten abgeschwächt zu *hinreichenden* und *notwendigen* Anforderungen zur Implementierung der bereits abgeschwächten Systemanforderungen. Da für die Komponenten bereits *hinreichende* Anforderungen vorliegen, werden durch

[2]Die Abstraktion wird in diesem Zusammenhang als einstellig betrachtet, da dieser Operator nur einen Prozeß als Argument hat, während das zweite Argument ein Teilalphabet ist.

die Residuenbestimmung - wie die Sätze 4.2.1 und 4.2.2 feststellen - ebenfalls nichttriviale *hinreichende* und *notwendige* Anforderungen an die Komponenten bestimmt, falls die bisher vorliegenden hinreichenden Anforderungen nichttrival sind.

Diese Betrachtung macht deutlich, daß die Abspaltung der Kontrollflußaspekte von der Entwicklung der Datenflußaspekte sinnvoll ist:

- Im Fall der *Datenflußaspekte* ist stets die Innovativität des Entwicklers mit der Möglichkeit des wiederholten Durchlaufens eines "trial and error"-Prozesses gefordert, um eine Implementierung der Systemanforderungen durch die Komponentenanforderungen sicherzustellen.
- Im Fall der *Kontrollflußaspekte* kann eine schematische Vorgehensweise eingesetzt werden, um für die Komponentenanforderung speichereffizienten Realisierungen zu ermitteln.

Diese Trennung der Anforderungen an die einzelnen Entwicklungsphasen unterstützt damit eine möglichst zielgerichtete Entwicklung *minimaler* synchron kommunizierender Systeme.

4.2.2 Residuen und Kompositionsoperatoren

Da die hier beschriebenen verteilten Systeme mittels der beiden Operatoren zur Abstraktion und Parallelkomposition aus einfacheren Einheiten aufgebaut werden, muß für jeden der beiden Operatoren die Bestimmung des Residuums hinsichtlich dieses Operators angegeben werden. Dazu werden die in Abschnitt 4.1 eingeführten Anforderungen für beliebige einstellige Operationen auf die beiden Operatoren angewendet.

Zur besseren Lesbarkeit wird für die Darstellung der Anforderungen an die Residuen die Notation

$$\frac{P}{Q}$$

als alternative Schreibweise für die Aussage

$$P \Rightarrow Q$$

verwendet.[3]

Residuum bei Parallelkomposition

In Abschnitt 4.1 wurde beschrieben, wie sich die Begriffe *hinreichend*, *notwendig* und *Residuum* für beliebige zweistellige Operationen $\bullet$ definieren lassen. Um entsprechende Begriffe für die Parallelkomposition zu erhalten, muß diese Definition entsprechend instantiiert werden, nämlich

[3]Die Notation $\frac{P}{Q}$ beschreibt einen prädikatenlogischen Ausdruck und ist *nicht* - wie in Abschnitt 2.4 - als Kalkülregel zu verstehen.

- die zu bestimmende Eigenschaft P mit P_2,
- die Operation $\bullet$ mit $\|$ und
- die zu erzielende Eigenschaft R durch P.

Damit ergeben sich die entsprechende Begriffe für die Parallelkomposition sofort.

Definition 4.2.1 (Hinreichend hinsichtlich Parallelkomposition) Es seien P, P_1 und P_2 Failureprädikate der Form

- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$
- $P_1 : A_1^\omega \times \mathbb{P}(A_1) \to \mathbb{B}$
- $P_2 : A_2^\omega \times \mathbb{P}(A_2) \to \mathbb{B}$

wobei $A = A_1 \cup A_2$. Dann heißt P_2 *hinreichende Anforderung für P hinsichtlich der Parallelkomposition mit P_1*, wenn

$$\frac{}{\forall t \in A^\omega, R_1 \subseteq A_1, R_2 \subseteq A_2 . P_1(A_1 \copyright t, R_1) \wedge P_2(A_2 \copyright t, R_2) \Rightarrow P(t, R_1 \cup R_2)}$$

gilt. ∘

Definition 4.2.2 (Notwendig hinsichtlich Parallelkomposition) Es seien P_1, P_2 sowie P Failureprädikate der Form

- $P_1 : A_1^\omega \times \mathbb{P}(A_1) \to \mathbb{B}$
- $P_2 : A_2^\omega \times \mathbb{P}(A_2) \to \mathbb{B}$
- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$

wobei $A = A_1 \cup A_2$. Dann heißt P_2 *notwendige Anforderung für P hinsichtlich der Parallelkomposition mit P_1*, wenn

$$\frac{\forall t \in A^\omega, R_1 \subseteq A_1, R_2 \subseteq A_2 . P_1(A_1 \copyright t, R_1) \wedge P_2'(A_2 \copyright t, R_2) \Rightarrow P(t, R_1 \cup R_2)}{\forall t \in A_2^\omega, R \subseteq A_2 . P_2'(t, R) \Rightarrow P_2(t, R)}$$

gilt. ∘

Definition 4.2.3 (Residuum hinsichtlich Parallelkomposition) Es seien P_1, P_2 sowie P Failureprädikate der Form

- $P_1 : A_1^\omega \times \mathbb{P}(A_1) \to \mathbb{B}$
- $P_2 : A_2^\omega \times \mathbb{P}(A_2) \to \mathbb{B}$
- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$

wobei $A = A_1 \cup A_2$. Dann heißt P_2 *Residuum für P hinsichtlich der Parallelkomposition mit P_1* wenn es eine hinreichende und notwendige Anforderung für P hinsichtlich der Parallelkomposition mit P_1 darstellt. ∘

Ausgehend von der Definition des Residuums für die Parallelkomposition auf der Ebene der Failureprädikate stellt sich naturgemäß die Frage, welche Bedeutung die Bestimmung eines Residuums auf der Ebene der Spezifikationen hat. Der folgende Satz verdeutlicht diese Beziehung innerhalb des sat-Kalküls.

Satz 4.2.1 (Eigenschaften des Residuums) Seien P_1,P_2 und P Failureprädikate der Art

- $P_1 : A_1^\omega \times \mathbb{P}(A_1) \to \mathbb{B}$
- $P_2 : A_2^\omega \times \mathbb{P}(A_2) \to \mathbb{B}$
- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$

wobei $A = A_1 \cup A_2$, und sei P_2 das Residuum von P hinsichtlich der Parallelkomposition mit P_1. Dann gilt

$$\frac{P_1 \text{ sat } P_1(t,R) \quad P_2 \text{ sat } P_2(t,R)}{P_1 \parallel P_2 \text{ sat } P(t,R)}$$

•

Beweis 4.2.1 (Satz 4.2.1) Folgt unmittelbar aus den Definitionen 4.2.3 sowie 3.1.4 und 3.1.3. □

Residuum für Abstraktion

Wie bereits bei der Parallelkomposition werden auch für die Definition des Begriffs des Residuums hinsichtlich der Abstraktion lediglich die Begriffe aus Abschnitt 4.1 passend instantiiert. Dabei wird lediglich die zweistellig Operation • durch die einstellig Operation $\backslash A'$ ersetzt, nämlich:

- die zu bestimmtende Eigenschaft P durch Q,
- die Operation • durch $\backslash A'$ und
- die zu erzielende Eigenschaft R durch P.

Damit lassen sich nun die gewünschten Begriffe definieren.

Definition 4.2.4 (Hinreichend hinsichtlich Abstraktion) Seien P und Q Failureprädikate der Form

- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$
- $Q : (A\backslash A')^\omega \times \mathbb{P}(A\backslash A') \to \mathbb{B}$

Dann heißt Q eine *hinreichende Anforderung für P hinsichtlich der Abstraktion von A'*, wenn

$$\forall t \in A^\omega, R \subseteq A . Q(t, R \cup A') \Rightarrow P((A\backslash A')©t, R\backslash A')$$

gilt. ○

Definition 4.2.5 (Notwendig hinsichtlich Abstraktion) Es seien P und Q Failureprädikate der Form

- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$
- $Q : (A \backslash A')^\omega \times \mathbb{P}(A \backslash A') \to \mathbb{B}$

Dann heißt Q eine *notwendige Anforderung für P hinsichtlich der Abstraktion von A'*, wenn

$$\frac{\forall t \in A^\omega, R \subseteq A.Q'(t,R) \Rightarrow P((A \backslash A') \copyright t, R \backslash A')}{\forall t \in A^\omega, R \subseteq A.Q'(t,R) \Rightarrow Q(t,R)}$$

gilt. ∘

Definition 4.2.6 (Residuum hinsichtlich Abstraktion) Seien P und Q Failureprädikate der Form

- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$
- $Q : (A \backslash A')^\omega \times \mathbb{P}(A \backslash A') \to \mathbb{B}$

Dann heißt Q *Residuum für P hinsichtlich der Abstraktion von A'*, wenn es eine hinreichende und notwendige Anforderung für P hinsichtlich der Abstraktion von A' darstellt. ∘

Wie bereits bei der Parallelkomposition soll auch hier die Beziehung der Bestimmung des Residuums zur Spezifikation auf der Ebene der Failureprädikate deutlich gemacht werden. Der folgende Satz gibt die entsprechenden Resultate für den sat-Kalkül wieder.

Satz 4.2.2 (Eigenschaften des Residuums) Seien P und Q Failureprädikate der Form

- $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$
- $Q : (A \backslash A')^\omega \times \mathbb{P}(A \backslash A') \to \mathbb{B}$

und sei Q Residuum für P hinsichtlich der Abstraktion von A'. Dann gilt

$$\frac{P \text{ sat } P(t,R)}{P \backslash A' \text{ sat } Q(t,R)}$$

•

Beweis 4.2.2 (Satz 4.2.2) Folgt unmittelbar aus den Definitionen 4.2.6 sowie 3.1.5 und 3.1.3. □

4.2.3 Notwendigkeit des unendlichen Modells

Bereits in Kapitel 3 wurde darauf hingewiesen, daß sich das Spurmodell und das endliche Failuremodell hinsichtlich der Modellierung einer Klasse von Systemen, nämlich der *divergenten* Systemen, unterscheiden. Um einen durchgängigen Übergang zwischen diesen Modellen möglich zu machen, war es deshalb notwendig, eine einheitliche Sichtweise auf

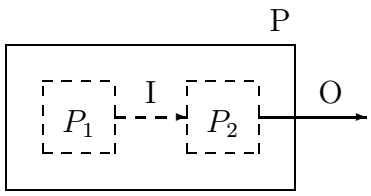

Abbildung 4.1: Das Pipelinesystem

diese Systeme zu verwenden. Dazu wurde die unendliche Failuresemantik - siehe auch Anhang A - eingeführt. Anhand des folgenden Beispiels soll erläutert werden, daß es nicht nur aus diesem Grund für eine durchgängige Vorgehensweise notwendig ist, die unendliche Failuresemantik zu verwenden. Auch bei der Bestimmung von Residuen divergenter Systeme treten bei der Verwendung der endlichen Failuresemantik unerwünschte Ergebnisse auf.

Beispiel 4.2.1 (Divergentes System) Zum Nachweis der Unzulänglichkeit des endlichen Failuremodells wird das in Abbildung 4.1 dargestellte, einfache Pipelinesystem verwendet. Es besteht auch einem Erzeugerprozeß P_1 sowie einer nachgeschalteten Einheit P_2, die die von P_1 erzeugten Pakete aus D bearbeitet und an die Umgebung weitergibt. Das Verhalten der Komponenten wird zunächst auf Spurebene beschrieben. Dazu werden entsprechende Definitionen von I, O und data wie in Beispiel 2.2.1 verwendet. Die Komponente P_1 erzeugt unbeschränkt beliebige Nachrichtenpakete. P_1 wird definiert als $(\emptyset, I, \{t \mid T_1(t)\})$, wobei $I \neq \emptyset$ und

$$T_1(t) \stackrel{\text{def}}{=} t \in I^\infty$$

Die Komponente P_2 verarbeitet diese Pakete und gibt sie an die Umgebung weiter. P_2 wird definiert als $(I, O, \{t \mid T_2(t)\})$ mit $O \neq \emptyset$ und

$$T_2(t) \stackrel{\text{def}}{=} (\forall s \sqsubseteq t.\, \mathsf{data}^*(O \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright s))) \wedge \mathsf{data}^*(O \copyright t) = \mathsf{f}^*(\mathsf{data}^*(I \copyright t))$$

wobei $\mathsf{f} : D \to D$ die surjektive Verarbeitungsfunktion von P_2 beschreibt, die hier nicht näher spezifiziert wird. Die Beschreibung des Gesamtsystems ergibt sich damit zu $(\emptyset, O, \{t \mid T(t)\})$, wobei

$$T(t) \stackrel{\text{def}}{=} t \in O^\infty$$

Die entsprechenden implizierten Failurebeschreibungen F_1, F_2 und F zu T_1, T_2 und T im *endlichen Failuremodell* sind

$$\begin{aligned}
F_1(t, R) &\stackrel{\text{def}}{=} I \not\subseteq R \\
F_2(t, R) &\stackrel{\text{def}}{=} (\forall s \sqsubseteq t.\, \mathsf{data}^*(O \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright s))) \wedge \\
&\quad\; I \cap R = \emptyset \wedge \\
&\quad\; \forall o \in O.(\mathsf{data}^*((O \copyright t) \circ o) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright t)) \Rightarrow o \notin R) \\
F(t, R) &\stackrel{\text{def}}{=} O \not\subseteq R
\end{aligned}$$

da wegen $t \in I^* \Rightarrow t \notin I^\infty$ auch $t \in I^* \Rightarrow \neg T_1(t)$ und entsprechend $t \in O^* \Rightarrow \neg T(t)$ gilt.

Soll nun die minimale Eingabebereitschaft von P_2 ermittelt werden, so führt dies im Fall der endlichen Failuresemantik nicht zum gewünschten Erfolg. Die Bestimmung des Residuums für F hinsichtlich der Abstraktion von I - unter Verwendung des später in Abschnitt 4.4.1 definierten Hilfsatzes 4.4.1 - liefert

$$Q(t, R) \stackrel{\text{def}}{=} I \not\subseteq R \vee O \not\subseteq R$$

Im zweiten Schritt wird das Residuum von Q hinsichtlich der Parallelkomposition mit F_1 bestimmt. Hier ist das entsprechende Residuum

$$Q_2(t, R) \stackrel{\text{def}}{=} \mathbf{T}$$

Dazu wird die hinreichende und notwendige Eigenschaft des Residuums nachgewiesen, also einerseits

$$\begin{array}{ll} & F_1(I \copyright t, R_1) \wedge Q_2((I \cup O) \copyright t, R_2) \\ \Longrightarrow & [\text{Def. } Q_2] \\ & F_1(I \copyright t, R_1) \wedge \mathbf{T} \\ \Longrightarrow & [\text{Def. } F_1] \\ & I \not\subseteq R_1 \\ \Longrightarrow & [I \cap R_2 = \emptyset] \\ & I \not\subseteq R_1 \cup R_2 \\ \Longrightarrow & [\text{Aussagenlogik}] \\ & I \not\subseteq R_1 \cup R_2 \vee O \not\subseteq R_1 \cup R_2 \\ \Longrightarrow & [\text{Def. } Q_1] \\ & Q(t, R_1 \cup R_2) \end{array}$$

Andererseits gilt die notwendige Eigenschaft trivialerweise, da $\mathbf{T}$ das schwächstmögliche Prädikat ist und keine andere notwendige Eigenschaft schwächer als $\mathbf{T}$ sein kann.

Damit hat die Bestimmung des Residuums für F_2 ergeben, daß jede beliebige Eingabebereitschaft für P_2 ausreichend ist, um die gewünschte Anforderung an das System ableiten zu können. Insbesondere müßte damit auch der niemals eingabebereite Prozeß ausreichend sein, was aber offensichtlich nicht der Fall ist. ◇

4.3 Vorgehensweise

In den Abschnitten 2.6, 3.5 und 3.7 wurde jeweils gezeigt, welche Schritte zu durchlaufen sind, um unter Ausnutzung der asynchronen Systemsicht eine synchrone Implementierung eines System zu entwickeln:

- die Entwicklung von Spurbeschreibungen von System und Komponenten sowie der Nachweis der Implementierung des Systems durch die Komponenten,

- die Umsetzung der Spurbeschreibungen von System und Komponenten in äquivalente Failurebeschreibungen,
- die Abschwächung der Anforderungen an die Ein- und Ausgabebereitschaft des Systems.

Damit bleibt als letzter Schritt des Übergangs von einer asynchronen zu einer synchronen Systemsicht die Abschwächung der Anforderungen an die Ein- und eventuell auch Ausgabebereitschaft der Komponenten des Systems. Dabei muß die Abschwächung einerseits die Implementierungsbeziehung zwischen System und Komponenten erhalten und andererseits die Realisierung möglichst speichereffizienter Systeme erlauben. Mit der Einführung der Residuen für die Parallelkomposition und die Abstraktion in Abschnitt 4.2 wurde die Voraussetzung geschaffen, aus den Anforderungen an ein System die minimalen Anforderungen an seine Komponenten zu bestimmen. Für die *schematische* Vorgehensweise, wie in Kapitel 1 gefordert, muß nun die Bestimmung von Residuen auf die Beschreibung synchroner Systeme abgestimmt werden. Dazu werden die beiden folgenden Techniken bereitgestellt:

- die Einführung einer am syntaktischen Aufbau eines Systems orientierte Verfahrensweise zur Bestimmung der minimalen Anforderungen an die Komponenten eines Systems in Abschnitt 4.3.1 und
- die Einschränkung der Bestimmung des Residuums einer Komponente auf die Eingabe- bzw. Ein- und Ausgabebereitschaft in Abschnitt 4.3.2.

In Abschnitt 4.3.3 werden diese beiden Techniken an einem einfachen Beispiel demonstriert.

4.3.1 Aufbauorientierte Verfahrensweise

In Abschnitt 2.6 und insbesondere in Abschnitt 2.6.3 wurde gezeigt, wie die Komponenten $S_1, \ldots, S_n$ zur Implementierung eines System S kombiniert werden können. Dabei wird stets die syntaktische Form

$$S = (S_1 \parallel \ldots \parallel S_n)\backslash H$$

verwendet. Entsprechend diesem Schema wird die Konstruktion eines Systems S aus den Komponenten $S_1, \ldots, S_n$ durch die folgenden beiden Schritte bewerkstelligt:

- Die Parallelkomposition der Komponenten $S_1, \ldots, S_n$ zu $S_1 \parallel \ldots S_n$, und
- die Abstraktion von internen Aktionen mittels $\backslash H$.

Wird dabei eine Komponente selbst wieder durch ein System von Unterkomponenten implementiert, so wird auch dafür das gleiche Schema verwendet. Im folgenden wird gezeigt, wie für derartig aufgebaute Systeme die Residuenbestimmung verwendet werden kann, um minimale Anforderungen an die Systemkomponenten zu bestimmen.

Auflösung der syntaktischen Struktur

Die durch die Abschwächung festgelegten Anforderungen an die Ein- und Ausgabebereitschaft beziehen sich auf das Gesamtsystem $(S_1 \parallel \ldots \parallel S_n)\backslash H$. Ausgehend von diesen Anforderungen müssen nun minimale Anforderungen an die einzelnen Komponenten $S_1, \ldots, S_n$ hergeleitet werden, die anschließend eine speichereffiziente Realisierung erlauben. Für diese Herleitung sollen die Residuen für Parallelkomposition und Abstraktion verwendet werden. Wie in Abschnitt 4.1 allgemein sowie in Abschnitt 4.2 für den Fall der verteilten Systeme gezeigt, ist die Bestimmung minimaler Anforderungen immer nur bezüglich eines Systemrests definiert. Im Fall der Parallelkomposition zweier Komponenten S_1 und S_2 zu $S_1 \parallel S_2$ bedeutet dies, daß eine Bestimmung des Residuums für $S_1 \parallel S_2$ die minimalen Anforderungen

- für S_1 liefert, wenn die Anforderungen an S_2 beibehalten werden, sowie entsprechend
- für S_2, wenn die Anforderungen an S_1 beibehalten werden.

Die Residuenbestimmung erlaubt also nicht die gleichzeitige Bestimmung der Anforderungen aller Komponenten einer Parallelkomposition. Da die Abstraktion eine einstellige Operation ist, liegt hier kein Systemrest vor. Mit dieser Beschränkung ergibt sich die folgende Vorgehensweise:

1. Aus den abgeschwächten Anforderungen an das Gesamtsystem S werden die Anforderungen an $S_1 \parallel \ldots \parallel S_n$ bestimmt, indem die Abstraktion von H rückgängig gemacht wird.
2. Es wird eine Komponente S_i festgelegt, zu der das Residuum für die Anforderungen bestimmt werden soll, die in Schritt 1 hergeleitet wurden. Die Bestimmung wird dabei *unter Beibehaltung* des Systemrests bestehend aus den Komponenten $S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n$ vorgenommen. Die Bestimmung liefert eine neue Komponente S_i' und damit ein neues System $S_1 \parallel \ldots \parallel S_{i-1} \parallel S_i' \parallel S_{i+1} \parallel \ldots \parallel S_n$.
3. Für jede weitere Komponente, für die die minimalen Anforderungen bestimmt werden sollen, wird Schritt 2 mit der bisher gewonnenen Systembeschreibung wiederholt.

Dabei ist die Reihenfolge der Bestimmung minimaler Anforderungen von Bedeutung. In Abhängigkeit der Reihenfolge, in der die minimalen Anforderungen der Komponenten bestimmt werden, ergeben sich unterschiedliche minimale Anforderungen an die einzelnen Komponenten.

Das System wird durch die Anwendung der Parallelkomposition und die Abstraktion aus den Komponenten aufgebaut. Zur Bestimmung der Anforderungen an eine einzelne Komponente muß diese aus dem System herausgelöst werden. Damit müssen die Parallelkomposition und die Abstraktion wieder rückgängig gemacht werden. Dazu werden die beiden Techniken

- Elimination der Abstraktion von internen Aktionen

- Elimination von parallelkomponierten Systemteilen

benötigt. Diese beiden Techniken werden im folgenden beschrieben.

Elimination der Abstraktion

Entsprechend dem Aufbau der hier beschriebenen Systeme wird es im allgemeinen nötig sein, zur Bestimmung der minimalen Anforderungen an eine einzelne Komponenten zuerst die Anwendung der Abstraktion interner Aktionen rückgängig zu machen. Der Elimination der Abstraktion auf syntaktischer Seite entspricht auf der Seite der Anforderungen die Bestimmung des Residuums hinsichtlich der Abstraktion. Dabei sind für die Elimination der Abstraktion auf syntaktischer Seite vorgegeben:

- das System $S\backslash C$ mit Alphabet $A\backslash C$
- das System S mit Alphabet A
- die verborgenen Aktionen C

Auf der Seite der Anforderungen liegen vor

- die Anforderung an das System $S\backslash C$ mit abstrahierten internen Aktionen C in der Form $F : (A\backslash C)^\omega \times \mathbb{P}(A\backslash C) \to \mathbb{B}$

Das Ergebnis der Elimination der Abstraktion sind

- die minimalen Anforderungen an das System S in der Form $F' : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$

Damit liefert die Bestimmung des Residuums von F hinsichtlich der Abstraktion von C die gewünschte Elimination der Abstraktion.

Elimination der Parallelkomposition

Nach der Elimination der Abstraktion von internen Aktionen des Systems liegt nun das System in der Form $S_1 \parallel \ldots \parallel S_n$ vor. Um die Anforderungen an eine Komponente S_i zu bestimmen, müssen die übrigen Komponenten $S_1, \ldots, \|, S_{i-1}, S_{i+1}, \ldots, S_n$ eliminiert werden. Dazu wird die Elimination der Parallelkomposition in Teilschritte zerlegt. In jedem Teilschritt wird dabei sukzessive eine der parallelkomponierten Komponenten eliminiert. Damit ist für *jeden Teilschritt* aus syntaktischer Sicht vorgegeben:

- das System $S'_1 \parallel \ldots \parallel S'_m$ mit dem Alphabet $A = A'_1 \cup \ldots \cup A'_m$, wobei A'_j das Alphabet der Komponente S'_j ist
- die zu eliminierende Komponente S'_k mit Alphabet A'_k

Dabei besteht das System $S'_1 \parallel \ldots \parallel S'_m$ aus allen noch zu eliminierenden Komponenten sowie der Komponente S_i, deren minimalen Anforderungen zu bestimmen sind. Auf der Seite der Anforderungen liegen damit vor

- die Anforderungen $F : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$ an das System $S'_1 \parallel \ldots \parallel S'_m$
- die Anforderungen $F_k : A_k^\omega \times \mathbb{P}(A_k) \to \mathbb{B}$ an die Komponente S'_k

Das Ergebnis eines Teilschritts ist damit auf syntaktischer Seite

- das System $S'_1 \parallel \ldots \parallel S'_{k-1} \parallel S'_{k+1} \parallel \ldots \parallel S'_m$ mit dem Alphabet $A' = A'_1 \cup \ldots \cup A'_{k-1} \cup A'_{k+1} \cup \ldots \cup A'_m$

sowie auf Seite der Anforderungen

- die minimalen Anforderungen $F' : A'^\omega \times \mathbb{P}(A) \to \mathbb{B}$ des verbleibenden Systems $S'_1 \parallel \ldots \parallel S'_{k-1} \parallel S'_{k+1} \parallel \ldots \parallel S'_m$

Der Ausgangspunkt des ersten Teilschritts ist damit das System $S_1 \parallel \ldots \parallel S_n$, der Endpunkt des letzten Teilschritts die gewünschte Komponente S_i.

Einsatz der Eliminationsschritte

Die Abbildungen 4.2, 4.3 und 4.4 zeigen die Anwendung der beiden Eliminationschritte exemplarisch für die Bestimmung der Anforderungen an die Komponente P_3 eines Systems bestehend aus P_1, P_2 und P_3. Im ersten Schritt wird zunächst die Abstraktion von internen Aktionen rückgängig gemacht und damit das Zusammenspiel der drei Komponenten wieder sichtbar. So werden aus den Anforderungen an das Gesamtsystem die Anforderungen an die Parallelkomposition dieser drei Komponenten hergeleitet.

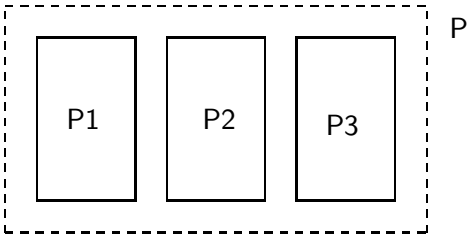

Abbildung 4.2: Elimination der Abstraktion

Um nun die Anforderungen für P_3 alleine herleiten zu können, wird zunächst die Parallelkomposition der Komponente P_1 rückgängig gemacht und damit die Anforderungen an die Parallelkomposition von P_2 und P_3 bestimmt. Diese Elimination der Parallelkomposition wird bis zur vollständigen Isolierung von P_3 wiederholt. Die Elimination der Parallelkomposition der letzten verbleibenden Komponente P_2 in einem dritten Schritt liefert bereits die Anforderungen an P_3.

Der Einsatz der schrittweisen Elimination von Abstraktion und Parallelkomposition wird in den Abschnitten 4.3.3 und 4.6 an Beispielen demonstriert.

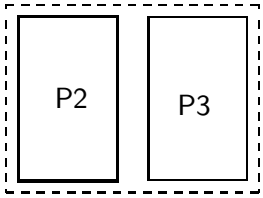

Abbildung 4.3: Elimination der ersten Komponente

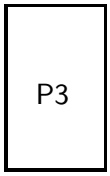

Abbildung 4.4: Elimination der zweiten Komponente

4.3.2 Residuum für Prozeßteile

In Abschnitt 4.3.1 wurde die Residuenbestimmung für das Herauslösen einer Komponenten aus dem System definiert und damit am syntaktischen Aufbau des Systems orientiert. Damit ist es möglich, die Anforderungen an das mittels Parallelkomposition und Abstraktion aufgebaute System auf die Anforderungen an die abzuschwächende Komponente einzuschränken. Wie aber bereits in den Kapitel 1 und 3 erwähnt, werden in der hier beschriebenen Vorgehensweise nicht die gesamten minimalen Anforderungen einer Komponente bestimmt: nur die Eingabebereitschaft der Komponenten eines System soll abgeschwächt werden, nicht aber das weitere, für die Komponenten bereits definierte Verhalten. Das bedeutet, daß nicht das gesamte Residuum für eine Komponente ermittelt werden soll, sondern nur das Residuum für die Eingabebereitschaft, falls die Beschreibung der partiellen Abläufe und der Ausgabebereitschaft beibehalten wird. Daher ist eine dritte Art der Residuenbestimmung nötig, die sich im Gegensatz zu den beiden vorherigen nicht am Aufbau des Systems mit den Konstruktionsoperatoren orientiert. Statt dessen ist es die Aufgabe dieses Residuums, die Anteile einer Prozeßspezifikation festzulegen, die benötigt werden, um zusammen mit weiteren vorgegebenen Eigenschaften die gesamte Spezifikation sicherzustellen.

Für die Failurebeschreibung eines Prozesses, die - wie in Kapitel 3 beschrieben - aus der Spurbeschreibung des Prozesses abgeleitet wird, sind drei Anteile vorgegeben: die Charakterisierung der partiellen Abläufe S, die Charakterisierung der Ausgabebereitschaft F^O und die Charakterisierung der Eingabebereitschaft F^I. Aus diesen ergibt sich die Failurebeschreibung des Prozesses F durch einfache Konjunktion:

$$F(t, R) \stackrel{\text{def}}{=} S(t, R) \wedge F^I(t, R) \wedge F^O(t, R)$$

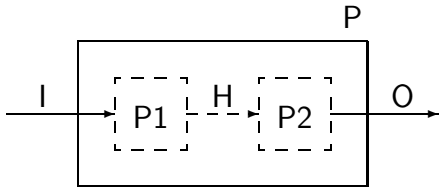

Abbildung 4.5: Das Pipelinesystem

Um also die Eingabebereitschaft zu minimieren, reicht es, das Residuum von F hinsichtlich der Konjunktion von S und F^O zu bestimmen.

In den Beispiel 4.1.1 und 4.1.2 wurde bereits die Eigenschaft bestimmt, die sowohl notwendig als auch hinreichend hinsichtlich der Konjunktion ist. Dies rechtfertigt die folgende Definition.

Definition 4.3.1 (Residuum hinsichtlich Konjunktion) Es seien P, P_1 und P_2 mit $P, P_1, P_2 : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$ Failureprädikate. Dann heißt P_2 mit

$$P_2(t, R) \stackrel{\text{def}}{=} P_1(t, R) \Rightarrow P(t, R)$$

das *Residuum für P hinsichtlich der Konjunktion von P_1*. ∘

4.3.3 Beispiel

Anhand des folgenden einfachen Beispiels werden die wesentlichen Schritte bei der Residuenbestimmung demonstriert. Dazu wird das in Abbildung 4.5 dargestellte System einer Pipeline betrachtet mit zwei sehr einfachen, stark unterspezifizierten Komponenten. Diese einfache Reihenschaltung wird beispielsweise in [JG88] zur Textransformation verwendet, allerdings mit komplexerem Verhalten der Komponenten. Um die schrittweise Vorgehensweise zu demonstrieren, wurde dieses einfache Beispiel gewählt. Ein realistischeres System wird in Beispiel 4.6.1 betrachtet.

Beispiel 4.3.1 (Pipelinesystem) Für die Beschreibung des Pipelinesystem werden entsprechende Definitionen wie in Beispiel 4.2.1 aus Abschnitt 4.2.3 verwendet. Das Pipelinesystem besteht aus zwei Komponenten: P1 mit Eingabekanal I und Ausgabekanal H sowie P2 mit Eingabekanal H und Ausgabekanal O. Beide Komponenten arbeiten mit einem sehr einfachen Modus, bei dem eine surjektive Funktion f bzw. g elementweise auf die eingehenden Pakete angewendet wird und diese dann ausgegeben werden. Das Gesamtsystem P, das durch die Parallelkomposition von P1 und P2 mit anschließendem Verbergen des internen Kanals H entsteht, entspricht der elementweisen Anwendung von $\mathsf{g} \circ \mathsf{f}$ auf die eingehenden Pakete und deren Ausgabe.

Mittels der relationalen Darstellung lassen sich die drei Prozesse wie folgt beschreiben:

Komponente P1: Der Prozeß mit der relationalen Beschreibung (P_1, C_1) mit

$$\begin{aligned} P_1(i,o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(i)) \\ C_1(i,o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) = \mathsf{f}^*(\mathsf{data}^*(i)) \end{aligned}$$

Komponente P2: Der Prozeß mit der relationalen Beschreibung (P_2, C_2) mit

$$\begin{aligned} P_2(i,o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) \sqsubseteq \mathsf{g}^*(\mathsf{data}^*(i)) \\ C_2(i,o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) = \mathsf{g}^*(\mathsf{data}^*(i)) \end{aligned}$$

System P: Der Prozeß mit der relationalen Beschreibung (P, C) mit

$$\begin{aligned} P(i,o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) \sqsubseteq (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(i)) \\ C(i,o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) = (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(i)) \end{aligned}$$

Im folgenden werden nun - entsprechend Abschnitt 3.5 - die Failuredarstellungen der Prozesse durch Ermittlung der implizierten Failureeigenschaft aus Definition 3.2.1 bzw. 3.4.2 festgelegt:

Komponente P1: Für die Komponente P_1 ergibt sich gemäß den expliziten, durch (P_1, C_1) implizierten Failureeigenschaften die Darstellung

$$\begin{aligned} S_{P_1,C_1}(t) &\stackrel{\text{def}}{=} \forall s \sqsubseteq t.\, \mathsf{data}^*(H \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright s)) \\ F^I_{P_1,C_1}(t,R) &\stackrel{\text{def}}{=} I \cap R = \emptyset \\ F^H_{P_1,C_1}(t,R) &\stackrel{\text{def}}{=} \mathsf{data}^*(H \copyright t) \sqsubset \mathsf{f}^*(\mathsf{data}^*(I \copyright t)) \Rightarrow \\ &\qquad \exists o \in H.(o \notin R \wedge \mathsf{data}^*((H \copyright t) \circ o) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright t))) \end{aligned}$$

Komponente P2: Analog wie für die Komponente P1 ergibt sich hier

$$\begin{aligned} S_{P_2,C_2}(t) &\stackrel{\text{def}}{=} \forall s \sqsubseteq t.\, \mathsf{data}^*(O \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(H \copyright s)) \\ F^H_{P_2,C_2}(t,R) &\stackrel{\text{def}}{=} H \cap R = \emptyset \\ F^O_{P_2,C_2}(t,R) &\stackrel{\text{def}}{=} \mathsf{data}^*(O \copyright t) \sqsubset \mathsf{f}^*(\mathsf{data}^*(H \copyright t)) \Rightarrow \\ &\qquad \exists o \in O.(o \notin R \wedge \mathsf{data}^*((O \copyright t) \circ o) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(H \copyright t))) \end{aligned}$$

System P: Für das Gesamtsystem werden - wie in 3.5 erläutert - nicht die expliziten Failureeigenschaften verwendet, da hier bereits das implizite Schema ausreicht. Damit ergibt sich die Anforderung an die Ausgabebereitschaft

$$F^O_{T_{P,C}} P(t,R) \stackrel{\text{def}}{=} O \subseteq R \Rightarrow T_{P,C}(t)$$

Da die partiellen Eigenschaften erhalten bleiben, wird an die Ausgabebereitschaft nur die Anforderung

$$F^O(t,R) \stackrel{\text{def}}{=} O \subseteq R \Rightarrow C(t)$$

gestellt. Als Anforderung an die Eingabebereitschaft wird gefordert, daß bei geleerter Pipeline stets eine Eingabe möglich sein soll:

$$F^I_P(t,R) \stackrel{\text{def}}{=} (I \copyright t \in I^* \wedge \mathsf{data}^*(I \copyright t) = (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(I \copyright t))) \Rightarrow I \cap R = \emptyset$$

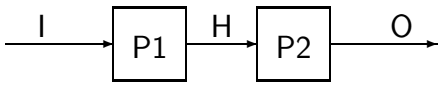

Abbildung 4.6: Elimination von $\backslash H$

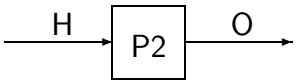

Abbildung 4.7: Elimination von $\| P_1$

Die Eingabe kann also nur verweigert werden, wenn sich noch eine Nachricht in der Pipeline befindet.

Nun können die Residuen der beiden Komponenten P1 und P2 bestimmt werden. Hier soll das Residuum für P2 bestimmt werden; dabei wird hier der Einfachheit halber nur die Ausgabebereitschaft betrachtet. Dazu wird - wie in Abbildung 4.6 dargestellt - im ersten Schritt "$\backslash H$" eliminiert. Das Residuum Q der Ausgabebereitschaft F_P^O des Systems P hinsichtlich der Abstraktion von H ist

$$Q(t,R) \stackrel{\text{def}}{=} \neg C(t) \Rightarrow O \cup H \not\subseteq R \tag{4.7}$$

Um nachzuweisen, daß Q tatsächlich das Residuum ist, ist zu zeigen, daß es sowohl eine hinreichende als auch eine notwendige Eigenschaft für F_P^O hinsichtlich der Abstraktion von H darstellt. Hilfsatz 4.4.1 aus Abschnitt 4.4.1 wird zeigen, daß diese Residuenbestimmung allgemein einsetzbar ist. Im zweiten Schritt wird - wie in Abbildung 4.7 gezeigt - nun "$\| P_1$" entfernt. Da nur die Eingabebereitschaft von P_2 abgeschwächt werden soll, werden die Charakterisierungen der partiellen Abläufe und der Ausgabebereitschaft beibehalten. Somit ergibt sich als schwächste Anforderung an die Eingabebereitschaft von P2 das Residuum Q_{P_2} mit

$$Q_{P_2}(t,R) \stackrel{\text{def}}{=} (H©t \in H^* \wedge \mathsf{g}^*(\mathsf{data}^*(O©t)) = \mathsf{data}^*(H©t)) \Rightarrow H \cap R = \emptyset \tag{4.8}$$

Auch hier ist wieder nachzuweisen, daß Q_{P_2} eine hinreichende und notwendige Anforderung für Q hinsichtlich der Parallelkomposition von P_1 und der Konjunktion von $F^O_{P_2,C_2}$ darstellt. Die Eigenschaft, hinreichend für Q zu sein, folgt dabei direkt aus Q_{P_2}, F_{P_1,C_1} und $F^O_{P_2,C_2}$:

$$\begin{array}{ll}
& O \cup H \subseteq R_1 \cup R_2 \\
\Longrightarrow & [R_1 \subseteq I_1 \cup H_1] \\
& O \subseteq R_2 \wedge H \subseteq R_1 \cup R_2 \\
\Longrightarrow & \left[F^O_{P_2,C_2}(A_2©t, R_2)\right] \\
& \mathsf{g}^*(\mathsf{data}^*(H©t)) = \mathsf{data}^*(O©t) \wedge H_1 \subseteq R_1 \cup R_2 \\
\Longrightarrow & [Q_{P_2}(A_2©t, R_2)] \\
& \mathsf{g}^*(\mathsf{data}^*(H©t)) = \mathsf{data}^*(O©t) \wedge (H©t \notin H^* \vee H \subseteq R_1) \\
\Longrightarrow & \left[F^H_{P_1,C_1}(A_1©t, R_1)\right]
\end{array}$$

$$
\begin{array}{l}
\quad \mathsf{g}^*(\mathsf{data}^*(H©t)) = \mathsf{data}^*(O©t) \wedge (H©t \notin H^* \vee \mathsf{f}^*(\mathsf{data}^*(I©t)) = \mathsf{data}^*(H©t)) \\
\Longrightarrow [S_{P_1,C_1}(A_1©t, R_1)] \\
\quad \mathsf{g}^*(\mathsf{data}^*(H©t)) = \mathsf{data}^*(O©t) \wedge \\
\quad ((H©t \notin H^* \wedge \mathsf{data}^*(H©t) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I©t))) \vee \mathsf{f}^*(\mathsf{data}^*(I©t)) = \mathsf{data}^*(H©t)) \\
\Longrightarrow [\text{Definition } \sqsubseteq] \\
\quad \mathsf{g}^*(\mathsf{data}^*(H©t)) = \mathsf{data}^*(O©t) \wedge \mathsf{f}^*(\mathsf{data}^*(I©t)) = \mathsf{data}^*(H©t) \\
\Longrightarrow [\text{Def. } (\mathsf{f} \circ \mathsf{g})^*] \\
\quad (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(I©t)) = \mathsf{data}^*(O©t)
\end{array}
$$

Weiterhin ist zu zeigen, daß Q_{P_2} auch notwendig ist für Q hinsichtlich der Parallelkomposition mit L_1. Auf diesen Nachweis wird hier verzichtet und statt dessen auf die in Abschnitt 4.4.2 beschriebene direkte Darstellung der Residuen verwiesen. Dort wird auch speziell die Bestimmung der direkten Darstellung dieses Residuums demonstriert.

Für die Bestimmung der minimalen Eingabebereitschaft des Prozesses P_1 müssen die entsprechenden Schritte angewendet werden. Entsprechend ergibt sich damit

$$(I©t \in I^* \wedge \mathsf{f}^*(\mathsf{data}^*(H©t)) = \mathsf{data}^*(I©t)) \Rightarrow I \cap R \neq \emptyset$$

als Residuum für P_2. ◇

4.4 Direkte Residuendarstellung

In Abschnitt 4.2 wurden die Residuen für den Abstraktions- und den Parallelkompositionsoperator definiert durch deren charakterisierende Eigenschaften, sowie deren Bedeutung für den Entwicklungsprozeß diskutiert. Für eine methodische Vorgehensweise ist es jedoch darüber hinaus wichtig, eine direkte und kompakte Form der Darstellung anzugeben. Daher besteht das Ziel des folgenden Abschnitts darin, aus der indirekten, charakterisierenden Beschreibung der Residuen für die Kompositionsoperatoren eine direkte Darstellung herzuleiten.

4.4.1 Abstraktion

Bereits im Beispiels 4.3.1 hat sich gezeigt, daß die Bestimmung des Residuums hinsichtlich der Abstraktion sehr einfach und sogar syntaktischer Natur ist. Diese Beobachtung läßt sich verallgemeinern. Für die Definition des Herleitungsschemas für die Abstraktion wird der folgende Hilfsatz verwendet:

Hilfsatz 4.4.1 (Abstraktionselimination) Für ein Failuresprädikat $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$ mit einem Alphabet H mit $A \cap H = \emptyset$ ist $Q : (A \cup H)^\omega \times \mathbb{P}(A \cup H) \to \mathbb{B}$ mit

$$Q(t, R) \stackrel{\text{def}}{=} P(A©t, R \backslash H) \vee H \not\subseteq R \tag{4.9}$$

das Residuum hinsichtlich der Abstraktion von H. △

Der Beweis des Hilfsatzes folgt unmittelbar aus der Definition 4.2.6.

Beweis 4.4.1 (Hilfsatz 4.4.1) Für den Nachweis der Residuumseigenschaft wird zuerst der Nachweis geführt, daß mittels 4.9 eine hinreichende Eigenschaft für P definiert ist:

$$
\begin{array}{ll}
& Q(t, R \cup H) \\
\Longleftrightarrow & [4.9] \\
& P(A©t, R \cup H) \backslash H) \vee H \not\subseteq R \cup H) \\
\Longleftrightarrow & [(R \cup H) \backslash H = R \backslash H \text{ und } H \subseteq R \cup H] \\
& P(A©t, R \backslash H)
\end{array}
$$

Wegen $Q(t, R \cup H) \Longleftrightarrow P(A©t, R \backslash H)$ folgt somit auch sofort die Notwendigkeit dieser Eigenschaft hinsichtlich der Abstraktion. □

Damit läßt sich für die Elimination der Abstraktion ein einfaches syntaktisches Schema definieren.

4.4.2 Parallelkomposition

Bei der Elimination der Abstraktion war wegen der einfachen Charakterisierung und der damit verbundenen trivialen Schematisierung eine einfache Bestimmung des Residuums möglich. Im Gegensatz dazu ist die Bestimmung des Residuums wesentlich komplexer für die Parallelkomposition $P_1 \parallel P_2$. Zwar läßt sich auch hier leicht eine Darstellung des Residuums von P_2 für P hinsichtlich der Parallelkomposition mit P_1 angeben, wie der folgende Hilfsatz zeigt:

Hilfsatz 4.4.2 (Parallelkompositionselimination) Für Failuresprädikate $P : A^\omega \times \mathbb{P}(A) \to \mathbb{B}$ und $P_1 : A_1^\omega \times \mathbb{P}(A_1) \to \mathbb{B}$ und Alphabete A_1, A_2 und A mit $A = A_1 \cup A_2$ ist

$$Q(t, R) \stackrel{\text{def}}{=} \forall R', t'.t = A_2©t' \wedge P_1(A_1©t', R') \Rightarrow P(t', R \cup R') \tag{4.10}$$

das Residuum für P hinsichtlich der Parallelkomposition von P_1. △

Der Beweis des Hilfsatzes folgt unmittelbar aus der Definition 4.2.3.

Beweis 4.4.2 (Hilfsatz 4.4.2) Für den Nachweis der Residuumseigenschaft wird zuerst der Nachweis geführt, daß mittels 4.10 eine hinreichende Eigenschaft für P definiert ist:

$$
\begin{array}{ll}
& P_1(A_1©t, R_1) \wedge Q(A_2©t, R_2) \\
\Longleftrightarrow & [\text{Definition 4.10}] \\
& P_1(A_1©t, R_1) \wedge (\forall R', t'.A_2©t = A_2©t' \wedge P_1(A_1©t', R') \Rightarrow P(t', R_2 \cup R')) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& P_1(A_1©t, R_1) \wedge P_1(A_1©t, R_1) \Rightarrow P(t, R_1 \cup R_2) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& P(t, R_1 \cup R_2)
\end{array}
$$

Entsprechend wird gezeigt, daß mittels 4.10 ebenfalls eine notwendige Eigenschaft für P definiert ist:

$$\forall t, R_1, R_2.P_1(A_1©t, R_1) \wedge Q'(A_2©t, R_2) \Rightarrow P(t, R_1 \cup R_2)$$

$\Longrightarrow$ [Prädikatenlogik]
$\forall t, R_1, R_2.Q'(A_2 © t, R_2) \Rightarrow (P_1(A_1 © t, R_1) \Rightarrow P(t, R_1 \cup R_2))$
$\Longrightarrow$ [Prädikatenlogik]
$\forall s, t, R_2.A_2 © t = s \wedge Q'(s, R_2) \Rightarrow (\forall R_1.P_1(A_1 © t, R_1) \Rightarrow P(t, R_1 \cup R_2))$
$\Longrightarrow$ [Prädikatenlogik]
$\forall s, t, R_2.Q'(s, R_2) \Rightarrow (\forall R_1.A_2 © t = s \wedge P_1(A_1 © t, R_1) \Rightarrow P(t, R_1 \cup R_2))$
$\Longrightarrow$ [Prädikatenlogik]
$\forall s, R_2.Q'(s, R_2) \Rightarrow (\forall t, R_1.A_2 © t = s \wedge P_1(A_1 © t, R_1) \Rightarrow P(t, R_1 \cup R_2))$
$\Longrightarrow$ [Definition 4.10]
$\forall s, R_2.Q'(s, R_2) \Rightarrow Q(s, R_2)$

□

Diese Darstellung beschreibt aber das Residuum nur implizit: die Beschreibung enthält noch Anteile aus $A_1 \backslash A_2$, also Anteile, die nicht im Alphabet des Residuums selbst liegen. Für die Durchführung weiterer Residuenbestimmungen und insbesondere für die anschließende Implementierung ist aber eine explizite Beschreibung ohne alphabetsfremde Anteile wünschenswert. Diese explizite Beschreibung des Residuums ohne die alphabetsfremden Anteile aus $A_1 \backslash A_2$ ist jedoch abhängig von der Definition von P_1 und P und damit nicht allgemein möglich.

Für eine schematisierte Bestimmung der Residuen bietet sich hier jedoch die Möglichkeit, geeignete Hilfsätze zur Unterstützung der Bestimmung zu entwickeln. Im folgenden Beispiel wird anhand des in Beispiel 4.3.3 eingeführten Pipelinesystems gezeigt, wie aus der impliziten Darstellung des Residuums hinsichtlich der Parallelkomposition eine explizite Darstellung entwickelt wird. Gleichzeitig werden bei der Entwicklung einzelne Beweisschritte herausgestellt, die Beispiele für geeignete Hilfsätze zur schematischen Residuenbestimmung darstellen.

Beispiel 4.4.1 (Pipelinesystem) In Beispiel 4.3.3 wurde gezeigt, wie die Elimination der Abstraktion und der Parallelkomposition eingesetzt wird, um die minimale Anforderung an die Eingabebereitschaft der Komponente P_2 zu bestimmen, die zur Sicherstellung der Ausgabebereitschaft des Systems

$$\mathsf{data}^*(O © t) \sqsubset (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(I © t)) \Rightarrow O \not\subseteq R$$

benötigt wird. Die Elimination der Abstraktion entsprechend der direkten Darstellung aus Hilfsatz 4.4.1 liefert als Residuum P

$$(\mathsf{data}^*(O © t) \sqsubset (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(I © t)) \Rightarrow O \not\subseteq R) \vee H \not\subseteq R$$

bzw.

$$\mathsf{data}^*(O © t) \sqsubset (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(I © t)) \Rightarrow O \cup H \not\subseteq R$$

Für die Elimination der Parallelkomposition von P_1 ist damit entsprechend Hilfsatz 4.4.2 das Residuum definiert als

$$\forall r, S.(A_2 © r = t \wedge P_1(A_1 © r, S)) \Rightarrow P(r, R \cup S)$$

mit obigem Residuum P sowie der Definition von P_1 wie in Beispiel 4.3.3 angeführt. Für die Bestimmung der expliziten Darstellung des Residuums lassen sich zum Teil allgemeine Schritte anwenden, beispielweise für die Elimination des alphabetsfremden Refusalanteils in Form von S. Hierzu wird im folgenden $P_1(A_1 \copyright r, S) \Rightarrow P(r, R \cup S)$ vereinfacht, mit P_1 wie in Beispiel 4.3.3 und obigem P. Dabei wird, wie in Beispiel 4.3.3 bereits angesprochen, das Residuum nicht nur hinsichtlich der Parallelkomposition von P_1 sondern gleichzeitig auch hinsichtlich der Konjunktion von $F^O_{P_2,C_2}$ bestimmt:

$$
\begin{array}{l}
\quad S_{P_1,C_1}(A_1 \copyright r) \wedge F^I_{P_1,C_1}(A_1 \copyright r, S) \wedge F^H_{P_1,C_1}(A_1 \copyright r, S) \wedge \\
\quad F^O_{P_2,C_2}(A_2 \copyright r, R) \wedge \mathsf{data}^*(O \copyright r) \sqsubset (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(I \copyright r)) \Rightarrow \\
\quad\ O \cup H \not\subseteq S \cup R \\
\Longleftrightarrow \left[\text{Definition } S_{P_1,C_1},\, F^I_{P_1,C_1},\, F^H_{P_1,C_1},\, F^O_{P_2,C_2}\right] \\
\quad (\forall s \sqsubseteq r.\, \mathsf{data}^*(H \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright s)) \wedge I \cap S = \emptyset \wedge \\
\quad (\mathsf{data}^*(H \copyright r) = \mathsf{f}^*(\mathsf{data}^*(I \copyright r)) \vee \\
\qquad (\exists h \in H.h \notin S \wedge \mathsf{data}^*((H \copyright r) \circ h) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright r)))) \wedge \\
\quad (\mathsf{data}^*(O \copyright r) = \mathsf{g}^*(\mathsf{data}^*(H \copyright r)) \vee \\
\qquad (\exists o \in O.o \notin S \wedge \mathsf{data}^*((O \copyright r) \circ o) \sqsubseteq \mathsf{g}^*(\mathsf{data}^*(H \copyright r)))) \wedge \\
\quad \mathsf{data}^*(O \copyright r) \sqsubset (\mathsf{f} \circ \mathsf{g})^*(\mathsf{data}^*(I \copyright r)) \Rightarrow \\
\quad\ O \cup H \not\subseteq S \cup R
\end{array}
$$

Dieser Anteil des Residuums kann durch die Verwendung von der Definition von P_1 bzw. P unabhängige Hilfsätze vereinfacht werden. Dabei treten schematisch anwendbare Hilfsätze auf, wie beispielsweise der folgende Hilfsatz zur Elimination des alphabetsfremden Refusalanteils

$$(\forall S.H \not\subseteq S \Rightarrow H \not\subseteq S \cup R) \Leftrightarrow (H \cap R = \emptyset)$$

Diese äquivalenten Umformungen führen zu

$$
\begin{array}{l}
\quad (\mathsf{data}^*(O \copyright r) = \mathsf{g}^*(\mathsf{data}^*(H \copyright r)) \vee \\
\qquad (\exists o \in O.o \notin S \wedge \mathsf{data}^*((O \copyright r) \circ o) \sqsubseteq \mathsf{g}^*(\mathsf{data}^*(H \copyright r)))) \wedge \\
\quad \mathsf{data}^*(O \copyright r) = \mathsf{g}^*(\mathsf{data}^*(H \copyright r) \wedge \\
\quad (\forall s \sqsubseteq r.\, \mathsf{data}^*(H \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright s)) \wedge \mathsf{data}^*(H \copyright r) \sqsubset \mathsf{f}^*(\mathsf{data}^*(I \copyright r)) \Rightarrow \\
\quad\ H \cap R = \emptyset \\
\Longleftrightarrow \left[\text{Definition } F^O_{P_2,C_2}\right] \\
\quad F^O_{P_2,C_2}(A_2 \copyright r, R) \wedge \mathsf{data}^*(O \copyright r) = \mathsf{g}^*(\mathsf{data}^*(H \copyright r) \\
\quad (\forall s \sqsubseteq r.\, \mathsf{data}^*(H \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright s)) \wedge \mathsf{data}^*(H \copyright r) \sqsubset \mathsf{f}^*(\mathsf{data}^*(I \copyright r)) \Rightarrow \\
\quad\ H \cap R = \emptyset
\end{array}
$$

Da hier nur die minimale Anforderung an die Eingabebereitschaft von P_2 bestimmt werden soll, die Ausgabebereitschaft $F^O_{P_2,C_2}$ von P_2 jedoch beibehalten wird, ergibt sich entsprechend des in Abschnitt 4.3.2 und Definition 4.3.1 eingeführten Residuenbegriffs

$$
\begin{array}{l}
\forall r.A_2 \copyright r = t \wedge \mathsf{g}^*(\mathsf{data}^*(H \copyright r)) = \mathsf{data}^*(O \copyright r) \wedge \\
\quad (\forall s \sqsubseteq r.\, \mathsf{data}^*(H \copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I \copyright s)) \wedge \mathsf{data}^*(H \copyright r) \sqsubset \mathsf{f}^*(\mathsf{data}^*(I \copyright r)) \Rightarrow \\
\quad\ H \cap R = \emptyset
\end{array}
$$

Die bisher verwendeten Umformungen sind im wesentlichen unabhängig von der jeweiligen System- bzw. Komponentendefinition. Diese Umformungen sind daher einfach schematisierbar und gut durch Beweissysteme unterstützbar. Darüber hinaus werden für die Ermittlung der expliziten Darstellung jedoch auch spezifische Eigenschaften von P_1 und P benötigt. Im Fall der hier betrachteten Residuenbestimmung wird wesentlich die Surjektivität von f ausgenutzt:

$$
\begin{array}{l}
\quad \forall r.(H \cup O)\copyright t = r \wedge \mathsf{g}^*(\mathsf{data}^*(H\copyright r)) = \mathsf{data}^*(O\copyright r) \wedge \\
\qquad (\forall s \sqsubseteq r.\, \mathsf{data}^*(H\copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I\copyright s)) \wedge \mathsf{data}^*(H\copyright r) \sqsubset \mathsf{f}^*(\mathsf{data}^*(I\copyright r)) \Rightarrow \\
\qquad\quad H \cap R_2 = \emptyset \\
\Longleftrightarrow [\text{Definition } \sqsubset] \\
\quad \forall r.(H \cup O)\copyright t = r \wedge \mathsf{g}^*(\mathsf{data}^*(H\copyright r)) = \mathsf{data}^*(O\copyright r) \wedge H\copyright r \in H^* \wedge \\
\qquad (\forall s \sqsubseteq r.\, \mathsf{data}^*(H\copyright s) \sqsubseteq \mathsf{f}^*(\mathsf{data}^*(I\copyright s)) \wedge \mathsf{data}^*(H\copyright r) \sqsubset \mathsf{f}^*(\mathsf{data}^*(I\copyright r)) \Rightarrow \\
\qquad\quad H \cap R_2 = \emptyset \\
\Longleftrightarrow \left[\begin{array}{ll} \Longrightarrow & \text{Definition } \copyright,\ \text{Definition } f^*,\ H\copyright r \in H^* \\ \Longleftarrow & \text{Prädikatenlogik} \end{array} \right] \\
\quad \forall r.(H \cup O)\copyright t = r \wedge \mathsf{g}^*(\mathsf{data}^*(H\copyright r)) = \mathsf{data}^*(O\copyright r) \wedge H\copyright r \in H^* \wedge \\
\qquad \mathsf{data}^*(H\copyright r) \sqsubset \mathsf{f}^*(\mathsf{data}^*(I\copyright r)) \Rightarrow \\
\qquad\quad H \cap R_2 = \emptyset \\
\Longleftrightarrow \left[\begin{array}{ll} \Longrightarrow & \text{Definition } \copyright,\ f^* \text{ surjektiv},\ H\copyright r \in H^* \\ \Longleftarrow & \text{Prädikatenlogik} \end{array} \right] \\
\quad \forall r.(H \cup O)\copyright t = r \wedge \mathsf{g}^*(\mathsf{data}^*(H\copyright r)) = \mathsf{data}^*(O\copyright r) \wedge H\copyright r \in H^* \Rightarrow \\
\qquad\quad H \cap R_2 = \emptyset \\
\Longleftrightarrow [\text{Prädikatenlogik}] \\
\quad H\copyright t \in H^* \wedge \mathsf{g}^*(\mathsf{data}^*(H\copyright t)) = \mathsf{data}^*(O\copyright t) \Rightarrow H \cap R_2 = \emptyset
\end{array}
$$

Für die hier gezeigte Bestimmung der expliziten Darstellung wurde die Tatsache ausgenutzt, daß der Prozeß P_1 durch ein einfaches Spezifikationsschema beschrieben wird: der Prozeß wird durch die elementweise Anwendung einer surjektiven Funktion charakterisiert. Damit bietet es sich an, entsprechende Hilfsätze für weitere allgemeine Prozeßspezifikationsschemata basierend auf Basiskonstruktionen wie der elementweisen Anwendung zu erstellen, wie sie in vielen Systemspezifikationen verwendet werden. ◇

4.5 Kanalorientierte Realisierung

Für die letztendliche Realisierung sind die entwickelten Anforderungen in implementierbare Spezifikationen umzusetzen. Wie eingangs in Abschnitt 1.3 besprochen, wurde die programmiersprachliche Umsetzung entwickelter Anforderungen bereits in anderen Arbeiten vielfach untersucht[4] und ist daher nicht Teil dieser Arbeit. Jedoch ergeben sich durch die in dieser Arbeit beschriebene Vorgehensweise mit dem Wechsel von der asynchronen zur

[4] Vgl. z.B. ProCoS-Ansatz ([Old91], [ORSS92].

synchronen Sichtweise einige Besonderheiten, die daher im Zusammenhang dieser Arbeit besprochen werden:

- die Festlegung der Schnittstelle der Komponenten des entwickelten Systems und damit die Einführung von Kanälen als explizite Kommunikationsmedien, sowie
- die Einführung nicht explizit spezifizierter Verteilerprozesse bei der Abbildung von System- bzw. Komponentenprozesse auf Implementierungsprozesse.

Dazu wird in Abschnitt 4.5.1 kurz auf den Begriff des Kanals in synchronen Programmierparadigmen sowie auf die Modellierung von Kanälen im endlichen oder unendlichen Failuremodell eingegangen. Insbesondere werden für die informellen Anforderungen der Paradigmen an einen Kanal formale Definitionen angegeben. Abschnitt 4.5.2 zeigt, wie auf der Basis der in dieser Arbeit vorgestellten Systementwicklung eine einfache Vorgehensweise zur Bestimmung der Komponentenschnittstellen angegeben werden kann. Abschnitt 4.5.3 zeigt, unter welchen Umständen zusätzliche Verteilerprozesse mit der Einführung der Kanäle notwendig werden, um eine Realisierung der entwickelten Prozesse mit diesen Kanälen zu ermöglichen.

Mit der Behandlung dieser für die vorgestellte Vorgehensweise spezifischen Fragen ist so der Anschluß an die oben besprochenen Ansätze zur programmiersprachlichen Realisierung von Verhaltensbeschreibungen geschaffen.

4.5.1 Modellierung von Kanälen

Wie eingangs in Abschnitt 1 besprochen, zielt die in dieser Arbeit vorgestellte Vorgehensweise auf die Realisierung verteilter, mittels Nachrichten über *ungepufferte Kanäle* kommunizierende Systeme ab. In der bisher in dieser Arbeit vorgestellten Systementwicklung spielt jedoch der Begriff des *Kanals* noch keine Rolle. Weder in den Modellen der Spur- noch der Failuresemantik[5] trat das Konzept des Kanals auf und auch bei der Modellierung der Systemkomponenten durch Prozesse, dem Wechsel der Modellierungsebene und der Bestimmung minimaler Anforderungen an die Prozesse eines Systems wurden Kanäle nicht explizit verwendet. Für die Realisierung des Systems unter Verwendung der geeigneten Hardware, beispielsweise in Form von `occam`®, ist es über die Spezifikation des Verhaltens der Komponenten in Form von parallel ablaufenden Prozessen hinaus jedoch nötig, die *syntaktische Schnittstelle* der Komponenten in Form der Kommunikationskanäle festzulegen.

Zur Einführung kanalorientierter Kommunikation im unendlichen (und endlichen) Failuremodell wird das Alphabet eines Prozesses disjunkt in Kanalalphabete zerlegt, indem jedem Kanal ein Kanalalphabet zugeordnet wird. Dabei besteht eine Kommunikationsaktion eines Kanalalphabets C aus dem Kanalbezeichner c sowie einem Datenanteil d aus einer

[5] Andere Modelle, z.B. stromverarbeitende Funktionen [BDD+93], enthalten den Kanal als semantisches Konzept.

Menge von Daten D:

$$C = \{c.d \mid d \in D\}$$

Der schreibende Prozeß bietet zum Zeitpunkt des Schreiben ein - oder mehrere - Aktionen aus C an. Der lesende Prozeß eines Kanals bietet zum Zeitpunkt des Lesens alle Aktionen aus C an: wird vom lesenden Prozeß eine Aktion aus C abgelehnt, so können auch alle anderen Aktionen aus C abgelehnt werden. Die Anforderung an den lesenden Prozeß kann formal als Anforderung an die Failuremenge des Prozesses und das Teilalphabet C des Kanals formuliert werden:

Definition 4.5.1 (Eingabekanalalphabet) Eine Teilmenge $C \subseteq A$ des Alphabets A eines Prozesses (A, F) heißt *Eingabekanalalphabet*, wenn

$$(t, R) \in F \wedge C \cap R \neq \emptyset \Rightarrow (t, R \cup C) \in F \tag{4.11}$$

gilt. ∘

Die Eigenschaft 4.11 läßt sich formal für die unendliche Failuresemantik und kanalorientierte Kommunikationskonstrukte wie in [Hoa85] eingeführt zeigen. Auf den formalen Nachweis wird jedoch hier verzichtet, da sich diese Arbeit mit der verhaltensmäßigen Beschreibung von Prozessen, nicht aber deren Beschreibung mittels Prozeßtermen wie in [Hoa85] oder programmiersprachlichen Ausdrücken beschäftigt. Hier soll daher nur die verhaltensmäßige Charakterisierung von Kanälen betrachtet werden.

Eine solche Abschlußeigenschaft für Kanalalphabete gilt nur für Eingabekanalalphabete. Eine ähnliche Abschlußeigenschaft für Ausgabekanäle läßt sich nicht formulieren. Im folgenden wird nun gezeigt, unter welchen Umständen eine Teilmenge des Ein- oder Ausgabealphabets eines Prozesses durch einen Kanal realisiert werden kann.

Entsprechend der Definition von Kanälen in synchronen Programmiersprachen wie beispielweise `occam`®, ist ein Kanal eine

- exklusive
- gerichtete
- ungepufferte
- sequentiell genutzte

Verbindung zwischen zwei Prozessen. Ein Kanal kann also jeweils nur von zwei Prozessen genutzt werden, dabei von einem Prozeß ausschließlich (blockierend) schreibend sowie vom zweiten Prozeß ausschließlich (blockierend) lesend. Um eine Teilmenge des Systemalphabets durch *einen* Kanal realisieren zu können, dürfen die Aktionen dieser Teilmenge insbesondere nur im Ausgabealphabet eines *sequentiellen* Prozesses vorhanden sein.

Werden die Prozeßbeschreibungen entsprechend dem bisherigen Vorgehen entwickelt, so wird die *gerichtete* und *ungepufferte* Nutzung von Kanälen bereits durch die so entwickelten Prozeßbeschreibungen sichergestellt, unabhängig von einer kanalorientierten Beschreibung der Prozesse für alle beliebigen Alphabetsteilmengen: die gerichtete Nutzung wird

durch die Trennung der Prozeßalphabete in Ein- und Ausgaben und die in 2.2.3 geforderte Verträglichkeit garantiert, die ungepufferte Nutzung ist das Ergebnis der Bestimmung der minimalen Anforderungen. Für eine kanalorientierte Realisierung von Prozessen ist also im folgenden noch die Frage der exklusiven und der sequentiellen Nutzung durch den Sender- und den Empfängerprozeß zu klären.

4.5.2 Kanalzuordnung

Im hier vorgestellen Ansatz zur Systementwicklung spielte bisher der Begriff der Schnittstelle oder des Kanals keine wesentliche Rolle: die Schnittstelle eines Prozesses ist durch sein Alphabet gegeben, der Kanal tritt als eigenes Modellierungskonzept auf dieser Beschreibungsebene nicht auf. Wird jedoch von der Ebene der Verhaltensbeschreibung auf die Ebene der programmiersprachlichen Realisierung gewechselt, ist eine Zerlegung der Schnittstelle eines Prozesses in Form seines Alphabets nötig. Die entstehenden Teilalphabete werden dann mit Kanälen identifiziert.

Durch die entwickelte Verhaltensbeschreibung, insbesondere durch die hergeleiteten Anforderungen an die Ein- und Ausgabebereitschaft eines Prozesses bzw. einer Komponente werden jedoch bereits Randbedingungen für die Realisierung eines Teilalphabets durch einen Kanal festgelegt. Für die Festlegung der syntaktischen Schnittstelle in Form von Kanälen ist es damit notwendig, bereits auf der Ebene der verhaltensmäßigen Beschreibung Unterstützung für die Zerlegung des Alphabets eines Prozesses anzugeben. Im folgenden wird dazu die Definition der *sequentiellen Nutzung* eines Teilalphabets eine Prozesses eingeführt.

Wie in Abschnitt 4.5.1 bereits angemerkt wurde, darf ein Kanal von darauf zugreifenden Prozessen nur sequentiell genutzt werden. Das Alphabet eines Prozesses stellt jedoch im allgemeinen eine Schnittstelle zu mehreren Prozessen dar, die parallel auf diese Schnittstelle zugreifen. Wie diese Schnittstelle eines Prozesses von der Umgebung, also den darauf zugreifenden Prozessen genutzt wird, wird in der Verhaltenbeschreibung festgelegt, und dabei insbesondere in den Anforderungen an die Ein- und Ausgabebereitschaft.

Für eine kanalorientierte Beschreibung der Prozesse muß daher das Alphabet des Prozesses in Abhängigkeit der Anforderungen an die Ein- und Ausgabebereitschaft zerlegt werden. Dabei stellen *sequentiell genutzte* Teilalphabete geeignete Kandidaten für die Realisierung durch *einen* Kanal dar.

Definition 4.5.2 (Sequentielle Nutzung) Eine Teilmenge $C \subseteq A$ des Alphabets eines Prozesses (A, F) heißt *sequentiell genutzt*, wenn

$$(t, R) \in F \wedge C \copyright t \notin C^* \Rightarrow (t, R \cup C) \in F$$

gilt. ∘

Für Prozeßterme gebildet mit kanalorientierten Kommunikationskonstrukten läßt sich zeigen, daß sich eine Teilmenge C eines Ausgabealphabets dieses Prozeßterms nur dann durch

einen Kanal im obigen Sinne realisiert werden kann, wenn C durch den von der unendlichen Failuresemantik dem Prozeßterm zugeordneten Prozeß sequentiell genutzt wird.[6]

Anhand der sequentielle Nutzung einer Teilmenge können somit Kandidaten für mögliche Kanäle ermittelt werden, wobei der Grad der Granularität der verwendeten Kanäle vom Entwickler festzulegen ist: Die sequentielle Nutzung einer Teilmenge C führt entsprechend der Definition 4.5.2 auch zur sequentiellen Nutzung einer Teilmenge $C' \subseteq C$.[7] Anschaulich bedeutet dies, daß ein Kanal stets auch in mehrere Unterkanäle aufgespalten werden kann.

Im folgenden wird nun gezeigt, daß die oben eingeführte Definition 4.5.1 eines Eingabekanalalphabets zur Bestimmung sequentiell genutzter Teilmengen des Alphabets eines Prozesses verwendet werden kann. Hilfsatz 4.5.1 stellt fest, daß Eingabekanalalphabete tatsächlich geeignete Kandidaten für die Realisierung durch einen Kanal darstellen.

Hilfsatz 4.5.1 (Sequentielle Eingabe) Sei F_P die Failuredarstellung eines Prozesses P im Sinne von Definition A.3.1 mit Eingabealphabet I und Ausgabealphabet O. Sei weiterhin $C \subseteq I$ ein Eingabekanalalphabet. Dann wird C sequentiell von $F_{P,C}$ genutzt. $\triangle$

Beweis 4.5.1 (Hilfsatz 4.5.1) Der Nachweis folgt aus der Serialität von Aktionen, die Prozesse gemäß Definition A.3.1 erfüllen:

$$
\begin{array}{ll}
 & F_P(t,R) \wedge C \copyright t \notin C^* \\
\Longrightarrow & [C \text{ endlich}] \\
 & \exists c \in C.F_P(t,R) \wedge \{c\} \copyright t \notin C^* \\
\Longrightarrow & [\text{Definition A.3.1, Axiom 6}] \\
 & \exists c \in C.F_P(t, R \cup \{c\}) \\
\Longrightarrow & [\text{Anforderung 4.11}] \\
 & F_P(t, R \cup C)
\end{array}
$$

$\square$

Im Gegensatz zur Eingabe eines Prozesse, die von der Prozeßumgebung kontrolliert wird, wird das Ausgabealphabet eines Prozesses vom Prozeß selbst kontrolliert. Dementsprechend wird das Ausgabealphabet eines *sequentiellen* Prozesses nur *sequentiell genutzt*. Die in Abschnitt 3.4 eingeführte Failuredarstellung der Ausgabebereitschaft eines auf der Spurebene relational charakterisierten Prozesses beschreibt *einen sequentiellen* Prozeß. Daher wird das Ausgabealphabet der durch die in Abschnitt 3.4 beschriebene Transformation gewonnenen Prozesse sequentiell genutzt. Der folgenden Hilfsatz faßt diese Feststellung zusammen.

Hilfsatz 4.5.2 (Sequentielle Ausgabe) Sei $F_{P,C}$ die explizite, durch (P, C) implizierte Failuredarstellung eines monotonen oder unterbrechbaren Prozesses mit Eingabealphabet I und Ausgabealphabet O. Dann wird O sequentiell von $F_{P,C}$ genutzt. $\triangle$

[6] Der Nachweis wird entsprechend geführt wie der Nachweis der aktionsweisen Parallelität in 3.3.1 unter Verwendung von entsprechenden Definitionen für Eingabeaktionen $c?x$ und Ausgabeaktionen $c!y$ wie in [Hoa85].

[7] Der Nachweis dieser Transitivitätseigenschaft folgt aus der Definition von ©.

Beweis 4.5.2 (Satz 4.5.2) Der Nachweis der Aussage folgt im wesentlichen aus der Sequentialitätseigenschaft 2.4:

$$
\begin{array}{ll}
& F_{P,C}(t,R) \wedge O\copyright t \notin O^* \\
\Longrightarrow & [\text{Definition } F_{P,C}] \\
& F_{P,C}(t,R) \wedge P(I\copyright t, O\copyright) \wedge O\copyright t \notin O^* \\
\Longrightarrow & [(P,C) \text{ sequentiell (2.4)}] \\
& F_{P,C}(t,R) \wedge (\exists o'.C(I\copyright t, (O\copyright) \circ o')) \wedge O\copyright t \notin O^* \\
\Longrightarrow & [\text{Definition } \circ,\ O\copyright t \notin O^*] \\
& F_{P,C}(t,R) \wedge (\exists o'.C(I\copyright t, O\copyright t)) \wedge O\copyright t \notin O^* \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& F_{P,C}(t,R) \wedge C(I\copyright t, O\copyright t) \\
\Longrightarrow & [\text{Definition } F_{P,C}] \\
& S_{P,C}(t) \wedge F^I(t,R) \wedge F^0_{P,C}(t,R) \wedge C(I\copyright t, O\copyright t) \\
\Longrightarrow & \left[\text{Definition } F^I, F^O_{P,C}\right] \\
& S_{P,C}(t) \wedge F^I(t, R \cup O) \wedge F^0_{P,C}(t, R \cup O) \\
\Longrightarrow & [\text{Definition } F_{P,C}] \\
& F_{P,C}(t, R \cup O)
\end{array}
$$

□

Wie oben bemerkt, folgt aus der sequentiellen Nutzung eines Teilalphabets auch die sequentielle Nutzung aller Teilmengen dieses Teilalphabets. Daher kann das Ausgabealphabet eines solchen sequentiellen Prozesses beliebig in Kanäle aufgespalten werden.

Mit den Hilfsätzen 4.5.1 und 4.5.2 sind die notwendigen Kriterien eingeführt, um anhand der Verhaltenbeschreibung eines Prozesses die möglichen kanalorientierten Schnittstellen festzulegen. Für die Verwendung als Eingabekanal muß eine Teilmenge des Eingabealphabet eines Prozesses nur auf die Eigenschaft als Eingabekanalalphabet entsprechend Definition 4.5.1 überprüft werden. Für die Eignung als Ausgabekanal ist für eine Teilmenge des Ausgabealphabets eines Prozesses keine besondere Überprüfung notwendig. Wegen der Transitivität der sequentiellen Nutzung werden in beiden Fällen nur die gröbsten nötigen Kanalstrukturen festgelegt, beliebige feinere Strukturen können dabei vom Entwickler jeweils frei gewählt werden.

4.5.3 Prozeßrealisierung

In Abschnitt 4.5.1 wurde besprochen, daß für die kanalorientierte programmiersprachliche Nutzung eines Systems neben der sequentiellen auch die *exklusive* Nutzung von Kanälen durch jeweils *ein Paar* von Sender- und Empfängerprozessen notwendig ist. Wie in Beispiel 2.2.3 verwendet, erlaubt es die Parallelkomposition bei der Modellierung auf der Spurebene, die Ausgabe eines Prozesses als Eingabe *mehrerer* Prozesse zu verwenden. Diese Form der 1:n-Kommunikation ist bei einer exklusiven Nutzung eines Kanals von Sender und Empfänger nicht möglich. Hier sind nur 1:1-Verbindungen erlaubt. Daher wird es in diesen Fällen nötig, die Prozesse - wie in Abbildung 4.8 gezeigt - in zwei Schritten anzupassen:

- Anpassung der Alphabete
- Einführung eines Verteilerprozesses

Durch die Anpassung der Alphabete A_j der Empfängerprozesse $P_1, \ldots, P_n$ wird das gemeinsame Teilalphabet I der Empfängerprozesse $P_1, \ldots, P_n$ ersetzt durch eine entsprechende, dem jeweiligen Prozeß zugeordnete Kopie $I_1, \ldots, I_n$. Dazu wird jeweils eine Umbenennung $f_j : A_j \to A_j \backslash I \cup I_j$ auf den jeweiligen Prozeß P_j angewendet mit

$$f_j(c.v) \stackrel{\text{def}}{=} \begin{cases} c.d & \text{falls } c \neq i \\ i_j.d & \text{falls } c = i \end{cases}$$

Aufbauend auf diesen Alphabeten und den angepaßten Prozessen wird ein in der Spezifikation nicht vorhandener Verteilterprozeß eingeführt. Dieser Prozeß verteilt die auf dem Eingabekanal I ankommenden Eingaben an die Prozesse auf die neu eingeführten Kanäle $I_1, \ldots, I_n$.

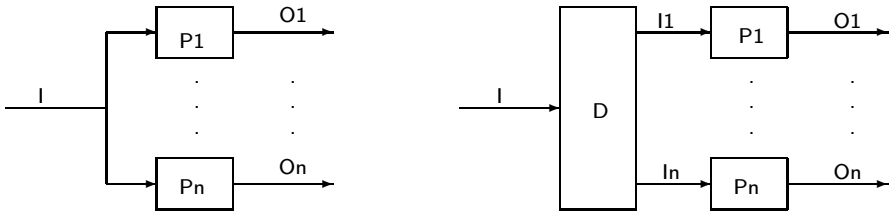

Abbildung 4.8: Komponente ohne und mit Verteilerkomponente

Eine mögliche Realisierung dieses Verteilerprozesses in occam® ist

```
CHAN OF DAT i:          -- Eingabekanal der Komponente
CHAN OF DAT h[n]:       -- Hilfskanaele fuer die n Prozesse
DAT data:               -- Zwischenspeicher fuer die Eingabe
WHILE TRUE              -- Der Prozess laeuft ohne vorzeitige Terminierung
  SEQ                   -- und fuehrt sequentiell aus
    i ? data            --  - Einlesen der Eingabe
    PAR k = 0 FOR n     --  - sowie fuer alle n Prozesse parallel
      h[k] ! data       --    Ausgeben auf dem Hilfskanal des Prozesses
```

Die hier gezeigte Realisierung gibt dabei die Eingabe parallel auf den Hilfskanälen aus. Dies entspricht direkt der Intuition sich verzweigender Ausgabekanäle: eine Nachricht steht an allen Enden des sich verzweigenden Ausgabekanals simultan zur Verfügung. Im hier vorgestellten Ansatz ist wegen der vorausgegangenen Bestimmung minimaler Anforderungen an die Eingabebereitschaft jedoch auch eine sequentielle Ausgabe ausreichend, da *alle* Empfängerprozesse zum Zeitpunkt der Ausgabe *gleichzeitig* eingabebereit sind.

Die vorgestellte Aufspaltung eines gemeinsamen Kanals in mehrere Kanäle verwendet einen separaten Verteilerprozeß. Der Verteilerprozeß muß jedoch nicht in jedem Fall explizit realisiert werden. Alternativ kann der Verteilerprozeß direkt in den Senderprozeß integriert werden. Dazu wird die Sendeaktion des Senders durch die entsprechende Anzahl von Sendeaktionen auf den neu eingeführten Kanälen ersetzt. Eine solche implizite Einführung einer Verteilerkomponente wird in Abschnitt 4.6 bei der Realisierung der Ausgabeprozesses verwendet.

4.6 Beispiel

Im folgenden Beispiel wird nochmals die gesamte Vorgehensweise bei der Bestimmung minimal puffernder Prozesse gezeigt,

- beginnend bei der relationalen Beschreibung des Systems und seiner Komponenten auf der Spurebene,
- über den Wechsel von der Spur- zur Failuredarstellung einschließlich der Abschwächung der Systemanforderungen sowie
- die Bestimmung der minimalen Anforderungen an die Eingabebereitschaft der Komponenten,
- bis hin zur kanalorientierten Realisierung des Systems.

Der Schwerpunkt liegt dabei auf der Bestimmung der Residuen hinsichtlich der Elimination der Parallelkomposition. Das Beispiel schließt mit einer occam®2-Realisierung einer Komponente.

Beispiel 4.6.1 (Multiplexer) Die Aufgabe des Multiplexer/Demultiplexersystems ist es, für zwei unidirektional und synchron kommunizierende Prozeßpaare *zwei* unidirektionale Kommunikationsverbindungen auf *einer* bidirektionalen Verbindung zu realisieren. Zur Realisierung sollen zwei Komponenten verwendet werden, auf der Senderseite ein Multiplexer M sowie auf der Empfängerseite ein Demultiplexer D. Dabei sind

- i1 und i2 die Bezeichner der Eingabekanäle des Multiplexers,
- o1 und o2 die Bezeichner der Ausgabekanäle des Demultiplexers
- m der Bezeichner des Multiplexkanals vom Multiplexer zum Demultiplexer und
- ack der Bezeichner des Quittungskanals vom Demultiplexer zum Multiplexer.

Abbildung 4.9 zeigt den Aufbau des Systems. Zur formalen Spezifikation werden die folgenden Alphabete verwendet:

Eingabekanäle i1 und i2: Die zugehörigen Alphabete $I_1 = \{\mathsf{i1}\} \times D$ und $I_2 = \{\mathsf{i2}\} \times D$ bestehen aus dem Kanalbezeichner sowie dem Datenanteil. $I = I_1 \cup I_2$ bezeichnet das gesamte Eingabealphabet.

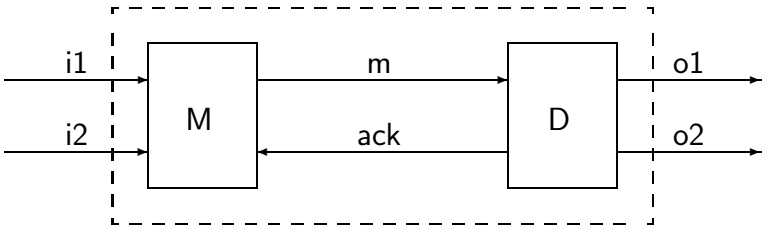

Abbildung 4.9: Das Multiplexer/Demultiplexersystem

Ausgabekanäle o1 und o2: Die zugehörigen Alphabete $O_1 = \{\mathsf{o1}\} \times D$ und $O_2 = \{\mathsf{o2}\} \times D$ bestehen aus dem Kanalbezeichner sowie dem Datenanteil. $O = O_1 \cup O_2$ bezeichnet das gesamte Ausgabealphabet.

Multiplexkanal m: Das zugehörige Alphabet $M = \{\mathsf{m}\} \times \{1, 2\} \times D$ besteht aus dem Kanalbezeichner, einem Identifikator für den Ausgabekanal sowie dem Datenanteil. Zusätzlich bezeichnen $M_1 = \{\mathsf{m}\} \times \{1\} \times D$ bzw. $M_2 = \{\mathsf{m}\} \times \{2\} \times D$ die für Kanal o1 bzw. Kanal o2 bestimmten Nachrichten auf m.

Quittungskanal ack: Das zugehörige Alphabet $Q = \{\mathsf{a}ck\} \times \{1, 2\}$ besteht aus dem Kanalbezeichner und einem Identifikator für den Ausgabekanal.

Weiterhin werden die einzelnen Alphabete zusammengefaßt zu

- $I_{\mathsf{M}} = I_1 \cup I_2$, $O_{\mathsf{M}} = M$, $A_{\mathsf{M}} = I_{\mathsf{M}} \cup O_{\mathsf{M}}$
- $I_{\mathsf{D}} = M$, $0_{\mathsf{D}} = O_1 \cup O_2$, $A_{\mathsf{D}} = I_{\mathsf{D}} \cup O_{\mathsf{D}}$

Unter Verwendung dieser Modellierung wurde bereits auf der Spurebene ein Protokoll spezifiziert, das das gewünschte Verhalten sicherstellt. Dazu wird vom Demultiplexer für jede zugestellte Nachricht eine Quittung über den Quittungskanal an den Multiplexer gesendet. Auf diesen Alphabeten wird entsprechend wie in Beispiel 2.2.3 die Funktion data^* definiert. Zusätzlich wird in diesem Beispiel die Funktion $\mathsf{id} : O \to \{1, 2\}$ verwendet mit

$$\begin{aligned} \mathsf{id}(\mathsf{o1}.d) &\stackrel{\text{def}}{=} 1 \\ \mathsf{id}(\mathsf{o2}.d) &\stackrel{\text{def}}{=} 2 \end{aligned}$$

Das Multiplexer/Demultiplexersystem soll eine Zweikanalverbindung von i1 nach o1 sowie von i2 nach o2 realisieren. Auf der Spurebene läßt sich dazu das Verhalten des Gesamtsystems relational durch zwei sequentielle und monotone Prozeßdarstellungen $(P_{\mathsf{o1}}, C_{\mathsf{o1}})$ bzw. $(P_{\mathsf{o2}}, C_{\mathsf{o2}})$ mit den Eingabealphabeten I_1 bzw. I_2 und den Ausgabealphabet O_1 und O_2 beschreiben, wobei

$$\begin{aligned} P_{\mathsf{o1}}(i, o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) \sqsubseteq \mathsf{data}^*(i) \\ C_{\mathsf{o1}}(i, o) &\stackrel{\text{def}}{=} \mathsf{data}^*(o) = \mathsf{data}^*(i) \end{aligned}$$

und P_{o2} bzw. C_{o2} entsprechend definiert sind.

Die Aufgabe des Multiplexers besteht darin, auf den Eingabekanälen i1 bzw. i2 Nachrichten entgegenzunehmen. Wurde alle Nachrichten von i1 bzw. i2 bisher vom Demultiplexer quittiert, so wird die nächste Nachricht mit einem Identifikator für den Zielkanal o1 bzw. o2 versehen und auf m übertragen. Die formale Definition des Verhaltens auf Spurebene erfolgt mit einer relationalen Darstellung. Dazu wird die Relation P_{M1} verwendet, die ausdrückt daß die auf m für Kanal o1 übermittelten Daten vorher auf i1 empfangen wurden, und auf m höchstens eine Nachricht mehr für Kanal o1 verschickt wurde als Quittungen dafür empfangen wurden:

$$P_{\mathsf{M1}}(i,o) \stackrel{\text{def}}{=} (\exists m \in M^{\omega}.\, \mathsf{data}^*(M_1 \copyright (o \circ m)) = \mathsf{data}^*(I_1 \copyright i)) \wedge \\ \#M_1 \copyright o \leq \#\mathsf{ack}.1 \copyright i + 1$$

Entsprechend wird C_{M1} verwendet, um die Übertragung aller bestätigten Daten auszudrücken:

$$C_{\mathsf{M1}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(M_1 \copyright o) = \mathsf{data}^*(I_1 \copyright i) \vee \#M_1 \copyright o = \#\mathsf{ack}.1 \copyright i + 1$$

Analog werden entsprechende Relationen für P_{M2} und C_{M2} definiert. Damit wird die monotone relationale Darstellung $(P_{\mathsf{M}}, C_{\mathsf{M}})$ des Multiplexers definiert mittels

$$P_{\mathsf{M}}(i,o) \stackrel{\text{def}}{=} P_{\mathsf{M1}}(i,o) \wedge P_{\mathsf{M2}}(i,o)$$
$$C_{\mathsf{M}}(i,o) \stackrel{\text{def}}{=} C_{\mathsf{M1}}(i,o) \wedge C_{\mathsf{M2}}(i,o)$$

Das Verhalten des Demultiplexers besteht darin, auf m die Nachrichten entgegenzunehmen, den Identifikator zu entfernen und die Ausgabe auf dem entsprechenden Kanal auszugeben. Auch der Demultiplexer wird relational beschrieben. Dazu wird die Relation P_{D1} verwendet, die beschreibt, daß auf o1 nur Nachrichten ausgegeben werden, die mit dem zugehörigen Identifikator übertragen wurden:

$$P_{\mathsf{D1}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(O_1 \copyright o) \sqsubseteq \mathsf{data}^*(M_1 \copyright i)$$

Entsprechend wird C_{D1} definiert, um die Ausgabe aller Nachrichten zu beschreiben:

$$C_{\mathsf{D1}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(O_1 \copyright o) = \mathsf{data}^*(M_1 \copyright i)$$

Analog werden entsprechende Relationen für P_{D2} und C_{D2} definiert. Schließlich wird die Relation P_{D0} verwendet, um zu beschreiben, daß nur für ausgegebene Nachrichten der Identifikator als Quittung an den Multiplexer geschickt wird:

$$P_{\mathsf{D0}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(o) \sqsubseteq \mathsf{id}^*(i)$$

C_{D0} drückt aus, daß alle Quittungen für ausgegebene Nachrichten zugestellt werden:

$$C_{\mathsf{D0}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(o) = \mathsf{id}^*(i)$$

Der Demultiplexer selbst wird durch drei Prozesse - ein Prozeß pro Ausgabekanal - beschrieben, um die gewünschte Parallelität sicherzustellen. Dabei ist $(P_{\mathsf{D1}}, C_{\mathsf{D1}})$ die monotone relationale Darstellung des Prozesses für den Ausgabekanal o1, $(P_{\mathsf{D2}}, C_{\mathsf{D2}})$ die für

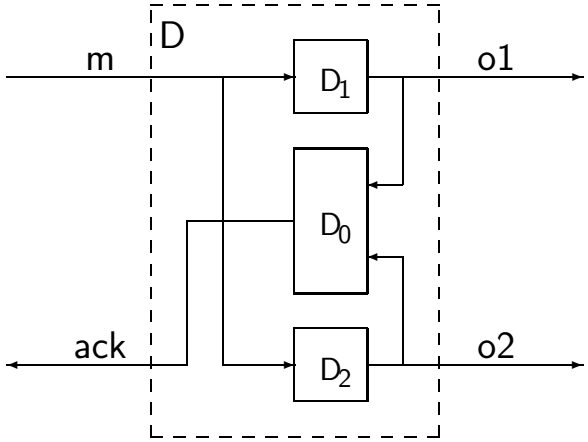

Abbildung 4.10: Prozeßstruktur des Demultiplexers

den Ausgabekanal o2 sowie $(P_{\mathsf{D0}}, C_{\mathsf{D0}})$ die für den Quittungskanal ack. Abbildung 4.10 zeigt die Prozeßstruktur des Demultiplexers.

Für die synchrone Realisierung ist es nun jedoch notwendig, die notwendige Eingabebereitschaft von Multiplexer und Demultiplexer zu ermitteln, die das gewünschte Verhalten des Gesamtsystem sicherstellt, in dem ein vorzeitiges Blockieren des Systems ausgeschlossen wird.

Für die Definition der implizierten Failurebeschreibung des Multiplexers werden die Hilfsrelation B_{M1} mit

$$B_{\mathsf{M1}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(M_1 \copyright o) \sqsubset \mathsf{data}^*(I_1 \copyright i) \wedge \#M_1 \copyright o < \#\mathsf{ack}.1 \copyright i + 1$$

zur Beschreibung der Sendebereitschaft sowie eine entsprechende Hilfsrelation B_{M2} verwendet. Damit lassen sich nun die Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft des Multiplexers entsprechend Definition 3.4.1 angeben unter Verwendung einfacher Umformungen:

$$\begin{array}{lcl} S_{\mathsf{M}}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.P_{\mathsf{M}}(In \copyright t, M \copyright t) \\ F^M_{\mathsf{M}}(t,R) & \stackrel{\text{def}}{=} & (B_{\mathsf{M1}}(In \copyright t, M \copyright t) \vee B_{\mathsf{M2}}(In \copyright t, M \copyright t)) \Rightarrow \\ & & \exists o \in M.o \notin R \wedge P_{\mathsf{M1}}(I \copyright t, (M \copyright t) \circ o) \wedge P_{\mathsf{M2}}(I \copyright t, (M \copyright t) \circ o) \end{array}$$

Für die Failurebeschreibung des Demultiplexers sind die implizierten Failurebeschreibungen der drei sequentiellen Prozesses des Demultiplexers zu bestimmen. Ähnlich wie im Fall des Multiplexers werden dazu Hilfsrelationen definiert. B_{D1} drückt dabei die Sendebereitschaft auf Kanal o1 aus:

$$B_{\mathsf{D1}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(O_1 \copyright o) \sqsubset \mathsf{data}^*(M_1 \copyright i)$$

Entsprechend wird B_{D2} für Kanal o2 definiert. Für den Ausgabekanal ack wird die Sendebereitschaft B_{D0} definiert als

$$B_{\mathsf{D0}}(i,o) \stackrel{\text{def}}{=} \mathsf{data}^*(o) \sqsubset \mathsf{id}^*(i)$$

Damit lassen sich nun die Charakterisierungen der partiellen Abläufe und der Ausgabebereitschaft der Prozesse angeben als

$$\begin{array}{lcl}
S_{\mathsf{D0}}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.P_{\mathsf{D0}}(O©t, Q©t) \\
S_{\mathsf{D1}}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.P_{\mathsf{D1}}(M©t, O_1©t) \\
S_{\mathsf{D2}}(t) & \stackrel{\text{def}}{=} & \forall s \sqsubseteq t.P_{\mathsf{D2}}(M©t, O_2©t) \\
F^{Q}_{\mathsf{D0}}(t,R) & \stackrel{\text{def}}{=} & B_{\mathsf{D0}}(O©t, Q©t) \Rightarrow \exists o \in Q.(o \notin R \wedge P_{\mathsf{D0}}(O©t, (Q©t) \circ o)) \\
F^{O_1}_{\mathsf{D1}}(t,R) & \stackrel{\text{def}}{=} & B_{\mathsf{D1}}(M©t, O_1©t) \Rightarrow \exists o \in O_1.(o \notin R \wedge P_{\mathsf{D1}}(M©t, (O_1©t) \circ o)) \\
F^{O_2}_{\mathsf{D2}}(t,R) & \stackrel{\text{def}}{=} & B_{\mathsf{D2}}(M©t, O_2©t) \Rightarrow \exists o \in O_2.(o \notin R \wedge P_{\mathsf{D2}}(M©t, (O_2©t) \circ o))
\end{array}$$

Die gesamte Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft des Demultiplexers ist die Parallelkomposition der drei sequentiellen Prozesse und damit

$$\begin{array}{lcl}
S_{\mathsf{D}}(t) & = & S_{\mathsf{D0}}(t) \wedge S_{\mathsf{D1}}(t) \wedge S_{\mathsf{D2}}(t) \\
F^{O_1 \cup O_2 \cup Q}_{\mathsf{D}}(t,R) & = & F^{Q}_{\mathsf{D0}}(t,R) \wedge F^{O_1}_{\mathsf{D1}}(t,R) \wedge F^{O_2}_{\mathsf{D2}}(t,R)
\end{array}$$

Zusätzlich erfüllt der Demultiplexer noch die Anforderungen an die Eingabebereitschaft $F^{M}_{P_{\mathsf{D1}},C_{\mathsf{D1}}}$, $F^{M}_{P_{\mathsf{D2}},C_{\mathsf{D2}}}$ und $F^{O}_{P_{\mathsf{D0}},C_{\mathsf{D0}}}$ seiner drei sequentiellen Prozesse. Da die ersten beiden identisch sind, wird im folgenden statt dessen die Bezeichnung $F^{M}_{P_{\mathsf{D}},C_{\mathsf{D}}}$ verwendet.

Da der Multiplexer/Demultiplexer beispielsweise die Eingabebereitschaft F^{I} und die Ausgabebereitschaft F^{O} mit

$$\begin{array}{lcl}
F^{I}(t,R) & \stackrel{\text{def}}{=} & (I_1©t \in I^* \Rightarrow I_1 \cap R = \emptyset) \wedge \\
 & & (I_2©t \in I^* \Rightarrow I_2 \cap R = \emptyset) \\
F^{O}(t,R) & \stackrel{\text{def}}{=} & (\mathsf{data}^*(O_1©t) \neq \mathsf{data}^*(I_1©t) \Rightarrow O_1 \not\subseteq R) \wedge \\
 & & (\mathsf{data}^*(O_2©t) \neq \mathsf{data}^*(I_2©t) \Rightarrow O_2 \not\subseteq R)
\end{array}$$

sicherstellt, eine unbeschränkte Eingabebereitschaft auf den Kanäle im synchronen Fall jedoch nicht notwendig ist, wird die Eingabebereitschaft abgeschwächt zu

$$\begin{array}{lcl}
F^{I_1,I_2} & \stackrel{\text{def}}{=} & ((I_1©t \in I^* \wedge I_1©t = O_1©t) \Rightarrow I_1 \cap R = \emptyset) \wedge \\
 & & ((I_2©t \in I^* \wedge I_2©t = O_2©t) \Rightarrow I_2 \cap R = \emptyset)
\end{array}$$

Die Eingabebereitschaft auf I_1 und I_2 wird also nur für die Fälle gefordert, bei denen die Umgebung zusätzliche die Anforderungen

$$U_1(t) \stackrel{\text{def}}{=} I_1©t = O_1©t$$

bzw.

$$U_2(t) \stackrel{\text{def}}{=} I_2©t = O_2©t$$

erfüllt. Die Eingabebereitschaft wird also jeweils nur dann gefordert, wenn alle bisherigen Nachrichten zugestellt wurden. Da das Multiplexer/Demultiplexersystem zwei unabhängige Kommunikationverbindungen sicherstellen soll, soll der Grad der Parallelität der Ausgabebereitschaft nicht eingeschränkt werden. Die Anforderung $F^{O}(t,R)$ wird bei der Abschwächung unverändert übernommen.

Im folgenden soll nun gezeigt werden, wie die für M geforderte Eingabebereitschaft des Demultiplexers abgeschwächt werden kann. Entsprechend der Definition der Eingabebereitschaft bei aktionsweiser Parallelität wurde hierfür

$$F^{M}_{(P_{\mathsf{D}},C_{\mathsf{D}})}(t,R) \stackrel{\text{def}}{=} M \cap t \in M^* \Rightarrow M \cap R = \emptyset$$

gefordert. Statt dieser starken Forderung soll nun bestimmt werden, welche minimalen Anforderungen an die Eingabebereitschaft des Demultiplexers auf der Alphabetmenge M erhoben werden müssen, um die Ausgabebereitschaft des Gesamtsystems sicherzustellen. Dazu ist also das Residuum für $F^{O}(t,R)$ zu bestimmen. Im folgenden wird nur das Residuum für die Anforderung

$$\mathsf{data}^*(O_1 © t) \sqsubset \mathsf{data}^*(I_1 © t) \Rightarrow O_1 \not\subseteq R$$

ermittelt. Die Bestimmung für die zweite Anforderung verläuft analog. Entsprechend dem Aufbau des Systems und der in 4.3 beschriebenen Schritte wird zuerst das Residuum hinsichtlich der Abstraktion von $M \cup Q$ bestimmt. Dies liefert unter Verwendung von Hilfsatz 4.4.1 als Anforderung an die verbleibende Parallelkomposition von M und D

$$\mathsf{data}^*(O_1 © t) \sqsubset \mathsf{data}^*(I_1 © t) \Rightarrow O_1 \cup M \cup Q \not\subseteq R$$

Die Elimination der Parallelkomposition von $F_{\mathsf{M}}(t,R)$ liefert bei Verwendung der in Abschnitt 4.4.2 beschriebenen Darstellung aus Satz 4.4.2 – der Einfachheit halber ohne den Anteil $\forall t.A_2 © t = s$ –

$$\begin{aligned}\forall R_1 \subseteq A_{\mathsf{M}}.\ & S_{\mathsf{M}}(A_{\mathsf{M}} © t) \wedge F^{I1,I2,Q}_{\mathsf{M}}(A_{\mathsf{M}} © t, R_1) \wedge F^{M}_{\mathsf{M}}(A_{\mathsf{M}} © t, R_1) \wedge \\ & \mathsf{data}^*(O_1 © t) \sqsubset \mathsf{data}^*(I_1 © t) \Rightarrow \\ & O_1 \cup M \cup Q \not\subseteq R_1 \cup R_2\end{aligned}$$

Da nur die Eingabebereitschaft des Demultiplexers minimiert, die Charakterisierung der partiellen Abläufe und der Ausgabebereitschaft jedoch beibehalten werden soll, wird zusätzlich noch die Konjunktion dieser Charakterisierung entsprechend des in Absatz 4.3.2 in Form von Definition 4.3.1 beschriebenen Schemas eliminiert, und es bleibt als Anforderung

$$\begin{aligned}\forall R_1 \subseteq A_{\mathsf{M}}.\ & S_{\mathsf{M}}(A_{\mathsf{M}} © t) \wedge F^{I1,I2,Q}_{\mathsf{M}}(A_{\mathsf{M}} © t, R_1) \wedge F^{M}_{\mathsf{M}}(A_{\mathsf{M}} © t, R_2) \wedge \\ & S_{\mathsf{D}}(A_{\mathsf{D}} © t) \wedge F^{O}_{P_{\mathsf{D}0},C_{\mathsf{D}0}}(A_{\mathsf{D}} © t, R_2) \wedge F^{O_1 \cup O_2 \cup Q}_{\mathsf{D}}(A_{\mathsf{D}} © t, R_2) \wedge \\ & \mathsf{data}^*(O_1 © t) \neq \mathsf{data}^*(I_1 © t) \Rightarrow \\ & O_1 \cup M \cup Q \not\subseteq R_1 \cup R_2\end{aligned}$$

Einfache äquivalente prädikatenlogische Umformungen unter Einsatz der Definitionen der definierten Operatoren führen zu Darstellung des Residuums von D ohne Refusalanteil:

$$\begin{aligned}& S_{\mathsf{M}}(A_{\mathsf{M}} © t) \wedge \\ & S_{\mathsf{D}}(A_{\mathsf{D}} © t) \wedge F^{O}_{P_{\mathsf{D}0},C_{\mathsf{D}0}}(A_{\mathsf{D}} © t, R_2) \wedge F^{O_1 \cup O_2 \cup Q}_{\mathsf{D}}(A_{\mathsf{D}} © t, R_2) \wedge \\ & \mathsf{data}^*(M_1 © t) \sqsubset \mathsf{data}^*(I_1 © t) \wedge \\ & \mathsf{data}^*(Q © t) = \mathsf{id}^*(O © t) \wedge \mathsf{data}^*(M_1 © t) = \mathsf{data}^*(O_1 © t) \wedge \\ & P_{\mathsf{M1}}(I © t, (M © t) \circ o) \wedge P_{\mathsf{M2}}(I © t, (M © t) \circ o) \wedge \\ & o \in M_1 \Rightarrow \\ & o \notin R_2\end{aligned}$$

Für eine explizite Darstellung des Residuums ist also noch der alphabetsfremde Anteil in

$$\begin{array}{l} S_{\mathsf{M}}(A_{\mathsf{M}} \copyright t) \wedge \\ \mathsf{data}^*(M_1 \copyright t) \sqsubset \mathsf{data}^*(I_1 \copyright t) \wedge \\ P_{\mathsf{M1}}(I \copyright t, (M \copyright t) \circ o) \wedge P_{\mathsf{M2}}(I \copyright t, (M \copyright t) \circ o) \end{array} \tag{4.12}$$

zu entfernen. Dies geschieht mittels äquivalenter Umformungen unter Einsatz der Definitionen der verwendeten Prädikate und Funktionen. Als Ergebnis bleibt

$$M \copyright t \in M^*$$

und mittels einfacher Umformungen bleibt damit als explizite Anforderung

$$\begin{array}{l} S_{\mathsf{D}}(A_{\mathsf{D}} \copyright t) \wedge F^{O}_{P_{\mathsf{D0}},C_{\mathsf{D0}}}(A_{\mathsf{D}} \copyright t, R) \wedge F^{O_1 \cup O_2 \cup Q}_{\mathsf{D}}(A_{\mathsf{D}} \copyright t, R) \wedge \\ \mathsf{data}^*(Q \copyright t) = \mathsf{id}^*(O \copyright t) \wedge \mathsf{data}^*(M_1 \copyright t) = \mathsf{data}^*(O_1 \copyright t) \wedge M \copyright t \in M^* \wedge \\ o \in M_1 \Rightarrow \\ \quad o \notin R \end{array}$$

Werden schließlich noch die Charakterisierungen der partiellen Abläufe und der Ausgabebereitschaft sowie der Eingabeanforderung $F^{O}_{P_{\mathsf{D0}},C_{\mathsf{D0}}}$ entfernt, bleibt als minimale Anforderung an die Eingabebereitschaft für M

$$\begin{array}{l} \mathsf{data}^*(Q \copyright t) = \mathsf{id}^*(O \copyright t) \wedge \mathsf{data}^*(M_1 \copyright t) = \mathsf{data}^*(O_1 \copyright t) \wedge M \copyright t \in M^* \wedge o \in M_1 \Rightarrow \\ \quad o \notin R \end{array}$$

Damit muß der Demultiplexer immer auf M_1 dann eingabebereit sein, wenn

- für jede auf o1 oder o2 ausgegebene Nachricht eine entsprechende Quittung auf ack verwendet wurde ($\mathsf{data}^*(Q \copyright t) = \mathsf{id}^*(O \copyright t)$)
- jede auf m empfangene Nachricht für o1 ausgegeben wurde ($\mathsf{data}^*(M_1 \copyright t) = \mathsf{data}^*(O_1 \copyright t)$)
- bisher nur endlich viele Nachrichten auf m entgegengenommen wurden ($M \copyright t \in M^*$)

Da m als *ein* Eingabekanal realisiert werden soll und dazu die in Definition 4.5.1 beschriebene Eigenschaft gelten muß, wird die Anforderung von $o \in M_1$ auf $o \in M$ ausgedehnt. Die entsprechende Bestimmung der minimalen Eingabebereitschaft für die Ausgabebereitschaft auf O_2 liefert analog

$$\begin{array}{l} \mathsf{data}^*(Q \copyright t) = \mathsf{id}^*(O \copyright t) \wedge \mathsf{data}^*(M_2 \copyright t) = \mathsf{data}^*(O_2 \copyright t) \wedge M \copyright t \in M^* \wedge o \in M \Rightarrow \\ \quad o \notin R \end{array}$$

Damit bleibt insgesamt als Anforderung an die Eingabebereitschaft auf M

$$\begin{array}{l} \mathsf{data}^*(Q \copyright t) = \mathsf{id}^*(O \copyright t) \wedge \\ (\mathsf{data}^*(M_1 \copyright t) = \mathsf{data}^*(O_1 \copyright t) \vee \mathsf{data}^*(M_2 \copyright t) = \mathsf{data}^*(O_2 \copyright t)) \wedge M \copyright t \in M^* \Rightarrow \\ \quad R \cap M = \emptyset \end{array} \tag{4.13}$$

Für eine effiziente Realisierung des Multiplexers ist weiterhin die Anforderung an die Eingabebereitschaft von O durch $F^{O}_{P_{\mathsf{D0}},C_{\mathsf{D0}}}$ einzuschränken. Auch diese Bestimmung der minimalen Eingabebereitschaft verläuft analog und wird daher nicht gezeigt.

Damit sind die minimalen Anforderungen an die Eingabebereitschaft der Demultiplexerprozesse bestimmt. Für eine weitere kanalorientierte Realisierung sind noch, wie in Abschnitt 4.5 beschrieben, die die Kanäle festzulegen, sowie - falls nötig - Verteilerprozesse einzuführen. Da die Alphabetsmengen Q, O_1 und O_2 Ausgabealphabete sequentieller Prozesses sind, lassen sich diese Prozesse, wie in Abschnitt 4.5.2 beschrieben, durch einen Kanal realisieren. Entsprechend der Anforderung 4.13 an die Eingabebereitschaft der Ausgabeprozesse D1 und D2 folgt weiterhin, daß deren Eingabealphabet M die in Definition 4.5.1 geforderte Eigenschaft 4.11 erfüllt. Damit wird auch M sequentiell genutzt. Damit bleibt noch die Aufgabe, entsprechend der in Abschnitt 4.5.3 beschriebene Vorgehensweise die explizite Nutzung von Kanälen sicherzustellen.

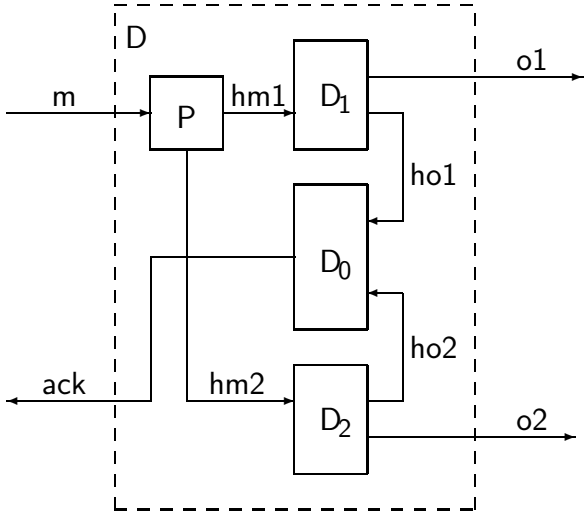

Abbildung 4.11: Kanalorientierte Realisierung des Demultiplexers

Abbildung 4.11 zeigt die kanalorientierte Struktur, die sich durch das Einfügen der zusätzlichen Kanäle sowie des Verteilers P ergibt. Für die Aufspaltung des Kanals m in die beide Eingabekanäle hm1 und hm2 der Ausgabeprozesse D1 und D2 wird ein expliziter Verteilter P eingeführt. Für die Aufspaltung der Ausgabe o1 und o2 für die Rückkopplung zum Quittungsprozeß D0 mittels ho1 und ho2 werden keine expliziten Verteilerprozesse eingeführt. Statt dessen wird die in Abschnitt 4.5 beschriebene Möglichkeit der Realisierung mittels Anpassung der Ausgabeaktion der Sendeprozesse verwendet.

Eine mögliche Realisierung des Demultiplexers in occam® 2 zeigt der folgende Programmtext.[8] Wie bereits in der Spurspezifikation erfaßt, werden die Ausgabekanäle des Demultiplexers durch drei parallel ablaufende Prozesse bedient. Zusätzlich wird noch ein vierter Prozeß zur Pufferung benötigt, der die Eingaben in den Pufferelementen `CHAN OF DAT ho[2]` zwischenspeichert. Aus Synchronisierungsgründen muß dazu - ebenso wie für den Quittungspuffer `CHAN OF INT ha[2]` - in occam® 2 ein Kanal verwendet werden.

[8]Für die Beschreibung von occam® 2 siehe [JG88].

Der Quittungsprozeß ist, im Gegensatz zu den anderen Prozessen des Demultiplexers, ein nichtmonotoner, unterbrechbarer Prozeß. Zu seiner Realisierung in TCSP wird daher der in Abschnitt A.5 beschriebene Unterbrechungsoperator verwendet:

$$
\begin{aligned}
H1 &\stackrel{\text{def}}{=} ho1?h \rightarrow ack!h \rightarrow H1 \\
H12 &\stackrel{\text{def}}{=} H1 \triangle (ho2?h \rightarrow ack!h \rightarrow H21) \\
H2 &\stackrel{\text{def}}{=} ho2?h \rightarrow ack!h \rightarrow H2 \\
H21 &\stackrel{\text{def}}{=} H2 \triangle (ho1?h \rightarrow ack!h \rightarrow H12) \\
H &\stackrel{\text{def}}{=} H1
\end{aligned}
$$

Zur Realisierung des Quittungsprozesses in occam®2 wird ein Quittungsmultiplexer mittels des priorisierten Auswahloperators (PRI ALT) verwendet.[9] Eine mögliche Realisierung des gesamten Demultiplexers in occam®2 lautet

```
PROTOCOL MUX INT; DATA          -- Multiplexnachrichtenpakete
PROTOCOL DAT DATA               -- Ausgabenachrichtenpakete
PROTOCOL ACK INT                -- Quittungsnachrichtenpakete
CHAN OF MUX m:                  -- Der Multiplexkanal
CHAN OF DAT o[2]:               -- Die Ausgabekanaele
CHAN OF ACK ack:                -- Der Quittungskanal
CHAN OF MUX hm[2]:              -- Hilfskanaele fuer die Nachrichenpaktete
CHAN OF INT ha[2]:              -- Mulitplexer fuer Quittung
PAR                             -- Drei Arten von Prozessen:
  WHILE TRUE                    -- 1. Der nichtterminierende Verteilter
    SEQ                         --    der sequentielle ablaeuft,
      DATA data:                --    einen Puffer fuer Paketdaten und
      INT to:                   --    einen Puffer fuer Paketrichtung hat,
      m ? data ; to             --    ein Paket von m liest und
      PAR n = 0 FOR 2           --    parallel fuer jeden Empfaenger
      hm[n] ! to ; data         --    das Paket dem Empfaenger zustellt
  PAR n = 0 FOR 2               -- 2. Pro Ausgabekanal ein Prozess,
    WHILE TRUE                  --    der nicht terminiert,
      SEQ                       --    sequentiell ablaeuft,
        DATA data:              --    einen Puffer fuer Paketdaten und
        INT to:                 --    einen Puffer fuer Paketrichtung hat,
        hm[n] ? data ; to       --    ein Paket vom Hilfskanal liest, und,
        IF                      --    und, falls
          n = to                --    die Daten fuer ihn bestimmt sind,
            o[n] ! data         --    auf den Ausgabekanal schreibt und
            ho[n] ! n           --    die Ausgabe quittiert.
  WHILE TRUE                    -- 3. Der Quittungsmultiplexer
```

[9] occam®2 besitzt kein explizites Unterbrechungskonstrukt; die Verwendung von PRI ALT führt jedoch zum gleichen Verhalten.

```
SEQ n = 0 FOR 2           --    der mittels Round-Robin ueber n
  INT h:                  --    unter Verwendung der Variablen h
  PRI ALT                 --    fuer jeden Prozess priorisiert
    ho[n \ 2] ? h         --    vom ersten Prozess die Quittung liest
    ho[(n + 1) \ 2] ? h   --    oder vom naechstpriorisierten
  ack ! h                 --    und die Quittung zustellt
```

Für die vollständige synchrone Realisierung des Multiplexer/Demultiplexersystems muß analog auch eine Realisierung des Multiplexers entwickelt werden. Eine entsprechende Realisierung sowie eine ausführliche Erläuterung eines vergleichbaren synchronen Multiplexer/Demultiplexersystems findet sich in [JG88].[10] ◇

4.7 Zusammenfassung

Das Ziel der in dieser Arbeit beschriebenen Vorgehensweise ist das gezielte Abschwächen von Beschreibungen asynchronen Verhaltens von Komponenten eines Systems hin zu synchronem Verhalten. In diesem Kapitel wurde dazu der Begriff des Residuums für die Kompositionsoperatoren zur Beschreibung von aus Komponenten zusammengesetzten Systemen eingeführt. Damit wurde ein methodisches Vorgehen für die Bestimmung von Anforderungen an einzelne Komponenten eines System definiert, orientiert am syntaktischen Aufbau des Systems.

Für die Bestimmung der minimalen Anforderungen an die Komponenten wurden dazu direkte Darstellungen für die Residuen hinsichtlich der Parallelkomposition und der Abstraktion definiert. Während bei den anderen Schritten des vorgestellten Entwicklungsverfahrens eine Mechanisierung einfach möglich ist, ist die Bestimmung einer expliziten Darstellung des Residuums nicht unmittelbar automatisierbar. Hier liegt es hier nahe, geeignete allgemeine Schemata in Form von Hilfsätzen für die verschiedenen Konstruktionsoperatoren wie die punktweise Anwendung oder die Projektionsfunktion © zu erarbeiten, um so den Bestimmungsprozeß stärker zu mechanisieren.

Schließlich wurde ein methodisches Vorgehen für die Entwicklung kanalorientierter Systembeschreibungen eingeführt und so ein Anschluß an Verfahren geschaffen, die die Entwicklung programmiersprachlicher Beschreibungen kanalorientierter verteilter Systeme ausgehend von einer verhaltensmäßigen Darstellung unterstützen.

[10] Dort wird eine Variante gezeigt, bei der der Verteilerprozeß bereits das Ausfiltern von Paketen je nach Emfänger vornimmt.

Kapitel 5

Zusammenfassung und Ausblick

Program development is often inspired by flashes of intuition. The intuition may be difficult to articulate because it may be based on experience, analogy and insight. Formalism complements, but does not supplant, intuition. For example, intuition may lead us to suggest the introduction of concepts such as forks, bottles, and tokens during program design. A role of formalism is to support intuition by representing these concepts in formal terms. Although we expect formalism to support intuition, we do not expect intuition to support formalism. Once we have modeled our intuition in formals terms, such as force = mass × acceleration*, we expect to use symbol manipulation to deduce that* acceleration = force / mass*; no intuitve support is required to justify this step. These manipulations are second nature to us.*

M. Chandy, J. Misra, [CM89]

Bei der Entwicklung paralleler, mittels Nachrichtenaustausch kommunizierender Systeme wird, besonders in anwendungsorientierten Ansätzen, die asynchrone Systemsicht gegenüber der synchronen als die natürlichere und wünschenswerte Modellierungsebene angesehen. Damit stellt sich die Frage, ob die asynchrone Systemsicht auch für die Entwicklung synchroner Systeme als Ausgangspunkt verwendet werden kann. Die Arbeit zeigt, daß die asynchrone Systemsicht tatsächlich als abstraktere Sichtweise für die Entwicklung synchron kommunizierender Systeme eingesetzt werden kann, und führt dazu einen *zweistufigen methodischen Entwicklungsprozeß* ausgehend von einer asynchronen Systembeschreibung ein.

In diesem Kapitel wird nun der in dieser Arbeit beschriebene Ansatz zusammengefaßt

und hinsichtlich seiner Vorteile für den Entwicklungsprozeß synchroner Systeme bewertet. Abschließend werden noch einige Fragen und Anregungen vorgestellt, die sich aufbauend auf die Arbeit ergeben, jedoch über die ursprüngliche Aufgabenstellung hinausgehen.

5.1 Zusammenfassung

Um die Tauglichkeit der in dieser Arbeit vorgestellten Vorgehensweise zu beurteilen, wurden in Kapitel 1 die folgenden drei Fragen aufgeworfen:

- Kann die asynchrone Sichtweise von Systemen als Abstraktion synchron kommunizierender Systeme aufgefaßt werden?
- Läßt sich diese Abstraktion methodisch für die Modularisierung des Entwicklungsprozesses ausnutzen?
- Ist eine solche *zweistufige* Vorgehensweise vorteilhaft für den Entwicklungsprozeß synchron kommunizierender Systeme?

Die Ergebnisse der Arbeit zeigen, daß die asynchrone Sichtweise als abstrakte Beschreibung synchroner Systeme genutzt, und diese Abstraktion methodisch für eine Modularisierung des Entwicklungsprozesses eingesetzt werden kann. Diese Modularisierung erlaubt eine Aufspaltung des Entwicklungsprozesses in aufeinander aufbauende Teile mit schematischen Übergängen und unterstützt so eine gezielte Systementwicklung synchron kommunizierender Systeme.

Die Frage, ob asynchron kommunizierende Systeme als Abstraktion synchron kommunizierender Systeme gesehen werden können, wurde in Abschnitt 3.6 ausführlich diskutiert. Die Sinnfälligkeit dieser Abstraktionsform wurde dabei mit der erhöhten Komplexität der synchronen Modellierung und der dabei zusätzlich auftretenden Aspekte intuitiv belegt. Im Gegensatz zu anderen Abstraktionskonzepten wie beispielsweise der Strukturverfeinerung stellt die Abstraktion von Synchronisierungsaspekten aber keine formale Verfeinerungsbeziehung im Sinne der Implikation dar. Mit der eingeführten formalen Beziehung zwischen der asynchron und der synchron orientierten Sichtweise mittels der in 3.7 beschriebenen Vergröberung konnte jedoch die Voraussetzung für den Einsatz der Asynchronität als methodische Abstraktionsform geschaffen werden.

Um die Frage nach der methodischen Anwendung dieser Abstraktionsform zu beantworten, wurde ein zweistufiges Verfahren eingeführt, das zur gezielten Herleitung von Anforderungen an Komponenten eines synchron kommunizierenden Systems eingesetzt werden kann. Wesentlich für diese methodische Vorgehensweise ist die Aufspaltung des Systementwicklungsprozesses in zwei Phasen, nämlich

- Entwicklung der abstrakteren, asynchron orientierten Systemsicht mittels *trial and error* im einfacheren mathematischen Modell der Spursemantik sowie
- Entwicklung der konkreteren, synchron orientierten Systemsicht mittels gezielter Bestimmung im komplexeren mathematischen Modell der Failuresemantik.

Zusätzlich wurden für beide Phasen für die Vorgehensweise und die jeweilige Phase typische Aufgabenstellungen bestimmt, beispielsweise die Auswahl geeigneter Formen der Spurbeschreibungen oder die Bestimmung von Residuen, sowie schematische Lösungsansätze entwickelt.

Der Nachweis, daß die hier eingeführte Vorgehensweise für den Entwicklungsprozeß synchron kommunizierender Systeme als vorteilhaft angesehen werden kann, läßt sich im Rahmen dieser Arbeit nur bedingt erbringen. Der in der Vorgehensweise verwendete elegante mathematische Apparat bestehend aus den kompakten Modellen mit ihren einfachen Regeln, beispielweise für die Parallelkomposition und die Abstraktion, bietet eine geeignete Basis für den praktikablen Einsatz der eingeführten Vorgehensweise. Dies wird unterstützt durch die oben angesprochenen methodischen Aspekte, also den schematischen Modellwechsel und die schematisierte, schrittweise Herleitung von Residuen sowie der klaren Trennung von Daten- und Kontrollaspekten. Insgesamt ermöglicht dies eine stärker zielgerichtete, weniger fehleranfällige und damit effizientere Vorgehensweise als die direkte Entwicklung synchron kommunizierender Systeme, bei denen die Behandlung des Datenflusses im Vordergrund steht. Wie bei allen Entwicklungsansätzen kann dies letztendlich jedoch nur durch eine ausreichend große Menge von Fallstudien belegt werden.

Um die in dieser Arbeit beschriebene Vorgehensweise zu ermöglichen, wurde über die genannten Punkte hinaus die unendliche Failuresemantik zur Modellierung synchron kommunizierender Systeme eingeführt. In dieser Zusammenfassung wird jedoch nicht auf die unendliche Failuresemantik eingegangen. Die in Anhang A besprochenen und in Anhang B gezeigten Eigenschaften der unendlichen Failuresemantik stellen - über den Rahmen der Arbeit hinaus - einen eigenständigen Beitrag zur Modellierung synchroner Systeme dar. Daher werden auch die durch das neu eingeführte Modell gewonnenen Ergebnisse und die möglichen fortführenden Arbeiten gesondert in Abschnitt A.6 besprochen.

5.2 Ausblick

Die oben aufgeworfenen Fragen hinsichtlich eines methodischen Übergangs zwischen Systemen basierend auf asynchroner bzw. synchroner Kommunikation konnten im Rahmen dieser Arbeit durch die Einführung einer geeigneten Entwicklungsmethode zufriedenstellend beantwortet werden. Darüber hinaus stellen sich nun im Anschluß verschiedene, darauf aufbauende Fragen, die in dieser Arbeit nicht angesprochen bzw. nicht erschöpfend beantwortet wurden. Abschließend werden drei mögliche weitere Fragestellungen andiskutiert und damit Ansatzpunkte für weiterführende Arbeiten aufgezeigt.

5.2.1 Dynamische Netze

Die in dieser Arbeit vorgestellte Methode des systematischen Übergangs von asynchronen zu synchronen Systemen ist auf Systeme mit statischer Struktur beschränkt, da diese

Vorgehensweise auf die Entwicklung sicherer und korrekter Systeme mit endlichen Speicherressourcen abzielt. In den letzten Jahren hat sich jedoch die Aufmerksamkeit auch Systemen mit dynamischer Struktur zugewandt, zuerst in Form des π-Kalküls ([MPW92a], [MPW92b]), später auch in anderen Arbeiten (z.B. [GS95]). Hier ist es ohne Einschränkung möglich, neue Prozesse zu generieren und neue Kommunikationsverbindungen zu schaffen.

Damit stellt sich die Frage, ob die vorgestellte Vorgehensweise auch geeignet ist, um Systeme mit dynamischer Struktur zu behandeln. Da jedoch das Anwendungsgebiet dieser Systeme in erster Linie im Bereich mobiler Systeme und damit meist stark entkoppelter Systemkomponenten liegt, ist dort eine Realisierung basierend auf synchroner Kommunikation im allgemeinen nicht notwendig. Um die vorgestellte Vorgehensweise in anderen Anwendungsgebieten dynamischer Systeme einzusetzen, sind insbesondere die verwendeten semantischen Modelle zu erweitern bzw. deren Tauglichkeit zur Modellierung solcher Systeme nachzuweisen.

5.2.2 Parallelitätsgrad

Das Ergebnis dieser Arbeit ist eine durchgängige Methode, um ausgehend von asynchronen Systembeschreibungen synchron kommunizierende Systeme zu entwickeln, die trotz minimaler Anforderung an die Pufferung von Nachrichten keinen Sende-"Deadlock" verursachen. Mit einer optimalen Speichereffizienz, also in diesem Fall der minimalen Pufferkapazität, ist aber im Regelfall eine eingeschränkte Zeiteffizienz durch die Beschränkung der parallel stattfindenden Berechnungen verbunden; ein eventuell mögliches Vorausberechnen durch die Systemkomponenten ist mangels Pufferkapazität nicht immer möglich. Falls aber neben den qualitativen Aussagen wie der Robustheit des Systems auch quantitative Aussagen wie minimale Berechnungszeiten behandelt werden sollen, müssen zur Beschreibung solcher Systeme gezeitete Modelle verwendet werden (z.B. [RR88]). Hier ist zu untersuchen, ob die vorgestellte Vorgehensweise auch auf solche Modelle ausgeweitet werden kann.

5.2.3 Maschinelle Unterstützung

Wie in den Abschnitten 1, 4.3 und 4.4 wiederholt betont, ist die Arbeit wesentlich auf eine methodische und vorzugsweise schematische Behandlung des Übergangs von asynchronen zu synchronen Systemen ausgerichtet. Dies legt die Frage nahe, inwieweit dieser Prozeß maschinell unterstützt werden kann.

Auf der einen Seite sind Schritte wie die Transformation der Spureigenschaften in Failureeigenschaften oder die Bestimmung des Residuums hinsichtlich der Abstraktion bereits schematisiert und lassen sich einfach maschinell unterstützen. Auf der anderen Seite ist beispielsweise die Bestimmung des Residuums hinsichtlich der Parallelkomposition auch mit den vorgegebenen Richtlinien unter Umständen sehr aufwendig. Gerade hier kann ein beweisunterstützendes Werkzeug den Bestimmungsprozeß wesentlich erleichtern. Damit

empfiehlt es sich, beispielsweise das Beweissystem *Isabelle* ([Pau94c]) für die Unterstützung der Residuenbestimmung einzusetzen, insbesondere in Kombination mit der Objektlogik HOLCF ([Reg94]) und der zugehörigen FOCUS-Theorie ([San96]), um so die schematischen Anteile der methodischen Vorgehensweise weiter zu mechanisieren.

Anhang A

Unendliche Failuresemantik

Bereits im Kapitel 3 wurde die Unzulänglichkeit des in Abschnitt 2.3 vorgestellten endlichen Failuremodells angesprochen. Zur Behebung des Problems wurde im Zuge dieser Arbeit das unendliche Failuremodell als Erweiterung des endlichen Failuremodells zusammen mit zwei Operatoren, der Parallelkomposition und der Abstraktion, eingeführt und als Grundlage der entwickelten Vorgehensweise verwendet.

Die in dieser Arbeit vorgestellte Vorgehensweise konzentriert auf Prozeßspezifikationen mittels Verhaltensbeschreibung. Die letztendliche Realisierung dieser Beschreibungen durch programmiersprachliche Konstrukte steht - wie in Abbildung 1.1 in Abschnitt 1.3 skizziert - dabei im Hintergrund. Prinzipiell ist daher für diese Arbeit ausreichend, das unendliche Failuremodell einzuführen, ohne einen Bezug zu geeigneten programmiersprachlichen Konstrukten herzustellen. Für die Eignung des Modells für eine vollständige Vorgehensweise bis hin zur Realisierung ist jedoch die Tauglichkeit des Modells auch für diesen Schritt nachzuweisen. Damit ergeben sich die folgende Fragestellungen:

- Läßt sich eine denotationelle Semantik für die in dieser Arbeit betrachteten Systeme definieren?
- Ist diese denotationelle Semantik mit der denotationellen Semantik des endlichen Failuremodells verträglich ist?

Ziel dieses Kapitels ist es daher, eine für die in dieser Arbeit vorgestellte Vorgehensweise geeignete denotationelle Semantik synchron kommunizierender Systeme auf der Basis des unendlichen Failuremodells einzuführen. Eine erschöpfende Behandlung der Semantik im Umfang von TCSP stellt eine eigenständige Arbeit dar und würde daher den Rahmen eines Anhangs sprengen. Die Ziele dieses Kapitels sind daher im einzelnen:

- Die Erläuterung der Unzulänglichkeit der endlichen Failuresemantik sowie der Failure/Divergence-Semantik im Rahmen dieser Arbeit
- Die Einführung der zur endlichen Failuresemantik verträglichen Operationen für die unendliche Failuresemantik
- Die Illustration der Unterschiede zwischen endlicher und unendlicher Failuresemantik

- Der Nachweis der Wohldefiniertheit der unendlichen Failuresemantik für die in dieser Arbeit benötigten Prozesse bzw. Prozeßterme
- Eine kurze Diskussion der Eigenschaften der unendlichen Failuresemantik

In den Abschnitten A.2.1 und A.2.2 werden zunächst sowohl die endliche Failuresemantik als auch die Failure-Divergence-Semantik kurz umrissen. In Abschnitt A.3 wird dann die unendliche Failuresemantik als eine einfache Erweiterung der endlichen Failuresemantik namens eingeführt, und deren Unterschiede zur ursprünglichen Failuresemantik mittels zweier Beispiele in Abschnitt A.4 herausgestrichen. Weiterhin wird die Beziehung zu den Begriffen *Fairness*, *Interleaving Semantiken* und *echte Nebenläufigkeit* hergestellt. In Abschnitt A.5 wird gezeigt, wie sich der Sprachumfang erweitern läßt, um schwache Fairness und unbeschränkten Nichtdeterminismus zu modellieren. Abschnitt A.6 beschließt diesen Teil mit einer kurzen Bewertung.

A.1 Einleitung

Seit ihrer Einführung (siehe z.B. [Hoa85]) ist die Failure-Divergence-Semantik ein erfolgreiches Modell für synchron kommunizierende Prozesse geworden. Darüberhinaus wurden viele Erweiterungen der ursprünglichen Semantik für "Communicating Sequential Processes" (CSP) eingeführt:

- Timed Communicating Sequential Processes (z.B. [DS89])
- Probabilistic Communicating Sequential Processes (z.B. [Low93])
- Erweiterungen um unbeschränkten Nichtdeterminismus (z.B. [Ros88], [KP94])

und viele andere. Diese Semantiken spiegeln die Ansicht wider, daß Divergenz ein wesentlicher Teil der Modellierung synchron kommunizierender Systeme ist. Jedoch weist die Failure-Divergence-Semantik gewisse Schwächen auf, wenn sie eingesetzt wird, um asynchron kommunizierende Systeme zu beschreiben (siehe z.B. [JJH90], [Jos92]). Während andere Semantiken für asynchron kommunizierende Systeme (siehe z.B. [BDD$^+$93], [Jon90], [Dil89]) das Verhalten die modellierten Systeme einheitlich behandeln, weisen einige Systeme bei der Modellierung mit der Failure-Divergence-Semantik eine besondere Verhaltensform, genannt *Divergenz*, auf. Dieses Verhalten tritt auf, wenn eine unbeschränkte Anzahl von Aktionen, zum Beispiel von Kommunikationsereignissen, innerhalb eines Systems auftreten kann, ohne von außen wahrgenommen zu werden.

Das Ziel dieses Abschnitts ist es, auf der einen Seite die Notwendigkeit der Definition einer erweiterten Failuresemantik zu demonstrieren und auf der anderen Seite eine Motivation für die hier eingeführte unendliche Failuresemantik zu geben. Im übrigen sollte darauf hingewiesen werden, daß die hier vorgestellte Erweiterung in eine vollständige andere Richtung abzielt als die in [Ros88] vorgestellte Arbeit. Auch wenn [Ros88] ebenfalls das Failuremodell um unendliche Abläufe erweitert, wird hier ein vollständig anderes semantisches Modell gewählt, Während in dieser Arbeit das Hauptaugenmerk darauf liegt,

das Phänomen der Divergenz aus dem Modell zu entfernen, ist dort die Behandlung des unbeschränkten Nichtdeterminismus das Ziel; um dies zu erreichen werden dort unendliche Spuren als dritte Komponente zu den Failure- und den Divergenzmengen hinzugenommen, anstatt diese in den Failureanteil mitaufzunehmen. Abschnitt A.5 zeigt jedoch, daß das unendliche Failuremodell ebenfalls in der Lage ist, unbeschränkten Nichtdeterminismus zu modellieren.

Dieser Teil wurde auf Leser mit wenig Hintergrundwissen im Bereich der Modellierung mittels Failure- und Failure-Divergence-Modellen zugeschnitten. Für erfahrene Lesern bietet es sich daher an, Abschnitt A.2 vollständig zu überspringen.

A.2 Divergenz

In diesem Abschnitt wird kurz die Divergenz in der Form beleuchtet, wie sie in der für CSP eingeführten Semantik auftritt. Dazu wird zuerst die endliche Failuresemantik ohne Divergenz eingeführt; diese stellt auch die Basis für die unendliche Failuresemantik dar. Nach einem kurzen Blick auf deren Eigenschaften wird eine knappe Einführung in die Failure-Divergence-Semantik gegeben, und die Frage erhoben, ob die Modellierung der Divergenz in jedem Fall wünschenswert ist.

A.2.1 Endliche Failuresemantik

Die endliche Failuresemantik wurde als ausgefeilteres und ausdrucksstärkeres Modell für CSP eingeführt (siehe z.B. [BHR84]), als sich herausstellte, daß Spuren keine ausreichende Semantik liefern (siehe z.B. [Hoa80]). Da die endliche Failuresemantik die Basis für die neu eingeführte unendliche Failuresemantik darstellt, wird hier die endliche Failuresemantik zusammen mit einer Syntax zur Beschreibung kommunizierender Prozesse eingeführt, sowie das Modell und die Denotation angegeben. Ein kurzer Blick auf deren Eigenschaften mittels einiger Beispiele beschließt den Abschnitt.

Prozeßausdrücke

Vor der Definition des Modell wird ein Ausschnitt der Syntax zur Beschreibung synchron kommunizierender Prozesse angegeben, der für die auftretenden Beispiele relevant ist. Eine Beschreibung der vollständigen Syntax findet sich z.B. in [Hoa85],[1]

Definition A.2.1 (Wohlgeformte Prozeßterme) Sei im folgenden A eine Menge von Aktionen mit $H, A_1, A_2 \subseteq A$ und $A_1 \cup A_2 = A$. Die Menge der wohlgeformten Prozesse mit Alphabet A wird induktiv definiert als

[1] Insbesondere wird hier nicht das Konzept der freien Variablen für die Rekursion erläutert; dieses wird in [Hoa85] ausführlich behandelt.

- CHAOS_A ist ein Prozeßterm mit Alphabet A
- STOP_A ein Prozeßterm mit Alphabet A

Sind weiterhin P, P_1 und P_2 wohlgeformte Prozeßterme mit Alphabeten A, A_1 bzw. A_2, $a \in A$, und f eine injektive[2] Funktion auf A, dann ist

- $a \rightarrow P$ ein Prozeßterm mit Alphabet A
- $P_1 \parallel P_2$ ein Prozeßterm mit Alphabet $A_1 \cup A_2$
- $P_1 \,\Box\, P_2$ ein Prozeßterm mit Alphabet A, falls $A_1 = A_2 = A$
- $P_1 \sqcap P_2$ ein Prozeßterm mit Alphabet A, falls $A_1 = A_2 = A$
- $f(P)$ ein Prozeßterm mit Alphabet $\bar{f}(A)$[3]
- $P \backslash H$ ein Prozeßterm mit Alphabet A

Schließlich gilt, falls $P(X)$ ein wohlgeformter Prozeßterm mit Alphabet A und der freien Variablen X als eine seiner Teilprozeßterme mit Alphabet A, ist, auch

- $\mu X : A.P(X)$ ist ein Prozeßterm mit Alphabet A

$\circ$

Das Alphabet eines Prozeßterms P wird im folgenden auch mit αP bezeichnet.

Da einige dieser wohlgeformten Prozeßausdrücke in den folgenden Beispielen benötigt werden, wird zum einfacheren Verständnis vor der formalen Definition noch eine kurze Erläuterung und Intuition der folgenden Prozeßausdrücke angeben:

CHAOS_A**:** Der Prozeß mit beliebigem Verhalten; dieser maximal nichtdeterministische Prozeß kann nach jedem Ablauf jede beliebige Menge von Aktionen ablehnen.

STOP_A**:** Der Prozeß, der gleich zu Beginn jede Aktion aus A ablehnt.

$a \rightarrow P$**:** Der Prozeß, der die Ausführung der Aktion a anbietet, und sich dann wie der Prozeß P verhält.

$P_1 \parallel P_2$**:** Der Prozeß, der sich verhält wie P_1 und $P2$, wenn diese parallel ablaufen, und alle Aktionen aus ihren gemeinsamen Alphabeten synchronisierend ausführen.

$P_1 \,\Box\, P_2$**:** Der Prozeß, der jede beliebige initiale Aktion von sowohl P_1 als auch P_2 anbietet; nach der Durchführung dieser Aktion verhält er sich wie der entsprechende Prozeß, der diese Aktion angeboten hat.

$P_1 \sqcap P_2$**:** Der Prozeß, der sich entweder nur wie P_1 oder nur wie P_2 verhält.

$P \backslash H$**:** Der Prozeß, der sich wie P verhält, wenn alle Aktionen von H gegenüber der Umgebung verborgen werden; alle ausführbaren Aktionen aus H können dabei von P sofort ausgeführt werden.

[2]Da als Bildbereich von f hier stets $\{f(a) \mid a \in A\}$ verwendet wird, wird f als bijektive Funktion betrachtet.

[3]Dabei ist $\bar{f}(A)$ definiert als $\{f(a) \mid a \in A\}$.

$f(P)$**:** Der der Prozeß, der sich wie P verhält, wobei jede Aktion a von P durch die Aktion $f(a)$ ersetzt wird.

$\mu X : A.P(X)$**:** Der rekursiv definierte Prozeß mit Rekursionsvariable X und dem Rumpf $P(X)$ definiert als kleinster[4] Fixpunkt von $X = P(X)$.

Mit dieser Syntax kann nun nach Bereitstellen des entsprechenden Modells eine Denotation für Prozeßterme wie in [Hoa85] angegeben werden.

Prozesse und endliche Failuresemantik

Um die Semantik eines Prozeßterms zu definieren, muß zuerst ein geeignetes Modell angegeben werden. Dazu wird der Begriff des *Prozesses*, entsprechend wie in [Hoa85] eingeführt. Ein *Prozeß* wird im wesentlichen als Menge von *Failure*-Paaren (t, R) definiert, wobei t eine (endliche) Spur von Aktionen eines vorgegebenen Alphabets A von Aktionen, und R eine Teilmenge dieses Alphabets darstellt. Dabei drückt dieses Paar aus, daß der zugehörige Prozeß nach der Ausführung der Aktionssequenz t jede Aktion aus A zurückweisen kann.

Definition A.2.2 (Prozeß) Zu einem gegebenen Alphabet A von Aktionen wird eine Menge $P \subseteq A^* \times \mathbb{P}(A)$ als *Prozeß mit Alphabet A* bezeichnet, falls

1. $(\langle\rangle, \emptyset) \in P$[5]
2. $(s \circ t, R) \in P \Rightarrow (s, \emptyset) \in P$[6]
3. $(t, R) \in P \wedge S \subseteq R \Rightarrow (t, S) \in P$
4. $(t, R) \in P \Rightarrow (t \circ a, \emptyset) \in P \vee (t, R \cup \{a\}) \in P$

wobei $a \in A$. Die Menge $\mathcal{F}_A \subseteq \mathbb{P}(A^* \times \mathbb{P}(A))$ aller Prozesse mit Alphabet A wird als (endliches) Failuremodell mit Alphabet A bezeichnet.[7] Zusammen mit der Ordnungsrelation

$$F_1 \sqsubseteq F_2 \stackrel{\text{def}}{=} F_2 \subseteq F_1$$

definiert $(\mathcal{F}, \sqsubseteq)$ den *(endlichen) Failurebereich* mit $A^* \times \mathbb{P}(A)$ als schwächstes Element. ∘

Definition A.2.3 (Failuresemantik) Mit dem Begriff *endliche Failuresemantik F* wird die - partielle - Abbildung von Prozeßtermen auf Prozesse mit Alphabet A und den folgenden Eigenschaften verstanden:

- $\mathcal{F}[\mathsf{CHAOS}_A] = A^* \times \mathbb{P}(A)$
- $\mathcal{F}[\mathsf{STOP}_A] = \{(\langle\rangle, R) \mid R \subseteq A\}$

[4] Hierzu wird als Ordnung die umgekehrte Inklusionsordnung $\supseteq$ verwendet.

[5] $\langle\rangle$ bezeichnet die leere Spur, d.h. die Spur, die keine Aktionen enthält.

[6] $x \circ y$ bezeichnet die Spur, die aus der Konkatenation der Spuren x und y besteht, wobei x und y auch einfache Aktionen sein können.

[7] Falls das benutzte Alphabet eindeutig ist, kann auf dessen Angabe verzichtet werden.

- $\mathcal{F}[a \to P] = \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \cup \{(a \circ t, R) \mid (t, R) \in \mathcal{F}[P]\}$
- $\mathcal{F}[P_1 \parallel P_2] = \{(t, R_1 \cup R_2) \mid (A_1 © t, R_1) \in \mathcal{F}[P_1] \wedge (A_2 © t, R_2) \in \mathcal{F}[P_2]\}$
- $\mathcal{F}[P_1 \,\square\, P_2] = \{(t, R) \mid t \neq \langle\rangle \wedge (t, R) \in \mathcal{F}[P_1] \cup \mathcal{F}[P_2]\} \cup \{(\langle\rangle, R) \mid (\langle\rangle, R) \in \mathcal{F}[P_1] \cap \mathcal{F}[P_2]\}$
- $\mathcal{F}[P_1 \sqcap P_2] = \mathcal{F}[P_1] \cup \mathcal{F}[P_2]$
- $\mathcal{F}[P \backslash H) = \{((A \backslash H) © t, R \backslash H) \mid (t, R \cup H) \in \mathcal{F}[P]\}$
- $\mathcal{F}[F(P)] = \{(f^*(t), \bar{f}(R) \mid (t, R) \in \mathcal{F}[P]\}$[8]
- $\mathcal{F}[\mu X : A.P(X)] = \mathsf{lub}_i \, \mathcal{F}[P^i(\mathsf{CHAOS}_A)]$

wobei P_1 und P_2 Prozeßterme mit Alphabet A_1 bzw. A_2 sind. ∘

Beispiel

Die Tatsache, daß das endliche Failuremodell nicht ausreicht, um eine geeignete denotationelle Semantik für die in Definition A.2.1 eingeführten Prozeßterme zu konstruieren, wurde bereits in [BHR84] gezeigt. Im folgenden wird ein Beispiel eines divergenten Systems angegeben, um einerseits das Phänomen der Divergenz und die Unzulänglichkeit der endlichen Failuresemantik zu demonstrieren, und andererseits die Unterschiede zur unendlichen Failuresemantik aufzeigen zu können.

Beispiel A.2.1 (Unbeschränkte Aktionen) Im folgenden sei $A = \{a, b\}$ das Alphabet des Prozeßausdrucks E. Der vom Prozeßausdruck E beschriebene Prozeß führt unbeschränkt viele Aktionen a oder b aus. Der durch D beschriebene Prozeß entsteht durch die Abstraktion der Aktion a.

$$E = \mu X : A.(a \to X \,\square\, b \to X) \tag{A.1}$$
$$D = E \backslash \{a\} \tag{A.2}$$

Entsprechend der Definition der Failuresemantik läßt sich leicht herleiten, daß für den Prozeß von E

$$F[E] = \{(t, R) \mid t \in A^* \wedge R = \emptyset\}$$

gilt. Weiterhin gilt für das Verbergen von Aktionen entsprechend der Definition der Failuresemantik

$$\begin{aligned} F[D] &= \{(t, R \backslash \{a\}) \mid (t, R \cup \{a\}) \in F[E]\} \\ &= \emptyset \end{aligned}$$

Insgesamt wird so dem Prozeßausdruck D bei Verwendung der endlichen Failuresemantik die leere Menge als Failuremenge zugeordnet. ◇

[8] Dabei ist $\bar{f} : \mathbb{P}(A) \to \mathbb{P}(A')$ zu gegebenem $f : a \to A'$ definiert als $\bar{f}(R) \stackrel{\text{def}}{=} \{f(a) \mid a \in R\}$.

Beispiel A.2.1 zeigt die Unzulänglichkeit der endlichen Failuresemantik bei divergenten Systemen: Entsprechend der Definition der endlichen Failuresemantik läßt sich dem wohlgeformten Prozeßausdruck E ein Prozeß im Sinne von 2.3.1 zuordnen. Nach dem Verbergen der Aktion "a" ist dies jedoch nicht mehr möglich. Denn entsprechend der Definition der endlichen Failuresemantik wird dem divergenten Prozeß $F[D]$ lediglich die leere Menge als Failuremenge zugeordnet. Dies ist jedoch kein Prozeß im Sinne von 2.3.1; auch anschaulich betrachtet ist dies nicht wünschenswert, da es einem wohlgeformten Prozeßausdruck eine Eigenschaften zuordnen kann. Insgesamt ist damit also die endliche Failuresemantik nicht auf allen wohlgeformten Prozeßausrücken definiert.

A.2.2 Failure-Divergence-Semantik

Aus Platzgründen wird die Divergenzsemantik hier nicht formal eingeführt; der interessierte Leser sei auf [Hoa85] verwiesen. Hier wird nur eine knappe Einführung in den Divergenzbereich angegeben, um ein Gefühl für die Komplexität der zugehörigen Denotation zu geben.

Im wesentlichen wird das Failuremodell um eine zusätzliche Komponente, genannt *Divergenzmenge*, erweitert. Damit wird ein Prozeß $D \in \mathcal{D}$ zu einer Teilmenge von $\mathbb{P}(A^* \times \mathbb{P}(A)) \times \mathbb{P}(A^*)$. Diese neu eingeführte Menge enthält alle solche Spuren, nach denen der zugehörige Prozeß divergieren kann. Dementsprechend wird der am stärksten nichtdeterministische Prozeß CHAOS als $\mathbb{P}(A^* \times \mathbb{P}(A)) \times \mathbb{P}(A^*)$ definiert.

Die semantische Abbildung wird nun so definiert, daß ein divergenter Prozeß am Divergenzpunkt das gleiche Verhalten aufweist wie CHAOS. Die interessanteste Definition ist hier natürlich die Definition der Divergenz für das Verbergen von einer Menge von Aktionen H eines Prozesses P mit Alphabet A. Diese wird hier aus Demonstrationszwecken angegeben. Der Divergenzanteil der Denotation wird definiert als

$$\{(A\backslash H)©s \circ t \mid t \in (A\backslash H) \wedge (\ s \in \text{divergences}(P) \vee (\forall n.\exists u \in H^*.\#u > n \wedge (s \circ u) \in traces(P)))\}$$

wobei $\#u$ die Länge (Anzahl der Elemente) u von bezeichnet.

A.2.3 Unzulänglichkeit der Divergenz

Die explizite Unterscheidung zwischen divergenten und nichtdivergenten Systemen hat, wie oben gezeigt, eine zusätzliche Komplexität der Semantik in Form der Divergenzmengen zur Folge. Dabei ist sich Hoare durchaus der Komplexität dieser Semantik bewußt; er weist jedoch darauf hin, daß die explizite Behandlung der Divergenz durchaus ihre Berechtigung hat (vgl. [Hoa85]:

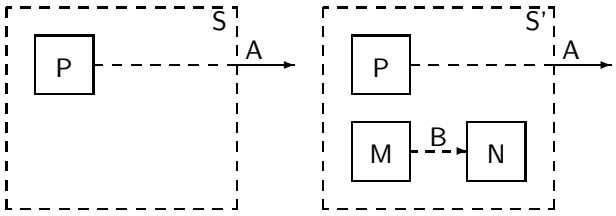

Abbildung A.1: Nichtdivergentes System S und divergentes System S'

> It is a shame to devote so much attention to divergence, when divergence is always something we do *not* want. Unfortunately, it seems to be an inevitable consequence of any efficient or even computable method of implementation. It can arise from either concealment or unguarded recursion; and it is part of the task of a system designer to prove that for his particular design the problem will not occur. In order to prove that something can't happen, we need to use a mathematical theory in which it can!

Während die *Möglichkeit* der Erkennung solcher Systeme für die Entwicklung verteilter Systeme ein wesentlicher Vorteil ist, ist jedoch die *Notwendigkeit*, solche Systeme gesondert zu behandeln, nicht auf allen Abstraktionsebenen unbedingt wünschenswert.

Anhand des folgenden Beispiels wird versucht, deutlich zu machen, warum eine explizite Behandlung von divergenten Systemen nicht immer wünschenwert sein muß.

Beispiel A.2.2 (Divergente Systeme) Abbildung A.1 zeigt zwei Systeme S und S' mit

- $S = P$
- $S' = (P \parallel (M \parallel N))\backslash B$

wobei

- P Alphabet A hat und $\mathcal{F}[P] = \{(t, R) \mid R \not\subseteq A\}$
- M Alphabet B hat und $\mathcal{F}[M] = \{(t, R) \mid R \not\subseteq B\}$
- N Alphabet B hat und $\mathcal{F}[N] = \{(t, R) \mid R \cap B = \emptyset\}$

mit

- $A \cap B = \emptyset$

Während System S nur aus Prozeß P besteht, ist System S' ähnlich wie Prozeß S aufgebaut, aber unter Hinzunahme zweier kommunizierender Prozesse M und N. Diese Prozesse M und N haben jedoch keinerlei Einfluß auf P; alle ihre Kommunikation nach außen ist verborgen. Die Prozesse P und M produzieren unbeschränkt Aktionen aus A bzw. B, während N unbeschränkt Elemente von B konsumiert. Aus intuitiver Sicht sollte das beobachtbare Verhalten von S und S' gleich sein, zumindest auf einer sehr abstrakten Darstellungsebene, da die Existenz der Unterkomponenten M und N zusammen

mit ihrer Kommunikation verborgen wird. Aber da die Failure-Divergence-Semantik eine Unterscheidung divergenter und nichtdivergenter Systeme erzwingt, unterscheidet sich ihr Verhalten essentiell. Während der Failureanteil von $\mathcal{D}[S]$ dem oben beschriebenen $\mathcal{F}[P]$ entspricht und daher einen Prozeß bezeichnet, der unbeschränkt Aktionen aus A produziert, liefert der Failureanteil von $\mathcal{D}[S']$ als Denotation CHAOS_A und damit einen divergenten Prozeß ohne weitere Information. Damit wird, obwohl S' intuitiv als Verfeinerung von P verstanden werden könnte,[9] S von der Failure-Divergence-Semantik als Verfeinerung von S' betrachtet, da S' mit dem maximal nichtdeterministischen Prozeß CHAOS gleichgesetzt wird. ◇

Das Beispiel A.2.2 zeigt, daß durch die Failure/Divergence-Semantik Prozesse unterschieden werden, deren einheitliche Behandlung unter Umständen durchaus wünschenswert wäre. Diese Problematik wird besonders deutlich, wenn die Failure/Divergence-Semantik in Zusammenhang mit anderen Semantiken gebracht werden, die divergente Systeme nicht gesondert betrachten, wie zum Beispiel die in Abschnitt 2.2 vorgestellte Spursemantik oder auch die stromverarbeitenden Funktionen (vgl. [BDD+93]) und die I/O-Automaten (vgl. [LT89]). Diese unterschiedliche Behandlung von Prozessen erschwert es, diese Semantiken in einen einheitlichen Rahmen einzubetten. Diese Einheitlichkeit ist jedoch für die in den vorangegangengen Kapiteln vorgestellte Vorgehensweise eine wesentliche Voraussetzung; daher ist es im Rahmen dieser Arbeit notwendig, eine Variante der Failure-Semantik zu verwenden, die eine einheitliche Prozeßsicht erlaubt.

A.3 Unendliche Failuresemantik

Nachdem bisher die Unzulänglichkeiten der endlichen Failuresemantik und der Failure-Divergence-Semantik vorgestellt wurden, wird nun ein neues Modell eingeführt, nämlich das *unendliche Failuremodell*. Es unterscheidet sich vom endlichen Failuremodell lediglich in der Hinsicht, daß der Spuranteil eines Failurepaares auch eine unendliche Spur sein kann. Während dies nur eine unwesentliche Erweiterung zu sein scheint, so ist doch der Einfluß auf die denotationelle Semantik durchaus bemerkenswert, wie sich in Abschnitt A.4 deutlich zeigt.

Auf der anderen Seite ist es etwas schwieriger einem Failurepaar mit einer unendlichen Spurkomponente eine intuitive Bedeutung zuzuordnen, da die Vorstellung einer Menge von Aktionen, die *nach* einer *unendlichen* Sequenz von Aktionen abgelehnt werden kann, kontraintuitiv ist. Einfach ist die folgende Interpretation: kann eine Aktion nicht in einer Refusalmenge zu einem Ablauf auftreten, so wird diese Aktion im Laufe dieses Ablaufs ununterbrochen angeboten. Abschnitt A.4.3 wird diese Eigenschaft genauer untersuchen.

[9] Diese Verfeinerung ist auch in anderen Formalismen möglich wie den stromverarbeitenden Funktionen (z.B. [BDD+93]), den I/O-Automaten (z.B. [LT89], [Jon90]) oder einigen spurbasierten Formalismen (z.B. [Dil89]).

A.3.1 Bereich

Um die endliche Semantik zu erweitern, wird die Menge der endlichen und unendlichen Spuren A^ω über einer gegebenen Menge A von Aktionen definiert als

$$A^\omega \stackrel{\text{def}}{=} A^* \cup A^\infty$$

wobei A^∞ die Menge der unendliche Spuren bezeichnet.

Aufbauend auf dieser Erweiterung der endlichen Spuren kann nun der Begriff des Prozesses analog zum Fall der endlichen Failuresemantik eingeführt werden.

Definition A.3.1 (Prozeß (Neufassung)) Zu einer gegebenen Menge (Alphabet) A von Aktionen wird die Menge $I \subseteq A^\omega \times \mathbb{P}(A)$ als *Prozeß mit Alphabet A* bezeichnet, falls

1. $(\langle\rangle, \emptyset) \in I$
2. $(s \circ t, R) \in I \Rightarrow (s, \emptyset) \in I$
3. $(t, R) \in I \wedge S \subseteq R \Rightarrow (t, S) \in I$
4. $(t, R) \in I \Rightarrow (t, R \cup \{a\}) \in I \vee (t \circ a, \emptyset) \in I$
5. $r \in A^* \wedge (r, \emptyset) \in I \wedge B \subseteq A \Rightarrow \exists s \in B^\omega.(r \circ s, B) \in I$
6. $(t, R) \in I \wedge a \copyright t \notin A^* \Rightarrow (t, R \cup \{a\}) \in I$
7. $(\mathsf{lub}_i\, t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I \Rightarrow (\mathsf{lub}_i\, t_i, R \cup \{a\}) \in I$

wobei $a \in A$. ∘

Zu den in Definition A.2.2 aufgeführten Eigenschaften wurden zusätzlich drei weitere hinzugenommen:

Terminierbarkeitsaxiom 5: Im unendlichen Failuremodell kann jeder endliche Ablauf r eines Prozesses durch eine entsprechende Erweiterung s zu einem Ablauf ergänzt werden, der den Prozeß zur Termination führt.

Serialitätsaxiom 6: Eine Aktion im unendlichen Failuremodell kann nicht zu sich selbst parallel ablaufen; tritt also eine Aktion a in einem Ablauf t unbeschränkt oft auf, so kann sie stets im zugehörigen Refusalanteil eines Failurepaares (t, R) auftreten.

Erweiterbarkeitsaxiom 7: Eine Aktion a, die in einem Ablauf $\mathsf{lub}_i\, t_i$ immer wieder nicht ausgeführt werden kann, kann stets im zugehörigen Refusalanteil eines Failurepaares $(\mathsf{lub}_i\, t_i, R)$ auftreten.

Die Menge $\mathcal{I}_A = \mathbb{P}(A^\omega \times \mathbb{P}(A))$ eines gegebenen Alphabets A wird als *unendliches Failuremodell* bezeichnet. Zusammen mit der Ordnungsrelation $I_1 \sqsubseteq I_2 \stackrel{\text{def}}{=} I_1 \supseteq I_2$ und $A^\omega \times \mathbb{P}(A)$ als schwächstes Element bilden sie den unendlichen Failurebereich.

A.3.2 Semantik

Nach der Einführung des semantischen Modells wird nun die Abbildung von Prozeßtermen auf Prozesse vorgestellt. Auch hier wird die Abbildung auf solche Prozeßterme beschränkt, wie sie in A.2.1 eingeführt wurden. Der Vergleich mit A.2.3 zeigt, daß die eingeführten Definitionen genau denjenigen der endlichen Failuresemantik entsprechen; dies gilt auch für die Konstruktoren, die hier nicht besprochen wurden. Auch die Definition der Abbildung rekursiver Prozeßterme ist identisch zur Definition für die endliche Failuresemantik. Der einzige Unterschied liegt im verwendeten Bereich:

Definition A.3.2 (Unendliche Failuresemantik) Die *unendliche Failuresemantik* ist eine Abbildung von Prozeßtermen mit Alphabet A auf Prozesse mit Alphabet A mit den Eigenschaften

- $\mathcal{I}[\mathsf{CHAOS}_A] = A^\omega \times \mathbb{P}(A)$
- $\mathcal{F}[\mathsf{STOP}_A] = \{(\langle\rangle, R) \mid R \subseteq A\}$
- $\mathcal{I}[a \to P] = \{(t, R) \mid (t = \langle\rangle \wedge R = \alpha P \backslash \{a\}) \vee (t = a \circ t' \wedge (t', R) \in \mathcal{I}[P])\}$
- $\mathcal{I}[P_1 \,\Box\, P_2] = \{(t, R) \mid \begin{array}{l}(t = \langle\rangle \wedge (t, R) \in \mathcal{I}[P_1] \cap \mathcal{I}[P_2]) \vee \\ (t \neq \langle\rangle \wedge (t, R) \in \mathcal{I}[P_1] \cup \mathcal{I}[P_2])\}\end{array}$
- $\mathcal{I}[P_1 \sqcap P_2] = \mathcal{I}[P_1] \cup \mathcal{I}[P_2]$
- $\mathcal{I}[P_1 \parallel P_2] = \{(t, R_1 \cup R_2) \mid (\alpha(P_1) \copyright t, R_1) \in \mathcal{I}[P_1] \wedge (\alpha(P_2) \copyright t, R_2) \in \mathcal{I}[P_1]\}$
- $\mathcal{I}[P \backslash H] = \{((A \backslash H) \copyright t, R \backslash H) \mid (t, R \cup H) \in \mathcal{I}[P]\}$
- $\mathcal{I}[f(P)] = \{(f^*(t), \bar{f}(R) \mid (t, R) \in \mathcal{I}[P]\}$[10]
- $\mathcal{I}[\mu X : A.P(X)] = \mathsf{lub}_{\mathcal{I}}\, \mathcal{I}[P^i(\mathsf{CHAOS})]$

wobei P_1 und P_2 Prozeßterme mit Alphabet A_1 bzw. A_2 sind. ∘

Wie bereits angesprochen, unterscheidet sich die Semantik der Operatoren zur Konstruktion von Prozeßtermen des unendlichen Failuremodells nicht von der des endlichen Failuremodells, abgesehen von der Definition rekursiver Prozesse mit unterschiedlichem CHAOS-Element. Dieser Unterschied hat jedoch wesentliche Folgen bei der Darstellung divergenter Systeme. Dies wird anhand der Beispiele im folgenden Abschnitt verdeutlicht.

Wie zu Beginn des Kapitels bemerkt, stellt die erschöpfende Behandlung der unendlichen Failuresemantik im Umfang der für TCSP eingeführten Prozeßbeschreibungssprache eine eigenständige Arbeit dar. Der volle Umfang ist auch in der hier vorgestellten Vorgehensweise nicht nötig, da hier nur beschränkte Systeme, also Systeme mit einer festen Anzahl parallel ablaufender Komponenten, betrachtet werden. Für die Untersuchungen der Eigenschaften der eingeführten denotationellen Semantik wird daher nicht der volle Sprachumfang von TCSP betrachtet. Statt dessen wird die denotationelle Semantik auf die im folgenden definierten Systeme sequentieller Prozesse eingeschränkt.

[10]Dabei ist $\bar{f} : \mathbb{P}(A) \to \mathbb{P}(A')$ zu gegebenem $f : a \to A'$ definiert als $\bar{f}(R) \stackrel{\text{def}}{=} \{f(a) \mid a \in R\}$.

Definition A.3.3 (Systeme sequentieller Prozeß) Ein wohlgeformter Prozeßterm gebildet durch die Anwendung der Konstruktoren CHAOS_A, STOP_A, $a \rightarrow P$, $P_1 \,\square\, P_2$, $P_1 \sqcap P_2$ sowie $\mu X : A.F(X)$ heißt *Term eines sequentiellen Prozesses*. Ein wohlgeformter Prozeßterm gebildet aus Termen sequentieller Prozesse durch die Anwendung der Konstruktoren $P_1 \parallel P_2$, $f(P)$ und $P \backslash H$ heißt *Term eines Systems sequentieller Prozesse*. ∘

Systeme sequentieller Prozesse beschreiben Realisierungen von Systemen synchron kommunizierender Prozesse mit einer statischen Anzahl von nebenläufigen Prozessen. Dies sind beispielweise CSP-Systeme wie in [Hoa78] beschrieben. Damit eignen sich diese Systeme sequentieller Systeme zur Realisierung der in Abschnitt 4.3 beschriebenen Systeme, wie sie in der in dieser Arbeit vorgestellten Vorgehensweise benötigt werden. Insbesondere reicht diese Klasse von Systeme auch aus, um Systeme zu beschreiben, die mittels hardwarenaher synchroner Sprachen wie `occam®2` realisierbar sind.

Für solche Systeme sequentieller Prozesse läßt sich die Wohldefiniertheit der unendlichen Failuresemantik zeigen:

Satz A.3.1 (Wohldefiniertheit der Semantik) Die unendliche Failuresemantik ist für Systeme sequentieller Prozesse wohldefiniert. •

Beweis A.3.1 (Satz A.3.1) Für die Wohldefiniertheit der Semantik sind zwei Eigenschaften nachzuweisen:

- die Wohldefiniertheit des kleinsten Fixpunktes $\mu X : A.F(X)$ für die Terme sequentieller Prozesse
- die Prozeßeigenschaften der Failuremengen, die den Termen der Systeme sequentieller Prozesse zugeordnet werden

Für den Nachweis der Wohldefiniertheit des kleinsten Fixpunktes kann im wesentlichen auf bekannte Literatur zurückgegriffen werden. Zu einer gegebenen Alphabetsmenge A bildet die Menge $\mathbb{P}(A^\omega \times \mathbb{P}(A))$ einen Bereich hinsichtlich der umgekehrten Mengeninklusion $\supseteq$ als Ordnung sowie CHAOS_A als kleinstem Element (vgl. z.B. [Win93]). Weiterhin sind die Operatoren CHAOS_A, STOP_A, $a \rightarrow P$, $P_1 \,\square\, P_2$, $P_1 \sqcap P_2$, $P_1 \parallel P_2$, $f(P)$ sowie $\mu X : A.F(X)$ stetig[11] (vgl. [Hoa85]). Damit ist die Definition des kleinsten Fixpunktes wohldefiniert.

Für den Nachweis der in Definition A.3.1 beschriebenen Prozeßeigenschaften wird auf Anhang B verweisen. □

A.4 Beispiele

In den folgenden beiden Unterabschnitten wird die oben eingeführte unendliche Failuresemantik anhand zweier Beispiele näher erläutert. Dabei wird besonders auf die unterschiedliche semantische Behandlung von Prozeßausdrücken durch die endliche und die unendliche

[11] Dabei definiert $\bigcap_{i \in \mathbb{N}} P_i$ die kleinste obere Schranke einer Kette $(P_i)_{I \in \mathbb{N}}$.

Failuresemantik eingegangen. Da der Unterschied gerade bei der Behandlung der Parallelität im Falle unbeschränkt aktiver Prozesse deutlich wird, werden zwei unterschiedliche Prozesse betrachtet, nämlich

- ein *sequentieller* Prozeß, der unbeschränkt oft zwei Aktionen a und b anbietet sowie
- zwei sequentielle Prozesse, die *parallel* zueinander ablaufen und jeweils unbeschränkt oft die Aktion a bzw. b anbieten.

Bei jedem dieser Beispiele wird kurz auf die unterschiedlichen Eigenschaften der endlichen und der unendlichen Failuresemantik hinsichtlich ihres Verhaltens bei der Abstraktion von internen Aktionen eingegangen. Am Ende dieses Abschnitts werden die Ergebnisse nochmals kurz zusammengefaßt und hinsichtlich ihrer Bedeutung für die Modellierung reaktiver Systeme interpretiert.

A.4.1 Sequentielle Prozesse

Der Unterschied zwischen der endlichen und der unendlichen Failuresemantik wird zunächst am Beispiel eines sequentiellen Prozesses deutlich gemacht. Der in A.3 und A.4 definierte Prozeßausdruck beschreibt dabei einen Prozeß, der unbeschränkt oft wahlweise die Aktion a oder die Aktion b akzeptiert:

$$P \stackrel{\text{def}}{=} \mu X : A.F(X) \tag{A.3}$$

$$F(X) \stackrel{\text{def}}{=} (a \to X) \,\Box\, (b \to X) \tag{A.4}$$

Dabei ist $A = \{a, b\}$. Die endliche Failuresemantik ordnet diesem Prozeßausdruck einen Prozeß zu, der nach keinem Ablauf eine Aktion ablehnen kann; dies wird in A.5 formalisiert:

$$\mathcal{F}[P] = \{(t, R) \mid t \in A^* \wedge R = \emptyset\} \tag{A.5}$$

Die unendliche Failuresemantik weist demgegenüber dank der Verwendung von Spuren aus A^ω einen wesentlichen Unterschied auf. Um dies zu verdeutlichen, wird zuerst in A.6 die Funktion definiert, deren kleinster Fixpunkt den Prozeß von P hinsichtlich der unendliche Failuresemantik darstellt. Entsprechend der Definition gilt:

$$\begin{aligned} \mathcal{I}[F](X) = \; & \{(\langle\rangle, \emptyset)\} \cup \\ & \{(x \circ t, R) \mid x \in \{a, b\} \wedge (t, R) \in X\} \end{aligned} \tag{A.6}$$

Der kleinste Fixpunkt dieser Funktion ist dabei der Prozeß, der sich – wie A.7 zeigt – hinsichtlich endlicher Abläufe wie in A.5 definiert verhält, im unendlichen Fall jedoch jede Aktion ablehnen kann:

$$\mathcal{I}[P] = \{(t, R) \mid t \in A^\omega \wedge R \subseteq A \wedge (t \in A^* \Rightarrow R = \emptyset)\} \tag{A.7}$$

Im folgenden wird gezeigt, daß $\mathcal{I}[P]$ einen Fixpunkt von A.6 darstellt; auf den Nachweis, daß dies in der Tat der kleinste Fixpunkt ist, wird hier verzichtet:[12]

$$
\begin{aligned}
&\mathcal{I}[P] \\
=\ &[\text{A.7}] \\
&\{(t,R) \mid t \in A^\omega \wedge R \subseteq A \wedge (t \in A^* \Rightarrow R = \emptyset)\} \\
=\ &[\text{Prädikatenlogik}] \\
&\{(t,\emptyset) \mid t \in A^*\} \cup \\
&\{(t,R) \mid t \in A^\infty \wedge R \subseteq A\} \\
=\ &[t = \langle\rangle \vee \exists x, t'.t = x \circ t'] \\
&\{\langle\rangle,\emptyset\} \cup \\
&\{(x \circ t,\emptyset) \mid x \in A \wedge t \in A^*\} \cup \\
&\{(t,R) \mid t \in A^\infty \wedge R \subseteq A\} \\
=\ &[t \in A^\infty \Leftrightarrow (x \circ t) \in A^\infty] \\
&\{\langle\rangle,\emptyset\} \cup \\
&\{(x \circ t,\emptyset) \mid x \in \{a,b\} \wedge t \in A^*\} \cup \\
&\{(x \circ t,R) \mid x \in \{a,b\} \wedge t \in A^\infty \wedge R \subseteq A\} \\
=\ &[\text{Prädikatenlogik}] \\
&\{\langle\rangle,\emptyset\} \cup \\
&\{(x \circ t,R) \mid x \in \{a,b\} \wedge t \in A^\omega \wedge R \subseteq A \wedge (t \in A^* \Rightarrow R = \emptyset)\} \\
=\ &[\text{A.7}] \\
&\{(\langle\rangle,\emptyset)\} \cup \\
&\{(x \circ t,R) \mid x \in \{a,b\} \wedge (t,R) \in \mathcal{I}[P]\} \\
=\ &[\text{A.6}] \\
&\mathcal{I}[F](\mathcal{I}[P])
\end{aligned}
$$

Abschließend wird anhand der Abstraktion hinsichtlich der Aktion b nochmals deutlich gemacht, wie sich dieser Unterschied auf die Modellierung divergenter Prozesse auswirkt. Dazu wird wiederum die endliche und die unendliche Failuresemantik des Prozeßausdrucks $P\backslash\{b\}$ betrachtet. Für erstere gilt, wie in A.8 gezeigt, daß diese keine geeignete Modellierung liefert:

$$\mathcal{F}[P\backslash\{b\}] = \emptyset \tag{A.8}$$

Die etwas komplexere Failure-Divergence-Semantik hingegen erlaubt es, diesem divergenten Prozeß eine Prozeß, nämlich dem Failure-Divergence-Prozeß $\mathsf{CHAOS}_{\{a\}}$, zuzuweisen. Das gleiche Ergebnis liefert die Verwendung der unendlichen Failuresemantik (vgl. A.9). Entsprechend der Definition der Semantik wird nämlich diesem Prozeßausdruck ein Prozeß zugeordnet, der dem schwächsten Prozeß mit Alphabet $\{a\}$ entspricht:

$$
\begin{aligned}
\mathcal{I}[P\backslash\{b\}] &= \{(t,R) \mid t \in \{a\}^\omega \wedge R \subset \{a\}\} \\
&= \mathsf{CHAOS}_{\{a\}}
\end{aligned}
\tag{A.9}
$$

[12] Für Theoreme wie das Exhaustionstheorem $t = \langle\rangle \vee \exists x, t'.t = x \circ t'$ siehe z.B. [Reg94].

A.4.2 Parallele Prozesse

Während das vorherige Beispiel die unterschiedliche Darstellung unbeschränkt aktiver sequentieller Prozesse behandelt, steht im folgenden Beispiel die Darstellung paralleler Prozesse im Vordergrund. Dazu werden in A.10 zwei Prozeßausdrücke Q_1 und Q_2 eingeführt, die - entsprechend dem vorherigen Beispiel - zu einer unbeschränkten Ausführung von Aktionen in der Lage sind. Im Gegensatz zu dort agiert jedoch hier Q_1 nur auf dem Alphabet $\{a\}$, sowie Q_2 nur auf dem Alphabet $\{b\}$.

$$\begin{array}{lcl} Q & \stackrel{\text{def}}{=} & Q_1 \parallel Q_2 \\ Q_1 & \stackrel{\text{def}}{=} & \mu X : a \rightarrow X \\ Q_2 & \stackrel{\text{def}}{=} & \mu X : b \rightarrow X \end{array} \tag{A.10}$$

Insgesamt ergibt sich durch die Parallelkomposition $Q_1 \parallel Q_2$ ein Prozeßausdruck Q, dessen Verhalten auf den ersten Blick dem Verhalten von P im vorherigen Beispiel sehr ähnlich ist. Beide sind in der Lage, unbeschränkt oft die Aktionen a und b auszuführen. Tatsächlich unterscheidet sich das System P vom System Q an einer entscheidenden Stelle, wobei dieser Unterschied nur im unendlichen Fall deutlich wird.

In diesem Unterschied spiegeln sich aber deutlich die unterschiedlichen Modellierungen mittels der beiden Semantiken wieder. Im Fall der endlichen Failuresemantik ergibt sich als Interpretation des Prozeßausdrucks Q der Prozeß:

$$\mathcal{F}[Q] = \{(t, R) \mid t \in A^* \wedge R = \emptyset\} \tag{A.11}$$

wobei $A = \{a, b\}$. Dieser stimmt mit der Darstellung mittels der unendlichen Failuresemantik nur im endlichen Fall überein:

$$\begin{array}{lll} \mathcal{I}[Q] = \{(t, R) \mid & t \in A^\omega \wedge R \subseteq A \wedge & \\ & (a \copyright t \in \{a\}^* \Rightarrow \{a\} \not\subseteq R) \wedge & \\ & (b \copyright t \in \{b\}^* \Rightarrow \{b\} \not\subseteq R)\} & \end{array} \tag{A.12}$$

Insgesamt kann also festgestellt werden, daß diese Unterscheidung zwischen P und Q bei der ersten Form der Darstellung nicht möglich ist. Tatsächlich ist die Interpretation der Prozeßausdrücke im Falle der endlichen Failuresemantik auch für beide dieselbe (vgl. A.5 und A.11):

$$\mathcal{F}[P] = \mathcal{F}[Q] \tag{A.13}$$

Hingegen ergibt sich für den Prozeßausdruck Q in der unendlichen Failuresemantik ein anderes Modell (vgl. A.7 und A.12):

$$\mathcal{I}[P] \neq \mathcal{I}[Q] \tag{A.14}$$

Dieser Unterschied wirkt sich dementsprechend in der gleichen Weise wie im vorherigen Beispiel auf die Anwendung der Abstraktion auf den Prozeßausdruck Q aus. Entsprechend dem Fall von $P\backslash\{b\}$ in A.8 gilt dann wegen A.13:

$$\mathcal{F}[Q\backslash\{b\}] = \emptyset \tag{A.15}$$

Entsprechend läßt sich natürlich in der Failure-Divergence-Semantik diesem Prozeß ebenfalls der Prozeß CHAOS zuordnen. Hingegen ist auch hier der Abstraktion von Q mittels der unendlichen Failuresemantik ein sinnvolles Modell zugeordnet:

$$\mathcal{I}[Q\backslash\{b\}] = \{(t, R) \mid t \in \{a\}^\omega \wedge R \subseteq \{a\} \wedge (t \in \{a\}^* \Rightarrow R = \emptyset\} \tag{A.16}$$

Das Verbergen des parallel ablaufenden Prozesses Q_2 hat also wegen $\mathcal{I}[Q\backslash\{b\}] = \mathcal{I}[Q_1]$ den Prozeß nicht beeinflußt.

A.4.3 Deutung der Ergebnisse

Wie oben gezeigt, behandelt die unendliche Failuresemantik verschiedene Fälle von Divergenz unterschiedlich:

Divergenz sequentieller Prozesse: Ein sequentieller Prozeß, wie in A.4.1 beschrieben, der divergiert, weil eine oder mehrere seiner Alternativen verborgen werden, kann die Aktivierung einer der Auswahlmöglichkeiten durch die Umgebung zulassen oder verweigern.

Divergenz paralleler Prozesse: Ein System, das einen divergenten Prozeß wie in A.4.2 enthält, wird sich dennoch immer noch wie ein entsprechendes System ohne diesen Prozeß verhalten, falls dieser Prozeß ansonsten keinen Einfluß auf das restliche System nimmt.

Dies bedeutet jedoch, daß die unendliche Failuresemantik im Gegensatz zur Failure-Divergence-Semantik parallele Prozesse anders behandelt als sequentielle Prozesse. Während ein Zweig einer Alternative in einem sequentiellen Prozeß fortlaufend ignoriert werden kann, gilt dies nicht für die Aktion eines nebenläufigen Prozesses. Damit wird ein sequentieller Prozeß "unfair" abgearbeitet, während eine paralleler Prozeß stets "schwach fair" behandelt wird.[13] Dies kann als natürliches Konzept von Fairness angesehen werden, da die faire Behandlung von Alternativen in einem sequentiellen Programm vom Programmierer sichergestellt werden sollte, während das gelegentliche Voranschreiten von parallel ablaufenden Prozessen eine allgemeine Beobachtung ist.

Dieses von der unendlichen Failuresemantik realisierte Fairnesskonzept wird in der Literatur auch als "faire Berechnung hinsichtlich der überlappenden Semantik" (*fair overlapping computation*, [Fra86]) bezeichnet. Das Konzept dieses Fairnessmodells verdeutlicht das folgende Beispiel.

[13] Siehe e.g. [Fra86] über die Definition unterschiedlicher Klassen von Fairness.

Beispiel A.4.1 (Überlappungssemantik) Zu einer Menge von Aktionen $\{a, b\}$ wird ein Prozeß definiert, der unbeschränkt wahlweise a und b ausführen kann:

$$\mu P : \{a, b\}.(a \to P) \,\square\, (b \to P)$$

Dieser Prozeß wird eingebettet in eine Umgebung bestehend aus den Prozessen

$$a \to \mathsf{STOP}$$

also einem Prozeß, der nach der Ausführung von a terminiert, und

$$\mu Q.b \to Q$$

einem Prozeß, der unbeschränkt b ausführen kann. Entsprechend der überlappenden Semantik wird die unbeschränkte Ausführung von b als fair betrachtet, weil die Auswahl von b die Bereitschaft von a durch den Prozeß P vorübergehend beendet und damit eine "Verschwörung" (*conspiracy*) gegen a stattfindet. Damit ist a nicht ununterbrochen ausführbereit und die Berechnung fair. ◇

In diesem Zusammenhang wird auch darauf hingewiesen, daß die überlappende Semantik wohl das natürlichere Modell darstellt, wenngleich die Verifikation solcher Systeme komplizierter ist als unter der Annahme der allgemeinen schwachen Fairness ([Fra86]):

> The overlapping semantics is, of course, closer to the nature of CSP as originally defined (...). It is most likely that truly-concurrent implementation fo the language will adhere to some form of overlapping semantics. On the other hand, the serialized semantics is handier to reason about, reflecting interleavings of communications (...).

Mit diesem Aspekt ist die Frage verbunden, ob eine denotationelle Semantik ausdrücklich zwischen vertauschbaren aber sequentiellen Aktionsfolgen - wie in Beispiel A.4.1 - und einem nebenläufigen Ablauf der gleichen Aktionen - wie in Beispiel A.4.2 - unterscheiden sollte. Während die Einfachheit des Modells oft als Argument für die erste Sichtweise, genannt "Interleaving"-Semantiken, angesehen wird, werden Probleme wie die besprochene "Divergenz" als Argument für die letztere, genannt "Echte Nebenläufigkeit"-Semantiken, verwendet. Während nun das unendliche Failuremodell die Interleaving-Sicht des endlichen Failuremodells verwendet, wird es durch die Erweiterung um unendliche Abläufe einen Schritt näher an die Sichtweise der echten Nebenläufigkeit herangeführt.[14] Es stellt damit einen geeigneten Kompromiß zwischen einfacher Modellierung und Ausdrucksmächtigkeit bei der Behandlung synchroner Systeme dar.

Die oben erwähnte, vom unendliche Failuremodell realisierte äußerst schwache Form von Fairness wird besonders deutlich, wenn die mit Fairness verbundene Nichtstetigkeit bzw. Nichtzulässigkeit betrachtet wird.[15] Dazu muß die im endlichen Failuremodell gültige Aus-

[14] Siehe z.B. [Bro92] für einen anderen Ansatz mit ähnlicher Zielsetzung; hier findet sich auch eine ausführlichere Diskussion von *Fairness*, *Interleaving*, *echte Nebenläufigkeit*.

[15] "In denotational semantics, (the study of fairness-like constructs)(...) revealed the non-(ω-)continuity of the semantic functions (...)." ([Fra86]).

sage "Parallelität entspricht Interleaving" herangezogen werden. Diese läßt sich formalisieren als

$$(a \to P) \parallel (b \to Q) = a \to (P \parallel (b \to Q)) \,\Box\, b \to ((a \to P) \parallel Q) \tag{A.17}$$

Diese Aussage erlaubt die Entfaltung paralleler Prozesse zu sequentiellen Prozessen unter Verwendung des $\Box$-Operators. Insbesondere kann damit im klassischen Failuremodell gezeigt werden, daß für die Prozesse der Art $P \stackrel{\text{def}}{=} a \to P$, $Q \stackrel{\text{def}}{=} b \to Q$ und $S \stackrel{\text{def}}{=} (a \to S) \,\Box\, (b \to S)$ gilt:

$$P \parallel Q = S$$

Dies zeigt sich leicht - unter Verwendung der in [Hoa85] eingeführten algebraischen Gesetze für Prozeßterme - mit obiger Aussage:

$P \parallel Q$
$=$ [Definition P,Q]
$(a \to P) \parallel (b \to Q)$
$=$ [Eigenschaft A.17]
$a \to (P \parallel (b \to Q)) \,\Box\, b \to ((a \to P) \parallel Q)$
$=$ [Definition P, Q]
$a \to (P \parallel Q) \,\Box\, b \to (P \parallel Q)$

Dies entspricht genau der Definition von S bei Substitution von $P \parallel Q$. Entsprechend den algebraischen Regeln von TCSP gilt damit $P \parallel Q = S$.

Diese Entfaltungsregel darf jedoch in der unendlichen Failuresemantik zwar auf alle endlichen Approximation von Prozessen, nicht jedoch auf die rekursiven Prozesse selbst angewandt werden. Ersteres folgt sofort aus Definition A.3.2, letzteres aus den Beispielen A.4.1 und A.4.2. Damit wird deutlich, daß die unendliche Failuresemantik parallele und sequentielle Prozesses zwar nicht im endlichen, aber im unendlichen Verhalten unterscheidet.

A.5 Unterbrechung

Die im unendlichen Failuremodell eingeführte Form der Synchronisierung erlaubt, wie oben beschrieben, nur eine sehr eingeschränkte Form der Fairness, die nicht unmittelbar der Definition der *schwachen Fairness* entspricht. Tatsächlich wird jedoch auch die Möglichkeit zur Beschreibung von unbeschränktem Nichtdeterminismus oder entsprechend der Möglichkeit zur Integration von Fairness im allgemeinen als wünschenswert angesehen ([Fra86]:

> Thus, our basic belief is that at the appropriate level of abstraction one should identify one common property that generalizes all actual behaviors of a program and adopt it as a starting point for design and reasoning. Fairness is such a generalization.

Auch einige in dieser Arbeit eingeführte Beispiele benötigen Sprachkonstrukte, die eine schwach faire Auswahl einer Alternative benötigen (z.B. die Empfängerkomponente aus Beispiel 3.4.2 oder der Quittungsmultiplexer aus 4.6.1). Für diese Form des Verhaltens wurde in Abschnitt 2.2.5 auf der Spurebene die Klasse der *unterbrechbaren Prozesse* definiert. Entsprechend wird in diesem Abschnitt ein Prozeßoperator eingeführt, der die Konstruktion einer schwach fairen Alternative für synchron kommunizierende Prozesse in Form einer Unterbrechungsbehandlung erlaubt. Abschließend mit dem neu eingeführten Operator beispielhaft ein schwach fairer Prozeß, der unterbrechbare Sender, definiert.

A.5.1 Prozeßoperator

Bereits in [Hoa85] wird ein Unterbrechungsoperator $P \triangle Q$ eingeführt. Dieser hat die Eigenschaft, die Ausführung des unterbrechbaren Prozesses P zu beenden, sobald eine Unterbrechungsaktion ausgeführt werden kann. Eine Unterbrechungsaktion ist dabei eine Aktion, die der unterbrechende Prozeß Q zu Beginn seines Ablaufs ausführen kann. Zur Konstruktion eines solchen Prozesses in der unendlichen Failuresemantik wird ein entsprechender Unterbrechungsoperator "$\triangle$" eingeführt. So bezeichnet $P \triangle Q$ den Prozeß, der

- sich solange wie der Prozeß P verhält, wie die Umgebung keine Unterbrechungsaktion ausführt, und
- sich ab der Annahme einer Unterbrechungsaktion wie der Prozeß Q verhält.

Entsprechend wird die unendliche Failuresemantik erweitert, um diesen Operator darzustellen:

Definition A.5.1 (Unterbrechung) Seien P, Q zwei Prozesse mit $\alpha P \subseteq \alpha Q$. Dann ist $\alpha(P \triangle Q) \stackrel{\text{def}}{=} \alpha Q$ und

$$\mathcal{I}(P(\triangle Q) \stackrel{\text{def}}{=} \mathcal{I}(P) \cup \{(s \circ t, R) \mid s \in (\alpha P)^* \wedge t \neq \langle\rangle \wedge (s, \emptyset) \in \mathcal{I}(P) \wedge (t, R) \in \mathcal{I}(Q)\}$$

$\circ$

Aufgrund dieser Definition des Unterbrechungsoperators werden diejenigen Aktionen i als Unterbrechungsaktionen definiert, die

- nur von Q ausgeführt werden ($i \in \alpha Q \backslash \alpha P$), und
- die von Q zu Beginn angeboten werden ($(\langle\rangle, \{i\}) \notin \mathcal{I}[Q]$).

Dabei wird - um das formale Arbeiten mit dem Operator zu vereinfachen - eine entsprechende Bedingung wie in [Hoa85] an die Unterbrechungsaktionen des Unterbrechungsprozeß Q erhoben: die von Q initial ausführbaren Aktionen dürfen nicht im Alphabet von P enthalten sein. Formal läßt sich diese Anforderung formulieren als

$$(i, \emptyset) \in \mathcal{I}[Q] \Rightarrow i \notin \alpha P \tag{A.18}$$

Im folgenden Beispiel wird gezeigt, wie sich mit solchen Aktionen unterbrechbare Prozesse definieren lassen.

Der Unterbrechungsoperator wird zur Menge der Konstruktionsoperatoren für sequentielle Prozesse hinzugenommen. Da die Stetigkeitsaussage der Konstruktoren auch für den Unterbrechungoperator gilt, läßt sich die Wohldefiniertheitsaussage von Satz A.3.1 auch auf diese Prozeßterme ausdehnen.

A.5.2 Beispiel

Mit dem Unterbrechungsoperator ist es möglich, sequentielle und schwach faire Prozesse zu definieren. Beispiele für Prozesse, die mit den bisher eingeführten Prozeßoperatoren nicht definierbar sind, sind der *faire* Mischknoten oder der - an den Sender in Beispiel 3.4.2 angelehnte - unterbrechbare Sender.

Beispiel A.5.1 (Sender) Mit Hilfe des Unterbrechungsoperators ist es möglich, Prozesse zu definieren, deren Ablauf von einer ständig ausführbaren Aktion unterbrochen werden kann. Dies ist beispielsweise nötig bei der Definition des Sendeprozesses im "Alternierenden Bit"-Protokoll. Hier wird nur eine vereinfachte Version gezeigt: ein Prozeß, der eine Ausgabe o solange wiederholt, bis er durch eine Eingabe i beendet wird. Dazu wird das triviale Alphabet $A = \{i, o\}$ verwendet. Der Prozeß wird definiert als

$$\begin{array}{lcl} P & \stackrel{\text{def}}{=} & Q \triangle (i \to \mathsf{STOP}_A) \\ Q & \stackrel{\text{def}}{=} & \mu X : \{o\}.o \to X \end{array}$$

Die unendliche Failuresemantik ordnet Q die Failuremenge

$$\mathcal{I}[Q] = (\{o\}^* \times \emptyset) \cup (\{o\}^\infty \times \mathbb{P}(\{o\}))$$

zu, sowie dem Sender P die Failuremenge

$$\mathcal{I}[P] = \mathcal{I}[Q] \cup \{(t \circ i, R) \mid t \in \{o\}^* \wedge R \subseteq A\}$$

Das Verbergen der Unterbrechungsaktion i macht dessen faire Auswahl mittels des Unterbrechungsoperators offensichtlich:

$$\begin{array}{rcl} \mathcal{I}[P\backslash\{i\}] & = & \{(\{o\}©t, R\backslash\{i\}) \mid i \in R \wedge (t, R) \in \mathcal{C}[P]\} \\ & = & \{\{o\}©(t \circ i, R\backslash\{i\}) \mid i \in R \wedge t \in \{o\}^* \wedge R \subseteq A\} \\ & = & \{(t, R) \mid t \in \{o\}^* \wedge R \subseteq \{o\}\} \end{array}$$

Mittels $P\backslash\{i\}$ wird also ein Prozeß beschrieben, der nach endlichem Ablauf $t \in \{o\}^*$ intern die Aktion i ausführt und so zur Termination des Prozesses nach einer endlichen aber unbeschränkten Anzahl von o-Schritten führt. ◇

A.6 Zusammenfassung und Ausblick

Zusammenfassend kann festgestellt werden, daß die in dieser Arbeit definierte unendliche Failuresemantik für die hier betrachteten Systeme sequentieller Prozesse eine einfache und elegante Formulierung einer Semantik erlaubt. Divergente und nichtdivergente Systeme werden auf eine einheitliche Weise behandelt. Die unendliche Failuresemantik kann damit gegenüber der endlichen Failuresemantik als abstrakterer Ansatz zur Beschreibung des Systemverhaltens synchron kommunizierender Systeme angesehen werden. Entsprechend der in A.4.3 gezeigten Ergebnisse kann die unendliche Failuresemantik gegenüber der endlichen in einiger Hinsicht sogar als intuitiver beurteilt werden kann.

Schließlich erlaubt es die unendliche Failuresemantik, die Kluft zwischen den Beobachtungsbegriffen, die für asynchron und synchron kommunizierende Systeme eingeführt wurden, zu schließen. Dies schafft die Voraussetzung für die Einbettung beider Paradigmen in eine gemeinsame Entwicklungsmethode.

Darüber hinaus bietet die unendliche Failuresemantik

Unbeschränkten Nichtdeterminismus: Ähnlich wie die in der Arbeit von [Ros88] eingeführte Erweiterung erlaubt die unendliche Failuresemantik die Darstellung von unbeschränktem Nichtdeterminismus. Im Gegensatz zu dem dort vorgestellten Ansatz benötigt die unendliche Failuresemantik jedoch keine komplexe Erweiterung des semantischen Modells.[16] Weiterhin erlaubt sie die programmiersprachliche Konstruktion unbeschränkt nichtdeterministischer Prozesse basierend auf dem der Semantik zugrundeliegenden Fairnessbegriff.

Überlappend-faire Abläufe: Die unendliche Failuresemantik realisiert mit der in [Fra86] beschriebenen *Fairness hinsichtlich der Überlappungssemantik* ein – verglichen mit der schwachen Fairness – abgeschwächtes Fairnesskonzept. Im wesentlichen garantiert dieses Fairnesskonzept das Fortschreiten unabhängiger nebenläufiger Prozesse und damit ein intuitiv sehr eingängiges Fairnesskonzept.

Nebenläufigkeit: Im Gegensatz zur endlichen Failuresemantik gilt in der unendlichen Failuresemantik nicht die Aussage "*Parallelität = beliebige Sequentialisierung*". Damit basiert die unendliche Failuresemantik zwar auf einem einfachen Interleaving-Modell, erlaubt aber, wie Semantiken mit echter Nebenläufigkeit, die Unterscheidung zwischen nebenläufigen Prozessen und deren wahlfreien Sequentialisierung.

Insgesamt erlaubt die unendliche Failureeigenschaft die Untersuchung von elementaren Konzepten der Modellierung reaktiver Systeme, die über die Möglichkeiten der endlichen Failuresemantik hinausgehen. Die bisher festgestellten Eigenschaften legen es nahe, die unendliche Failuresemantik unabhängig von der in dieser Arbeit eingeführten methodischen Vorgehensweise und über die bereits geführten Betrachtungen hinaus zu untersuchen.

[16] In [Ros88] wird das Modell neben den Failure- und Divergence-Mengen um eine dritte Komponente, die Menge der möglichen endlichen oder unendlichen Abläufe, erweitert.

Von besonderem Interesse ist in diesem Zusammenhang die Frage, ob die entwickelte denotationelle Semantik über die beschriebenen Systeme sequentieller Prozesse hinaus auf den vollen Umfang von TCSP-Prozeßtermen ausgeweitet werden kann. Dazu sind die Wohldefiniertheit und die Verträglichkeit für diese Systeme zu untersuchen. Dies ermöglicht insbesondere die Modellierung von Systemen mit dynamisch veränderbarer Prozeßstruktur.

Ein weiterer bedeutsamer Punkt, der bisher noch nicht angesprochen wurde, liegt in der Verträglichkeit der unendlichen Failuresemantik mit den anderen für TCSP eingeführten Semantiken. Aufgrund der analogen Definition der Semantik der Operatoren ist zu erwarten, daß die unendliche Failuresemantik auch die meisten algebraischen Gesetze und die Regeln des sat-Kalküls für TCSP respektiert. Dennoch sind auch hier weitere Untersuchung in diese Richtung notwendig.

Anhang B

Beweise

In Anhang A wurde die unendliche Failuresemantik $\mathcal{I}$ als Erweiterung der endlichen Failuresemantik eingeführt sowie ihre Vorzüge gegenüber der endliche Failuresemantik erläutert. Gleichzeitig wurde in Satz A.3.1 auf die Verträglichkeit der unendlichen mit der endlichen Failuresemantik verwiesen: die Eigenschaften von Prozessen der endlichen Failuresemantik wie in Definition A.2.2 beschrieben gelten ebenfalls für die in dieser Arbeit verwendeten Systeme sequentieller Prozesse.

Ziel dieses Kapitels ist es, diese Aussage formal nachzuweisen. Dabei wird der Beweis in drei Schritten geführt:

1. Zuerst wird gezeigt, daß sich die in Definition A.3.3 eingeführten Systeme in einer einheitliche Struktur beschreiben lassen. Wesentlich ist dabei, daß innerhalb dieser Struktur der Abstraktionsoperator nur einmal angewendet wird.
2. Im zweiten Schritt wird mit der denotationellen Semantik $\mathcal{C}$ eine Erweiterung der unendlichen Failuresemantik $\mathcal{I}$ eingeführt. Diese Erweiterung erlaubt es, die in Abschnitt A.4.3 erläuterten und mittels $\mathcal{I}$ nur indirekt beschreibbaren Fairnesskonzepte des unendlichen Failuremodells explizit darzustellen. Für $\mathcal{C}$ werden einige Abschlußeigenschaften nachgewiesen.
3. Im letzten Schritt werden aus den Abschlußeigenschaften von $\mathcal{C}$ entsprechende Eigenschaften von $\mathcal{I}$ abgeleitet und damit die Verträglichkeit der endlichen und der unendlichen Failuresemantik nachgewiesen.

Diese Eigenschaften sind, wie in Definition A.3.1 gezeigt - identisch mit den Eigenschaften der endlichen Failuresemantik; damit ist die in Satz A.3.1 geforderte Verträglichkeit nachgewiesen.

Darüber hinaus gelten in der unendlichen Failuresemantik weitere Eigenschaften, die in der endlichen Failuresemantik nicht formuliert werden können (vgl. Definition A.3.1). Diese werden im Anschluß an die obigen Abschlußeigenschaften und auf diese aufbauend nachgewiesen.

Entsprechend ergibt sich damit der Aufbau dieses Kapitels:

- In Abschnitt B.1 wird gezeigt, daß es für die in dieser Arbeit betrachteten Systeme sequentieller Prozesse ausreicht, die oben erwähnte einfache syntaktische Struktur zu verwenden.
- Die komplexe unendliche Failuresemantik $\mathcal{C}$ für Prozeßterme ohne die Abstraktion wird in Abschnitt B.2 eingeführt, ihre Eigenschaften in B.2.1 sowie der Nachweis der Abschlußeigenschaften in B.2.3 geführt.
- Der Bezug zwischen der komplexen unendlichen Failuresemantik $\mathcal{C}$ sowie der unendlichen Failuresemantik wird in den Abschnitten B.2.4 und B.2.5 hergestellt.
- In Abschnitt B.3 wird eine Eigenschaft der unendlichen und der erweiterten unendlichen Failuresemantik nachgewiesen, die im endlichen Failuremodell nicht ausdrückbar ist, nämlich die *Terminierbarkeit* von Prozessen.
- Die Terminierbarkeit wird in Abschnitt B.4.1 eingesetzt, um die von der endlichen Failuresemantik bekannten Abschlußeigenschaften der unendlichen Failuresemantik für Prozeßtermen mit einer Anwendung des Abstraktionsoperators nachzuweisen.
- Aufbauend auf diesen Abschlußeigenschaften werden abschließend in Abschnitt B.4.2 die zusätzlichen Eigenschaften nachgewiesen, die für das unendliche Failuremodell gelten und grundlegende Konzepte des Modells illustrieren.

Schließlich finden sich in Abschnitt B.5 Hilfsätze, die für die Beweisführung der vorangegangenen Abschnitte benötigt werden, nicht jedoch spezifisch zum eigentlichen Beweis beitragen.

B.1 Systemstrukturen

Ziel dieses Abschnittes ist es, zu zeigen, daß die in Definition A.3.3 beschriebenen Systeme sequentieller Systeme in einer einheitlichen syntaktischen Struktur dargestellt werden können. Wesentlich ist dabei, daß der Abstraktionsoperator innerhalb dieser Struktur nur einmal angewendet wird.

Im folgenden werden zur Verkürzung der Schreibweise die Konstruktionsoperatoren wie die Parallelkomposition $\|$ direkt auf Prozesse anstatt auf Prozeßterme angewendet. Dabei wird diesen Operatoren die Semantik der entsprechenden syntaktischen Operatoren zugeordnet. Dies erlaubt es, eine einfachere Formulierung der gewünschten Aussagen in Form algebraischer Eigenschaften über diese Operatoren zu verwenden.

Für den Nachweis der eigentlichen Aussage wird neben den in Abschnitt B.5.1 gezeigte Hilfsätzen noch die folgenden Definitionen benötigt.

Definition B.1.1 (Elementweise Erweiterung) Sei $f : A \to B$ eine Abbildung zwischen zwei Alphabeten A und B. Dann heißt $\bar{f} : \mathbb{P}(A) \to \mathbb{P}(B)$ mit

$$\bar{f}(S) \stackrel{\text{def}}{=} \{f(s) \mid s \in S\}$$

die *elementweise Erweiterung der Alphabetsfunktion* f. ∘

Die elementweise Erweiterung einer Alphabetsfunktion entspricht damit der punktweisen Erweiterung aus Definition 2.2.1 auf mengenwertige Argumente.

Mit dieser Definition läßt sich nun zeigen, daß es bei dem Nachweis der Prozeßeigenschaften für Systeme sequentieller synchroner Prozesse ausreicht, Systeme mit einer bestimmten Struktur S zu betrachten:

Satz B.1.1 (Systemstruktur) Zu jedem Prozeß beschrieben durch ein System sequentieller Prozesse existiert ein identischer Prozeß beschrieben durch ein System S mit

- System S entsteht mittels Verbergen aus System S':

  $S_1 = S_2 \backslash B$

- System S_2 wird durch Parallelkomposition und Umbenennung sequentieller Prozesse gebildet:

  $S_2 = f_1(P_1) \parallel \ldots \parallel f_n(P_n)$

  wobei $P_1, \ldots, P_n$ sequentielle Prozesse sind.

•

Die wesentliche Aussage ist dabei, daß jedes System sequentieller synchroner Prozesse durch ein entsprechendes System *mit nur einem Verbergeoperator* ersetzt werden kann.

Beweis B.1.1 (Satz B.1.1) Die Aussage wird mittels Induktion über den Aufbau von Systemen sequentieller synchron kommunizierender Prozesse durch die Operatoren

P: Verwendung eines sequentiellen Prozesses P

$f(P)$: Anwendung der Umbenennung f auf das System P

$P\backslash C$: Anwendung des Verbergens von Aktionen aus C auf das System P

$P_1 \parallel P_2$: Anwendung der Parallelkomposition der Systeme P_1 und P_2

gezeigt. Für die Induktion werden - entsprechend dem Aufbau - die folgenden Fälle betrachtet:

P: Für die sequentiellen Prozesse folgt die Aussage mittels

$P = \mathsf{id}(P)\backslash\emptyset$

also durch Anwendung der Identitätsfunktion id und Verbergen der leeren Aktionsmenge.[1]

$f(P)$: Entsprechend der Induktionsannahme liegt P in der Form

$P = P'\backslash C$

vor. Durch Verwendung einer injektiven Funktion g auf A mit $\forall a \in A\backslash C.f(a) = g(a)$, wobei $A = \alpha P'$, kann P in die gewünschte Form gebracht werden:

$f(P)$

[1] Für das verwendete Gesetz $P\backslash\emptyset = P$ siehe z.B. [Hoa85].

$=$ [Definition g, Satz B.5.5]
$g(P)$
$=$ [Definition P]
$g(P'\backslash C)$
$=$ [Definition g, Satz B.18]
$g(P')\backslash\bar{g}(C)$

und damit liegt $f(P)$ mit anschließender Anwendung von Satz B.5.3 in der gewünschten Form vor.

$P\backslash C$: Entsprechend der Induktionsannahme liegt P in der Form

$$P = P'\backslash C' \tag{B.1}$$

vor. Mittels Satz B.5.2 folgt dann

$P\backslash C$
$=$ [Definition P]
$(P'\backslash C')\backslash C$
$=$ [Satz B.5.2]
$P'\backslash(C' \cup C)$

und damit liegt $P\backslash C$ in der gewünschten Form vor.

$P_1 \parallel P_2$: Entsprechend der Induktionsannahme liegen P_1 und P_2 in der Form

$$P_1 = Q_1\backslash C_1$$

und

$$P_2 = Q_2\backslash C_2$$

vor. Durch Verwendung zweier injektiver Funktionen f und g mit

- $\forall a \in A_1\backslash C_1.f(a) = a$ und $\bar{f}(C_1) \cap A_2 = \emptyset$
- $\forall a \in A_2\backslash C_2.g(a) = a$ und $\bar{g}(C_2) \cap A_1 = \emptyset$
- $\bar{f}(C_1) \cap \bar{g}(C_2) = \emptyset$

– wobei $A_1 = \alpha Q_1$ und $A_2 = \alpha Q_2$ – kann $P_1 \parallel P_2$ in die gewünschte Form gebracht werden. Aufgrund der Definitionen gilt

$\alpha f(Q_1) \cap (\bar{f}(C_1) \cup \bar{g}(C_2)$
$=$ [Prädikatenlogik]
$(\alpha f(Q_1) \cap \bar{f}(C_1)) \cup (\alpha f(Q_1) \cap \bar{g}(C_2))$
$=$ [Definition $\alpha f(Q_1)$]
$(\bar{f}(A_1) \cap \bar{f}(C_1)) \cup (\bar{f}(A_1) \cap \bar{g}(C_2))$
$=$ [Definition $\bar{f}$, $\bar{g}$]
$\bar{f}(A_1 \cap C_1) \cup ((\bar{f}(A_1\backslash C_1) \cup \bar{f}(C_1)) \cap \bar{g}(C_2))$
$=$ [Prädikatenlogik]
$\bar{f}(A_1 \cap C_1) \cup (\bar{f}(A_1\backslash C_1) \cap \bar{g}(C_2)) \cup (\bar{f}(C_1) \cap \bar{g}(C_2))$
$=$ [Definition $\bar{f}$]
$\bar{f}(C_1) \cup ((A_1\backslash C_1) \cap \bar{g}(C_2)) \cup (\bar{f}(C_1) \cap \bar{g}(C_2))$

$= $ [Definition $\bar{f}$, $\bar{g}$]
$\bar{f}(C_1) \cup \emptyset \cup \emptyset$
$= $ [Prädikatenlogik]
$\bar{f}(C_1)$

und damit bei entsprechend Nachweis für Q_2 insgesamt

$$\alpha f(Q_1) \cap (\bar{f}(C_1) \cup \bar{g}(C_2)) = \bar{f}(C_1) \wedge \alpha f(Q_2) \cap (\bar{f}(C_1) \cup \bar{g}(C_2)) = \bar{g}(C_2) \qquad \text{(B.2)}$$

Damit folgt

$P_1 \parallel P_2$
$\Longrightarrow$ [Definition P_1, P_2]
$(Q_1 \backslash C_1) \parallel (Q_2 \backslash C_2)$
$\Longrightarrow$ [Definition f, g, Satz B.5.5]
$f(Q_1 \backslash C_1) \parallel g(Q_2 \backslash C_2)$
$\Longrightarrow$ [Definition f, g, Satz B.18]
$(f(Q_1) \backslash \bar{f}(C_1)) \parallel (g(Q_2) \backslash \bar{g}(C_2))$
$\Longrightarrow$ [Eigenschaft B.2]
$(f(Q_1) \backslash (\alpha f(Q_2) \cap (\bar{f}(C_1) \cup \bar{g}(C_2)))) \parallel (g(Q_2) \backslash (\alpha f(Q_2) \cap (\bar{f}(C_1) \cup \bar{g}(C_2))))$
$\Longrightarrow$ [Satz B.5.1]
$(f(Q_1) \parallel g(Q_2)) \backslash (\bar{f}(C_1) \cup \bar{g}(C_2)$

und damit liegt $P = P_1 \parallel P_2$ mit anschließender Anwendung von Satz B.5.3 in der gewünschten Form vor.

□

B.2 Erweitertes semantische Modell

Ziel des folgenden Abschnitts ist es, die Abschlußeigenschaften des unendlichen Failuremodells nachzuweisen. Dabei werden diese Eigenschaften nicht für beliebige Prozesse der unendlichen Failuresemantik nachgewiesen, sondern ausschließlich für Prozesse mit zwei Einschränkungen:

Endlichkeit des Alphabets: Die Menge der beobachtbaren Ereignisse (Aktionen) eines Prozesses ist beschränkt; diese Einschränkung wird bereits bei [Hoa85] erhoben.

Endlichkeit des Systems: Der Prozeß beschreibt ein System sequentieller, synchron kommunizierender Prozesse.

Diese Beschränkung erlaubt es, die anfallenden Beweisaufgaben auf ein im Rahmen dieser Arbeit behandelbares Maß zu reduzieren. Gleichzeitig ist diese Einschränkung für die in dieser Arbeit beschriebene Vorgehensweise unkritisch: die für diese Arbeit relevanten synchronen Prozesse stellen Systeme mit endlichen Ressourcen dar, wie sie beispielsweise mittels `occam`® beschrieben werden können. Solche Systeme besitzen natürlicherweise nur eine beschränkte Anzahl von Systemprozessen sowie eine beschränkte Anzahl von elementaren Kommunikationsereignissen.

Der Nachweis der Abschlußeigenschaften für Prozesse der unendlichen Failuresemantik $\mathcal{I}$ ist nicht direkt möglich. Daher wird in Abschnitt B.2.1 zuerst ein erweitertes unendliches Modell eingeführt. Dieses Modell und die in Abschnitt B.2.2 eingeführte, darauf aufbauende komplexe unendliche Failuresemantik $\mathcal{C}$ machen die in der unendlichen Failuresemantik $\mathcal{I}$ nur implizit vorhandenen Fairnesskonzepte explizit. Damit wird der in Abschnitt B.2.3 geführte Nachweis der Gültigkeit der Abschlußeigenschaften für Prozesses des komplexen Failuremodells möglich. Abschließend wird in Abschnitt B.2.4 gezeigt, daß die unendliche Failuresemantik $\mathcal{I}$ eine Abstraktion der komplexen Failuresemantik $\mathcal{C}$ darstellt, und damit - wie in Abschnitt B.2.5 gezeigt - die Prozeßeigenschaften der unendlichen Failuresemantik aus den Abschlußeigenschaften der komplexen Failuresemantik folgen.

B.2.1 Komplexes Failuremodell

Für den Nachweis der gewünschten Eigenschaften der unendlichen Failuresemantik $\mathcal{I}$ wird - wie oben erläutert - ein komplexeres semantisches Modell benötigt. Dazu werden im folgenden das Modell $\mathcal{C}$ sowie die dazugehörige denotationelle Semantik für Prozeßterme eingeführt. Dabei erweitert dieses semantische Modell das unendliche Failuremodell. Ein Element von $\mathcal{C}$ besteht aus zwei Anteilen (t, F), wobei

- $t \in A^\omega$ wie auch im unendlichen Failuremodell einen - möglicherweise - unendlichen Ablauf des beschriebenen Systems darstellt, und
- $F \subseteq A^\omega \times \mathbb{P}(A)$ eine Relation (r, R) über Anteile r des Ablaufs t und von r fair behandelte Mengen von Aktionen beschreibt.

Die Relation F wird verwendet, um eine der *ready trace* bzw. *failure trace* ähnliche Beschreibung von Prozessen zu erhalten (vgl. [BBK87], [vG96]). Die partielle Relation F, bestehend aus Paaren (r, R), ordnet einem Präfix r von t Mengen von Aktionen R zu, wobei in R diejenigen Aktionen enthalten sind, die nach dem Ablauf r bis zum Ende von t fair behandelt werden. Dabei werden Aktionen, wie in Abschnitt A.4.3 beschrieben, als fair behandelt betrachtet, wenn sie im Sinne einer *überlappenden Semantik* fair behandelt werden. Damit wird in dieser Arbeit mit $\mathcal{C}$ eine Semantik entwickelt, die eine explizite Darstellung des in [Fra86] eingeführten Begriffs der Überlappungssemantik erlaubt.

Da das Failuremodell durch Potenzmengenbildung definiert ist, läßt sich eine einfache Bereichskonstruktion verwenden. Wie allgemein alle Potenzmengen zusammen mit der umgekehrten Inklusionsbeziehung, so bildet auch $\mathcal{C}$ mit $\supseteq$ als Ordnung und $A^\omega \times \mathbb{P}(A^\omega \times \mathbb{P}(A))$ als kleinstem Element eine vollständige Halbordnung.[2]

Ein Prozeß wird im komplexen Failuremodell $\mathcal{C}$ beschrieben durch eine Menge C von Tupeln mit

$$C \subseteq A^\omega \times \mathbb{P}(A^\omega \times \mathbb{P}(A))$$

Darüber hinaus erfüllt ein Prozeß noch zusätzliche Eigenschaften:

[2] Vgl. [Win93].

Nichttrivialität: Prozesse weisen immer ein Verhalten, nämlich den leeren Ablauf, auf:

$$(\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C$$

Präfixabschluß: Jeder Präfix eines möglichen Ablaufs ist ein möglicher Ablauf eines Prozesses:

$$(r \circ s, F) \in C \Rightarrow (r, \{(r, \emptyset)\}) \in C$$

Spurbeschränkung: Der Ablaufanteil eines Fairnesspaares ist ein Teilablauf des Ablaufs des Prozesses:

$$(t, \{(r, R)\}) \in C \Rightarrow r \sqsubseteq t$$

Teilmengenabschluß: Können einer Menge von Teilabläufen fair behandelte Aktionen zugeordnet werden, dann auch jeder Teilmenge davon:

$$(t, F) \in C \wedge G \subseteq F \Rightarrow (t, G) \in C$$

Fairnessteilmengenabschluß: Wird nach einem Ablauf eine Menge von Aktionen fair behandelt, so wird auch jede Teilmenge davon fair behandelt:

$$(t, F) \in C \wedge (r, R) \in F \wedge S \subseteq R \Rightarrow (t, F \cup \{(r, S)\}) \in C$$

Erweiterbarkeit: Jeder Ablauf kann um eine Aktion erweitert werden, falls diese Aktion nicht abgelehnt werden kann:

$$(t, F) \in C \Rightarrow ((t, F \cup \{(t, \{a\})\}) \in C \vee (t \circ a, F) \in C)$$

Dabei kann ohne Beschränkung der Allgemeinheit $t \in A^*$ vorausgesetzt werden, da die Aussage für $t \in A^\infty$ entsprechend der Definition von $\circ$ trivial ist.

Fairnesspräfixabschluß: Die leere Aktionsmenge wird von jedem Teilablauf fair behandelt:

$$(r \circ s, F) \in C \Rightarrow (r \circ s, F \cup \{(r, \emptyset)\}) \in C$$

Fairnessereignisabschluß: Wird eine Aktion in einem Ablauf ausgeführt, so wird es bis zu seiner Ausführung fair behandelt:

$$(t, F) \in C \wedge (r \circ a, R) \in F \Rightarrow (t, F \cup \{(r, \{a\})\}) \in F$$

Dabei sei $r \in A^*$ und $a \in A$.

Fairnessrefusalabschluß: Eine Aktionsmenge, die von einem Ablauf fair behandelt wird, wird auch von einem Teilablaufs des Ablaufs fair behandelt.

$$(t, F) \in C \wedge (r \circ s, R) \in C \Rightarrow (t, F \cup \{(r, R))\}) \in C$$

Dabei kann ohne Beschränkung der Allgemeinheit $r \in A^*$ vorausgesetzt werden, da die Aussage für $r \in A^\infty$ entsprechend der Definition von $\circ$ trivial ist.

Fairnessvereinigungsabschluß: Werden zwei Aktionsmengen von einem Teilablauf fair behandelt, so wird auch deren Vereinigung fair behandelt:

$$(t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F \Rightarrow (t, F \cup \{(r, R_1 \cup R_2)\}) \in F$$

Spurkettenabschluß: Prozesse sind ablaufstetig und enthalten mit allen (endlichen) Teilabläufen auch den gesamten Ablauf:

$$(\forall i.(t_i, F) \in C) \Rightarrow (\operatorname{\mathsf{lub}}_i t_i, F) \in C$$

wobei $\forall i.t_i \sqsubseteq t_{i+1}$.

Kettenabschluß: Wird eine Menge von Aktionen für alle Teilabläufe eines Ablaufs fair behandelt, so wird diese Aktionsmenge auch vom gesamten Ablauf fair behandelt:

$$(\forall i.(t, \{(t_i, R)\}) \in C) \Rightarrow (t, \{(\operatorname{\mathsf{lub}}_i t_i, R)\}) \in C$$

wobei $\forall i.t_i \sqsubseteq t_{i+1}$.

B.2.2 Komplexe Failuresemantik

Neben der Einführung des Bereichs wird für die Definition der komplexen Failuresemantik die Abbildung der Prozeßterme in den Bereich $\mathcal{C}$ angegeben. Im folgenden bezeichnet $A = \alpha P$, $A_1 = \alpha P_1$ und $A_2 = \alpha P_2$:

- $\mathcal{C}[\mathsf{CHAOS}_A] \stackrel{\text{def}}{=} \{(t, F) \mid (r, R) \in F \Rightarrow r \sqsubseteq t\}$
- $\mathcal{C}[\mathsf{STOP}_A] \stackrel{\text{def}}{=} \{(\langle\rangle, F) \mid (t, R) \in F \Rightarrow t = \langle\rangle \wedge R \subseteq A\}$
- $\mathcal{C}[a \rightarrow P] \stackrel{\text{def}}{=} \{(\langle\rangle, F) \mid F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}\} \cup$
  $\{(a \circ t, F') \mid \exists F.F' \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ s, R) \mid (s, R) \in F\} \wedge$
  $(t, F) \in \mathcal{C}[P]\}$
- $\mathcal{C}[P_1 \cup P_2] \stackrel{\text{def}}{=} \mathcal{C}[P_1] \cup \mathcal{C}[P_2]$
- $\mathcal{C}[P_1 \,\square\, P_2] \stackrel{\text{def}}{=} \{(\langle\rangle, F) \mid (\langle\rangle, F) \in \mathcal{C}[P_1] \wedge (\langle\rangle, F) \in \mathcal{C}[P_2]\} \cup$
  $\{(t, F) \mid t \neq \langle\rangle \wedge (t, F) \in \mathcal{C}[P_1]\} \cup$
  $\{(t, F) \mid t \neq \langle\rangle \wedge (t, F) \in \mathcal{C}[P_2]\}$
- $\mathcal{C}[P_1 \parallel P_2] \stackrel{\text{def}}{=} \{(t, F) \mid \exists F_1, F_2.(A_1 \copyright t, F_1) \in \mathcal{C}[P_1] \wedge (A_2 \copyright t, F_2) \in \mathcal{C}[P_2] \wedge$
  $F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge$
  $(A_1 \copyright r, R_1) \in F_1 \wedge$
  $(A_2 \copyright r, R_2) \in F_2\}\}$
- $\mathcal{C}[f(P)] \stackrel{\text{def}}{=} \{(f^*(t), F) \mid \exists F'.(t, F') \in \mathcal{C}[P] \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}\}$
- $\mathcal{C}[P_1 \vartriangle P_2] \stackrel{\text{def}}{=} \mathcal{C}[P_1] \cup$
  $\{(r \circ s, F) \mid \exists F_1, F_2.(r, F_1) \in \mathcal{C}[P_1] \wedge (s, F_2) \in \mathcal{C}[P_2] \wedge$
  $r \in A_1^* \wedge s \neq \langle\rangle \wedge$
  $F \subseteq \{(u, R \cup S) \mid (u, S) \in F_1 \wedge (\langle\rangle, R) \in F_2\} \cup$
  $\{(r \circ v, R) \mid (r, R) \in F_2\}\}$
- $\mathcal{C}[\mu X : A.F(X)] \stackrel{\text{def}}{=} \bigcap_{i \in \mathbb{N}} \mathcal{C}[F^i(\mathsf{CHAOS}_A)]$

Die eingeführten Operatoren der komplexen Failuresemantik sind - wie die entsprechend definierten Operatoren der endlichen und unendlichen Failuresemantik - monoton und stetig hinsichtlich der umgekehrten Inklusionsordnung $\supseteq$. Der Nachweis verläuft entsprechend wie in [BHR84] für die endliche Failuresemantik gezeigt und wird hier nicht geführt.[3]

B.2.3 Nachweise der Abschlußeigenschaften

Im folgenden werden nicht alle in [Hoa85] aufgeführten Operatoren behandelt, um den Beweisanteil auf ein der Arbeit angemessenes Maß zu beschränken. Trotzdem werden alle wesentlichen Konstruktionsprinzipien berücksichtigt:

- Basisprozesse (CHAOS_A und STOP_A)
- sequentielle Prozesse ($a \rightarrow P$)
- konstruktiver Nichtdeterminismus ($P_1 \sqcap P_2$)
- parallele Prozesse ($P_1 \parallel P_2$)
- rekursive Prozesse ($\mu X : A.F(X)$)

Zusätzlich wird der Unterbrechungsoperator $P_1 \triangle P_2$ hinzugenommen, da er - wie in Abschnitt A.5 beschrieben, die Konstruktion unbeschränkt nichtdeterministischer Prozesse erlaubt, und somit die Ausdrucksmächtigkeit der beschreibbaren Prozesse wesentlich erhöht.

Auf die Nachweise für den Operator für angelischen Nichtdeterminismus $P_1 \,[\!]\, P_2$ wird wegen der prinzipiellen Ähnlichkeit mit dem Operator für erratischen Nichtdeterminismus verzichtet.

Mit den geführten Nachweisen gelten damit die Abschlußeigenschaften insbesondere für alle sequentiellen Prozesse, die - laut Definition A.3.3 - aus den obigen Operatoren ohne $P_1 \parallel P_2$ gebildet werden.

Nachweise für CHAOS_A

Der Nachweis aller Abschlußeigenschaften ist trivial, da alle in der Konklusion der Eigenschaften beschriebenen Elemente entsprechend der Definition von CHAOS_A in CHAOS_A enthalten sind.

Der Nachweis der Spurbeschränkung folgt unmittelbar aus der Definition von $\mathcal{C}[\mathsf{CHAOS}_A]$.

Nachweise für STOP_A

In den folgenden Nachweisen wird abkürzend $C = \mathcal{C}[STOP_A]$ verwendet.

[3] Der Abstraktionsoperator $P\backslash H$ ist nichtstetig und daher in der obigen Liste von Operatoren *nicht* enthalten.

Die Nichttrivialität von $STOP_A$ folgt aus

$$
\begin{array}{ll}
 & \emptyset \subseteq A \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (\langle\rangle, \emptyset) \in \{(\langle\rangle, R) \mid R \subseteq A\} \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in \{(\langle\rangle, \{(\langle\rangle, R)\}) \mid R \subseteq A\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_a]] \\
 & (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C
\end{array}
$$

Der Präfixabschluß von STOP_A folgt aus

$$
\begin{array}{ll}
 & (r \circ s, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[STOP_A]] \\
 & r \circ s = \langle\rangle \wedge (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C \\
\Longrightarrow & [\text{Definition } \circ] \\
 & r = \langle\rangle \wedge (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C \\
\Longrightarrow & [r = \langle\rangle] \\
 & (r, \{(r, \emptyset)\}) \in C
\end{array}
$$

Die Spurbeschränkung von STOP_A folgt aus

$$
\begin{array}{ll}
 & (t, \{(r, R)\}) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[STOP_A]] \\
 & t = \langle\rangle \wedge \{(r, R)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & t = \langle\rangle \wedge r = \langle\rangle \\
\Longrightarrow & [\text{Definition } \sqsubseteq] \\
 & r \sqsubseteq t
\end{array}
$$

Der Teilmengenabschluß von STOP_A folgt aus

$$
\begin{array}{ll}
 & (t, F) \in C \wedge G \subseteq F \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\
 & t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \wedge G \subseteq F \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & t = \langle\rangle \wedge G \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\
 & t = \langle\rangle \wedge (\langle\rangle, G) \in C \\
\Longrightarrow & [t = \langle\rangle] \\
 & (t, G) \in C
\end{array}
$$

Der Fairnessteilmengenabschluß von STOP_A folgt aus

$$
\begin{array}{ll}
 & (t, F) \in C \wedge (r, R) \in F \wedge S \subseteq R \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\
 & t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \wedge r = \langle\rangle \wedge R \subseteq A \wedge S \subseteq R \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \wedge r = \langle\rangle \wedge S \subseteq A \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & t = \langle\rangle \wedge F \cup \{(r, S)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\
 & (t, F \cup \{(r, S)\}) \in C
\end{array}
$$

Die Erweiterbarkeit von STOP_A folgt aus

$$
\begin{array}{ll}
 & (t, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\
 & t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \\
\Longrightarrow & [\text{Prädikatenlogik, } a \in A] \\
 & t = \langle\rangle \wedge F \cup \{(\langle\rangle, \{a\})\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\
 & t = \langle\rangle \wedge (\langle\rangle, F \cup \{(\langle\rangle, \{a\})\}) \in C \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (t, F \cup \{(t, \{a\})\} \in C \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (t, F \cup \{(t, \{a\})\} \in C \vee (r \circ a, F) \in C
\end{array}
$$

Der Nachweis des Fairnessereignisabschlusses von STOP_A folgt unmittelbar aus

$$\forall r \in A^*, a \in A.r \circ a \neq \langle\rangle$$

Der Fairnessrefusalabschluß von STOP_A folgt aus

$$
\begin{array}{ll}
 & (t, F) \in C \wedge (r \circ s, R) \in F \\
\Longrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\
 & (t, F) \in C \wedge (r \circ s, R) \in F \wedge r \circ s = \langle\rangle \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (t, F) \in C \wedge (\langle\rangle, R) \in F \wedge r \circ s = \langle\rangle \\
\Longrightarrow & [\text{Definition } \circ] \\
 & r = \langle\rangle \wedge s = \langle\rangle \wedge (t, F) \in F \wedge (\langle\rangle, R) \in F \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (t, F) \in F \wedge (r, R) \in F \\
\Longrightarrow & [\text{Prädikatenlogik}]
\end{array}
$$

$(t, F \cup \{(r, R\}) \in C$

Der Fairnessvereinigungsabschluß von STOP_A folgt aus

$(t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F$
$\Longrightarrow$ [Definition $\mathcal{C}[\mathsf{STOP}_A]$]
$t = \langle\rangle \wedge r = \langle\rangle \wedge R_1 \subseteq A \wedge R_2 \subseteq A$
$\Longrightarrow$ [Prädikatenlogik]
$t = \langle\rangle \wedge r = \langle\rangle \wedge R_1 \cup R_2 \subseteq A$
$\Longrightarrow$ [Definition $\mathcal{C}[\mathsf{STOP}_A]$]
$(t, F \cup \{(r, R_1 \cup R_2)\}) \in C$

Der Spurkettenabschluß von STOP_A folgt aus

$\forall i.(t_i, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[\mathsf{STOP}_A]$]
$\forall i.t_i = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\}$
$\Longrightarrow$ [Definition lub]
$\mathop{\mathsf{lub}}_i t_i = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\}$
$\Longrightarrow$ [Definition $\mathcal{C}[\mathsf{STOP}_A]$]
$(\mathop{\mathsf{lub}}_i t_i, F) \in C$

Der Kettenabschluß von STOP_A folgt aus

$\forall i.(t, \{(t_i, R)\}) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[\mathsf{STOP}_A]$]
$\forall i.(t = \langle\rangle \wedge t_i = \langle\rangle \wedge R \subseteq A)$
$\Longrightarrow$ [Prädikatenlogik]
$t = \langle\rangle \wedge \mathop{\mathsf{lub}}_i t_i = \langle\rangle \wedge \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \subseteq \{(\langle\rangle, R) \mid R \subseteq A\}$
$\Longrightarrow$ [Definition $\mathcal{C}[\mathsf{STOP}_A]$]
$(t, \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \in C$

Nachweise für $P_1 \sqcap P_2$

In den folgenden Nachweisen wird abkürzend $C_1 = \mathcal{C}[P_1]$, $C_2 = \mathcal{C}[P_2]$ sowie $C = \mathcal{C}[P_1 \sqcap P_2]$ verwendet.

Die Nichttrivialität von $P_1 \sqcap P_2$ folgt aus

$\Longrightarrow$ [Induktion]

$$
\begin{array}{l}
\quad (\langle\rangle, (\langle\rangle, \emptyset)) \in C_1 \vee (\langle\rangle, (\langle\rangle, \emptyset)) \in C_2 \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
\quad (\langle\rangle, (\langle\rangle, \emptyset)) \in C
\end{array}
$$

Die Präfixabgeschlossenheit von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{l}
\quad (r \circ s, F) \in C \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
\quad (r \circ s, F) \in C_1 \vee (r \circ s, F) \in C_2 \\
\Longrightarrow [\text{Induktion}] \\
\quad (r, \{(r, \emptyset)\}) \in C_1 \vee (r, \{(r, \emptyset)\}) \in C_2 \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad (r, \{(r, \emptyset)\}) \in C
\end{array}
$$

Die Spurbeschränktheit von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{l}
\quad (t, \{(r, R)\}) \in C \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
\quad (t, \{(r, R)\}) \in C_1 \vee (t, \{(r, R)\}) \in C_2 \\
\Longrightarrow [\text{Induktion}] \\
\quad r \sqsubseteq t \vee r \sqsubseteq t \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad r \sqsubseteq t
\end{array}
$$

Der Teilmengenabschluß von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{l}
\quad (t, F_1) \in C \wedge F_2 \subseteq F_1 \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
\quad ((t, F_1) \in C_1 \vee (t, F_1) \in C_2) \wedge F_2 \subseteq F_1 \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad ((t, F_1) \in C_1 \wedge F_2 \subseteq F_1) \vee ((t, F_1) \in C_2 \wedge F_2 \subseteq F_1) \\
\Longrightarrow [\text{Induktion}] \\
\quad (t, F_2) \in C_1 \vee (t, F_2) \in C_2 \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
\quad (t, F_2) \in C
\end{array}
$$

Der Fairnessteilmengenabschluß von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r, R) \in F \wedge S \subseteq R \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
\quad ((t, F) \in C_1 \vee (t, F) \in C_2) \wedge (r, R) \in F \wedge S \subseteq R
\end{array}
$$

$\Longrightarrow$ [Prädikatenlogik]
$\quad((t, F) \in C_1 \wedge (r, R) \in F \wedge S \subseteq R) \vee$
$\quad((t, F) \in C_2 \wedge (r, R) \in F \wedge S \subseteq R)$
$\Longrightarrow$ [Induktion]
$\quad(t, F \cup \{(r, S)\}) \in C_1 \vee (t, F \cup \{(r, S)\}) \in C_2$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \sqcap P_2]$]
$\quad(t, F \cup \{(r, S)\}) \in C$

Die Erweiterbarkeit von $P_1 \sqcap P_2$ folgt aus

$\quad(t, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \sqcap P_2]$]
$\quad((t, F) \in C_1) \vee (t, F) \in C_2)$
$\Longrightarrow$ [Induktion]
$\quad(t, F \cup \{(t, \{a\})\}) \in C_1 \vee (t \circ a, F) \in C_1) \vee$
$\quad(t, F \cup \{(t, \{a\})\}) \in C_2 \vee (t \circ a, F) \in C_2)$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \sqcap P_2]$]
$\quad(t, F \cup \{(t, \{a\})\}) \in C \vee (t \circ a, F) \in C)$

Der Fairnesspräfixabschluß von $P_1 \sqcap P_2$ folgt aus

$\quad(r \circ s, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \sqcap P_2]$]
$\quad(r \circ s, F) \in C_1 \vee (r \circ s, F) \in C_2$
$\Longrightarrow$ [Induktion]
$\quad(r \circ s, F \cup \{(r, \emptyset)\}) \in C_1 \vee (r \circ s, F \cup \{(r, \emptyset)\}) \in C_2$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \sqcap P_2]$]
$\quad(r \circ s, F \cup \{(r, \emptyset)\}) \in C$

Der Fairnessereignisabschluß von $P_1 \sqcap P_2$ folgt aus

$\quad(t, F) \in C \wedge (r \circ a, R) \in F$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \sqcap P_2]$]
$\quad((t, F) \in C_1 \vee (t, F) \in C_2) \wedge (r \circ a, R) \in F$
$\Longrightarrow$ [Prädikatenlogik]
$\quad((t, F) \in C_1 \wedge (r \circ a, R) \in F) \vee ((t, F) \in C_2 \wedge (r \circ a, R) \in F)$
$\Longrightarrow$ [Induktion]
$\quad(t, F \cup \{(r, \{a\})\}) \in C_1 \vee (t, F \cup \{(r, \{a\})\}) \in C_2$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \sqcap P_2]$]
$\quad(t, F \cup \{(r, \{a\})\}) \in C$

Der Fairnessrefusalabschluß von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{ll}
 & (t, F) \in C \wedge (r \circ s, R) \in F \\
\Longrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
 & ((t, F) \in C_1 \vee (t, F) \in C_2) \wedge (r \circ s, R) \in F \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & ((t, F) \in C_1 \wedge (r \circ s, R) \in F) \vee ((t, F) \in C_2 \wedge (r \circ s, R) \in F) \\
\Longrightarrow & [\text{Induktion}] \\
 & (t, F \cup \{(r, R)\}) \in C_1 \vee (t, F \cup \{(r, R)\}) \in C_2 \\
\Longrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
 & (t, F \cup \{(r, R)\}) \in C
\end{array}
$$

Der Fairnessvereinigungsabschluß von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{ll}
 & (t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F \\
\Longrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
 & ((t, F) \in C_1 \vee (t, F) \in C_2)) \wedge (r, R_1) \in F \wedge (r, R_2) \in F \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & \quad ((t, F) \in C_1 \wedge (r, R_1) \in F \wedge (r, R_2) \in F) \vee \\
 & \quad ((t, F) \in C_2 \wedge (r, R_1) \in F \wedge (r, R_2) \in F) \\
\Longrightarrow & [\text{Induktion}] \\
 & (t, F \cup \{(r, R_1 \cup R_2)\}) \in C_1 \vee (t, F \cup \{(r, R_1 \cup R_2)\}) \in C_2 \\
\Longrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
 & (t, F \cup \{(r, R_1 \cup R_2)\}) \in C
\end{array}
$$

Der Spurkettenabschluß von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{ll}
 & \forall i.(t_i, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
 & \forall i.(t_i, F) \in C_1 \vee (t_i, F) \in C_2 \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (\forall i.(t_i, F) \in C_1) \vee (\forall i.(t_i, F) \in C_2) \\
\Longrightarrow & [\text{Induktion}] \\
 & (\mathop{\mathsf{lub}}_i t_i, F) \in C_1 \vee (\mathop{\mathsf{lub}}_i t_i, F) \in C_2 \\
\Longrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
 & (\mathop{\mathsf{lub}}_i t_i, F) \in C
\end{array}
$$

Der Kettenabschluß von $P_1 \sqcap P_2$ folgt aus

$$
\begin{array}{ll}
 & \forall i.(t, \{(t_i, R)\}) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]]
\end{array}
$$

$$
\begin{array}{l}
\quad \forall i.((t, \{(t_i, R)\}) \in C_1 \vee (t, \{(t_i, R)\}) \in C_2) \\
\Longrightarrow [\text{Prädikatenlogik, Induktion}] \\
\quad (t, \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \in C_1 \vee (t, \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \in C_2 \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\
\quad (t, \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \in C
\end{array}
$$

Nachweise für $a \to P$

Im folgenden wird abkürzend $C = \mathcal{C}[a \to P]$ sowie $C' = \mathcal{C}[P]$ verwendet.

Die Nichttrivialität von $a \to P$ folgt aus

$$
\begin{array}{l}
\quad \emptyset \subseteq A \backslash \{a\} \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad (\langle\rangle, \emptyset) \in \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in \{(\langle\rangle, \{(\langle\rangle, R)\}) \mid R \subseteq A \backslash \{a\}\} \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C
\end{array}
$$

Der Nachweis des Präfixabschlusses von $a \to P$ erfolgt mittels Fallunterscheidung nach $r = \langle\rangle$. Im Fall $r = \langle\rangle$ folgt die Aussage unmittelbar aus der Nichttrivialität von $a \to P$. Anderenfalls gilt

$$
\begin{array}{l}
\quad (r \circ s, F) \in C \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad\ r \circ s = \langle\rangle \vee \\
\quad\ (\exists t, F'.r \circ s = a \circ t \wedge (t, F') \in C') \\
\Longrightarrow [r \neq \langle\rangle] \\
\quad \exists t, F'.r \circ s = a \circ t \wedge (t, F') \in C' \\
\Longrightarrow [\text{Definition } \circ] \\
\quad \exists t, F'.r = a \circ t \wedge (t \circ s, F') \in C' \\
\Longrightarrow [\text{Induktion}] \\
\quad \exists t.r = a \circ t \wedge (t, \{(t, \emptyset)\}) \in C' \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad \exists t.r = a \circ t \wedge (a \circ t, \{(a \circ t, \emptyset)\}) \in C \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad (r, \{(r, \emptyset)\}) \in C
\end{array}
$$

Die Spurbeschränkung von $a \to P$ folgt aus

$$(t, \{r, R)\}) \in C$$

$\Longrightarrow$ [Definition $\mathcal{C}[a \to P]$]
$(t = \langle\rangle \wedge (r, R) \in \{(\langle\rangle, R) \mid \subseteq A\}) \vee$
$(\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge$
$(r, R) \in \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})$
$\Longrightarrow$ [Prädikatenlogik]
$(t = \langle\rangle \wedge r = \langle\rangle) \vee$
$(\exists s.t = a \circ s \wedge r = \langle\rangle \wedge R \subseteq A \backslash \{a\}) \vee$
$(\exists s, u, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge$
$r = a \circ u \wedge (u, R) \in F' \wedge$
$(r, R) \in \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})$
$\Longrightarrow$ [Teilmengenabschluß von C']
$(t = \langle\rangle \wedge r = \langle\rangle) \vee$
$(\exists s.t = a \circ s \wedge r = \langle\rangle \wedge R \subseteq A \backslash \{a\}) \vee$
$(\exists s, u.t = a \circ s \wedge r = a \circ u \wedge (s, \{(u, R)\}) \in C')$
$\Longrightarrow$ [Induktion]
$(t = \langle\rangle \wedge r = \langle\rangle) \vee$
$(\exists s.t = a \circ s \wedge r = \langle\rangle \wedge R \subseteq A \backslash \{a\}) \vee$
$(\exists s, u.t = a \circ s \wedge r = a \circ u \wedge u \sqsubseteq s)$
$\Longrightarrow$ [Prädikatenlogik]
$(t = \langle\rangle \wedge r = \langle\rangle) \vee$
$(\exists s.t = a \circ s \wedge r = \langle\rangle) \vee$
$(\exists s, u.t = a \circ s \wedge r = a \circ u \wedge u \sqsubseteq s)$
$\Longrightarrow$ [Definition $\sqsubseteq$]
$(r \sqsubseteq t) \vee (r \sqsubseteq t) \vee (r \sqsubseteq t)$
$\Longrightarrow$ [Prädikatenlogik]
$r \sqsubseteq t$

Der Fairnessteilmengenabschluß von $a \to P$ folgt aus

$(t, F) \in C \wedge (r, R) \in F \wedge S \subseteq R$
$\Longrightarrow$ [Definition $\mathcal{C}[a \to P]$]
$((t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee$
$(\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge$
$F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge$
$(r, R) \in F \wedge S \subseteq R$
$\Longrightarrow$ [Prädikatenlogik]
$(t = \langle\rangle \wedge r = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \wedge R \subseteq A \backslash \{a\} \wedge S \subseteq R) \vee$
$(\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge$
$F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge$
$((r = \langle\rangle \wedge R \subseteq A) \vee (\exists u.r = a \circ u \wedge (u, R) \in F') \wedge$
$S \subseteq R))$
$\Longrightarrow$ [Prädikatenlogik]

$$
\begin{array}{l}
\quad (t = \langle\rangle \wedge r = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\backslash\{a\}\} \wedge S \subseteq A\backslash\{a\}) \vee \\
\quad (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\quad ((r = \langle\rangle \wedge S \subseteq A) \vee (\exists u.r = a \circ u \wedge (u, R) \in F' \wedge S \subseteq R))) \\
\Longrightarrow [\text{Induktion}] \\
\quad (t = \langle\rangle \wedge r = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\backslash\{a\}\} \wedge S \subseteq A\backslash\{a\}) \vee \\
\quad (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\quad ((r = \langle\rangle \wedge S \subseteq A) \vee (\exists u.r = a \circ u \wedge (u, S) \in F'))) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad (t = \langle\rangle \wedge F \cup \{(r, S)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\backslash\{a\}\}) \vee \\
\quad (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\quad F \cup \{(r, S)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}) \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad (t, F \cup \{(r, S)\}) \in C
\end{array}
$$

Der Teilmengenabschluß von $a \to P$ folgt aus

$$
\begin{array}{l}
\quad (t, F) \in C \wedge G \subseteq F \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad G \subseteq F \wedge \\
\quad ((t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\backslash\{a\}\}) \vee \\
\quad (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \\
\Longrightarrow [G \subseteq F] \\
\quad ((t = \langle\rangle \wedge G \subseteq \{(\langle\rangle, R) \mid R \subseteq A\backslash\{a\}\}) \vee \\
\quad (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\quad G \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad (t, G) \in C
\end{array}
$$

Die Erweiterbarkeit von $a \to P$ um b wird mittels Fallunterscheidung nach den Fällen $t = \langle\rangle$ und $a = b$ gezeigt:

$t = \langle\rangle \wedge a = b$**:** Dann gilt:

$$
\begin{array}{l}
\quad (t, F) \in C \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P],\ t = \langle\rangle] \\
\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\backslash\{a\}\} \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \\
\Longrightarrow [\text{Definition } \circ] \\
\quad a = \langle\rangle \circ a \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\}
\end{array}
$$

$$
\begin{array}{ll}
\Longrightarrow & [\text{Nichttrivialität von } C'] \\
& a = \langle\rangle \circ a \wedge (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C' \wedge \\
& F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ \langle\rangle, \emptyset)\} \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& a = \langle\rangle \circ a \wedge (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C' \wedge \\
& F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in \{(\langle\rangle, \emptyset)\}\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[A \to P]] \\
& (\langle\rangle \circ a, F) \in C \\
\Longrightarrow & [a = b,\, t = \langle\rangle] \\
& (t \circ b, F) \in C \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& (t, F \cup \{(t, \{b\})\}) \in C \vee (t \circ b, F) \in C
\end{array}
$$

$t = \langle\rangle \wedge a \neq b$**:** Dann gilt:

$$
\begin{array}{ll}
& (t, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[a \to P],\, t = \langle\rangle] \\
& F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \\
\Longrightarrow & [a \neq b,\, b \in A] \\
& F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \wedge \{b\} \subseteq A \backslash \{a\} \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& F \cup \{(\langle\rangle, \{b\})\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[a \to P],\, t = \langle\rangle] \\
& (t, F \cup \{(t, \{b\})\} \in C \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& (t, F \cup \{(t, \{b\})\}) \in C \vee (t \circ b, F) \in C
\end{array}
$$

$t \neq \langle\rangle$**:** Dann gilt:

$$
\begin{array}{ll}
& (t, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[a \to P],\, t \neq \langle\rangle] \\
& \exists r, F'.\ t = a \circ r \wedge (r, F') \in C' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\Longrightarrow & [\text{Induktion}] \\
& \exists r, F'.\ t = a \circ r \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
& \qquad ((r \circ b, F') \in C' \vee (r, F' \cup \{(r, \{b\})\}) \in F') \\
\Longrightarrow & [\text{Prädikatenlogik, Definition } \circ] \\
& (\exists r, F'.\ t \circ b = a \circ r \circ b \wedge (r \circ b, F') \in C' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}) \vee \\
& (\exists r, F'.\ t = a \circ r \wedge (r, F' \cup \{(r, \{b\})\}) \in F' \wedge \\
& \qquad F \cup \{(a \circ r, \{b\})\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \cup \\
& \qquad\qquad \{(a \circ u, R) \mid (u, R) \in F'\} \cup \\
& \qquad\qquad \{(a \circ r, \{b\})\}) \\
\Longrightarrow & [\text{Definition } \mathcal{C}[a \to P]]
\end{array}
$$

$$(t \circ b, F) \in C \vee (t, F \cup \{(t, \{b\})\}) \in C$$

Der Nachweis des Fairnesspräfixabschlusses von $a \to P$ folgt mittels Fallunterscheidung nach $r = \langle\rangle$:

$r = \langle\rangle$**:** Dann folgt

$$\begin{array}{ll}
& (r \circ s, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[a \to P]] \\
& (r \circ s = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
& (\exists t, F'.\ r \circ s = a \circ t \wedge (t, F') \in C' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& (r \circ s = \langle\rangle \wedge F \cup \{(\langle\rangle, \emptyset)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
& (\exists t, F'.\ r \circ s = a \circ t \wedge (t, F') \in C' \wedge \\
& \qquad F \cup \{(\langle\rangle, \emptyset)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}) \\
\Longrightarrow & [r = \langle\rangle] \\
& (r \circ s = \langle\rangle \wedge F \cup \{(r, \emptyset)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
& (\exists t, F'.\ r \circ s = a \circ t \wedge (t, F') \in C' \wedge \\
& \qquad F \cup \{(r, \emptyset)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}) \\
\Longrightarrow & [\text{Definition } \mathcal{C}[a \to P]] \\
& (r \circ s, F \cup \{(r, \emptyset)\}) \in C
\end{array}$$

$r \neq \langle\rangle$**:** Dann folgt $r \circ s \neq \langle\rangle$ und damit

$$\begin{array}{ll}
& (r \circ s, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[a \to P]] \\
& (r \circ s = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
& (\exists t, F'.\ r \circ s = a \circ t \wedge (t, F') \in C' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}) \\
\Longrightarrow & [r \circ s \neq \langle\rangle] \\
& \exists t, F'.\ r \circ s = a \circ t \wedge (t, F') \in C' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\Longrightarrow & [\text{Definition } \circ] \\
& \exists t, F'.\ r \circ s = a \circ t \wedge (t, F') \in C' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
& \qquad \exists u.r = a \circ u \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& \exists u, F'.\ r = a \circ u \wedge (u \circ s, F') \in C' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\Longrightarrow & [\text{Induktion}] \\
& \exists u, F'.\ r = a \circ u \wedge (u \circ s, F') \in C' \wedge (u, \emptyset) \in F' \wedge \\
& \qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\Longrightarrow & [\text{Prädikatenlogik, } t = u \circ s] \\
& \exists t, F'.\ t = a \circ s \wedge (t, F') \in C' \wedge \\
& \qquad F \cup \{(r, \emptyset)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}
\end{array}$$

$$\begin{array}{l}
\quad \Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\qquad (r \circ s, F \cup \{(r, \emptyset)\}) \in C
\end{array}$$

Der Nachweis des Fairnessereignisabschlusses von $a \to P$ folgt aus

$$\begin{array}{l}
\quad (t, F) \in C \wedge (r \circ b, R) \in F \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\qquad (\ (t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
\qquad\ (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \wedge \\
\qquad (r \circ b, R) \in F \\
\Longrightarrow [\text{Prädikatenlogik}, r \circ b \neq \langle\rangle] \\
\qquad (\exists s, F'.\ t = a \circ s \wedge (a \circ s, F') \in C' \wedge \\
\qquad\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}) \wedge \\
\qquad (r \circ b, R) \in F
\end{array}$$

Der weitere Nachweis folgt mittels Fallunterscheidung nach $r = \langle\rangle$ und $r \neq \langle\rangle$:

$r = \langle\rangle$**:** Dann folgt mit $r \circ b = b$

$$\begin{array}{l}
\qquad\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\quad (b, R) \in F \\
\qquad \Longrightarrow [\text{Prädikatenlogik}] \\
\qquad\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad\qquad F \cup \{\langle\rangle, \{b\}\}\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\qquad \Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\qquad\quad (t, F \cup \{(\langle\rangle, \{b\})\}) \in C \\
\qquad \Longrightarrow [r = \langle\rangle] \\
\qquad\quad (t, F \cup \{(r, \{b\})\}) \in C
\end{array}$$

$r \neq \langle\rangle$**:** Dann folgt

$$\begin{array}{l}
\qquad\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \wedge \\
\qquad\quad (r \circ b, R) \in F \\
\qquad \Longrightarrow [\text{Prädikatenlogik}] \\
\qquad\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \wedge \\
\qquad\quad \exists u.r = a \circ u \wedge (u \circ b, R) \in F' \\
\qquad \Longrightarrow [\text{Induktion}] \\
\qquad\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C \wedge \\
\qquad\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \wedge \\
\qquad\quad \exists u.r = a \circ u \wedge (s, F' \cup \{(u, \{b\})\}) \in C' \\
\qquad \Longrightarrow [\text{Prädikatenlogik}]
\end{array}$$

$$
\begin{array}{l}
\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C \wedge r = a \circ u \wedge \\
\qquad\qquad F \cup \{(r, \{b\})\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad (t, F \cup \{(r, \{b\})\}) \in C
\end{array}
$$

Der Nachweis des Fairnessrefusalabschlusses ist trivial für $s = \langle\rangle$, denn dann folgt die Behauptung unmittelbar aus $r \circ \langle\rangle = r$. Im Fall $s \neq \langle\rangle$ folgt entsprechend der Definition von $\circ$ auch $r \circ s \neq \langle\rangle$ und damit

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r \circ s, R) \\
\Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\quad (\ (t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
\qquad (\exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \wedge \\
\quad (r \circ s, R) \\
\Longrightarrow [\text{Prädikatenlogik}, r \circ s \neq \langle\rangle] \\
\quad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\qquad (r \circ s, R) \in F
\end{array}
$$

Der weitere Nachweis wird mittels Fallunterscheidung nach $r = \langle\rangle$ geführt:

$r = \langle\rangle$**:** Damit folgt

$$
\begin{array}{l}
\qquad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\qquad (r \circ s, R) \in F \\
\quad \Longrightarrow [\text{Prädikatenlogik, } R \subseteq A] \\
\qquad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \cup \{(\langle\rangle, R)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\quad \Longrightarrow [r = \langle\rangle] \\
\qquad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \cup \{(r, R)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\quad \Longrightarrow [\text{Definition } \mathcal{C}[a \to P]] \\
\qquad (t, F \cup \{(r, R)\}) \in C
\end{array}
$$

$r \neq \langle\rangle$**:** Damit folgt

$$
\begin{array}{l}
\qquad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\qquad (r \circ s, R) \in F \\
\quad \Longrightarrow [\text{Prädikatenlogik}, r \neq \langle\rangle] \\
\qquad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\qquad \exists u. r = a \circ u \wedge (u \circ s, R) \in F'
\end{array}
$$

$$
\begin{array}{l}
\Longrightarrow \text{[Induktion]} \\
\quad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\qquad \exists u.r = a \circ u \wedge (u, R) \in F' \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad \exists t', F'.\ t = a \circ t' \wedge (t', F') \in C' \wedge \\
\qquad\qquad F \cup \{(r, R)\}) \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\
\Longrightarrow \text{[Definition } \mathcal{C}[a \to P]\text{]} \\
\quad (t, F \cup \{(r, R)\}) \in C
\end{array}
$$

Der Fairnessvereinigungsabschluß erfolgt mittels Fallunterscheidung nach $r = \langle\rangle$:

$r = \langle\rangle$**:** Damit folgt

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F \\
\Longrightarrow \text{[Definition } \mathcal{C}[a \to P]\text{]} \\
\quad (\ (t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
\quad\ (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \wedge \\
\quad (r, R_1) \in F \wedge (r, R_2) \in F \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad (t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\} \wedge R_1 \subseteq A \backslash \{a\} \wedge R_2 \subseteq A \backslash \{a\}) \vee \\
\quad (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\qquad R_1 \subseteq A \wedge R_2 \subseteq A) \\
\Longrightarrow \text{[Prädikatenlogik, } r = \langle\rangle\text{]} \\
\quad (t = \langle\rangle \wedge F \cup \{(r, R_1 \cup R_2)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
\quad (\ \exists s, F'.\ \ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad F \cup \{(r, R_1 \cup R_2)\} \subseteq \ \{(\langle\rangle, R) \mid R \subseteq A\} \cup \\
\qquad\qquad\qquad\qquad\qquad \{(a \circ u, R) \mid (u, R) \in F'\}) \\
\Longrightarrow \text{[Definition } \mathcal{C}[a \to P]\text{]} \\
\quad (r \circ s, F \cup \{(r, R_1 \cup R_2)\}) \in C
\end{array}
$$

$r \neq \langle\rangle$**:** Damit folgt

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F \\
\Longrightarrow \text{[Definition } \mathcal{C}[a \to P]\text{]} \\
\quad (\ (t = \langle\rangle \wedge F \subseteq \{(\langle\rangle, R) \mid R \subseteq A \backslash \{a\}\}) \vee \\
\quad\ (\exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\})) \wedge \\
\quad (r, R_1) \in F \wedge (r, R_2) \in F \\
\Longrightarrow \text{[Prädikatenlogik,} r \neq \langle\rangle\text{]} \\
\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge \\
\qquad\qquad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge \\
\qquad\qquad (r, R_1) \in F \wedge (r, R_2) \in F
\end{array}
$$

$$
\begin{aligned}
&\Longrightarrow \text{[Prädikatenlogik, } r \neq \langle\rangle\text{]}\\
&\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge\\
&\qquad\quad \exists u.r = a \circ u \wedge (u, R_1) \in F' \wedge (u, R_2) \in F'\\
&\Longrightarrow \text{[Induktion]}\\
&\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \wedge\\
&\qquad\quad \exists u.r = a \circ u \wedge (u, R_1 \cup R_2) \in F'\\
&\Longrightarrow \text{[Prädikatenlogik]}\\
&\quad \exists s, F'.\ t = a \circ s \wedge (s, F') \in C' \wedge\\
&\qquad\quad F \cup \{(r, R_1 \cup R_2)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup\\
&\qquad\qquad\qquad\qquad \{(a \circ u, R) \mid (u, R) \in F'\} \wedge\\
&\Longrightarrow \text{[Definition } \mathcal{C}[a \to P]\text{]}\\
&\quad (t, F \cup \{(r, R_1 \cup R_2)\}) \in C
\end{aligned}
$$

Der Nachweis der Spurkettenvollständigkeit von $a \to P$ folgt unmittelbar für $\forall i.t_i = \langle\rangle$. Anderenfalls folgt $\exists j.t_j \neq \langle\rangle$ und wegen $\forall i.t_i \sqsubseteq t_{i+1}$ weiterhin $\exists j.\forall i.t_{i+j} \neq \langle\rangle$ und damit

$$
\begin{aligned}
&\quad \exists j.\forall i.t_{i+j} \neq \langle\rangle \wedge (t_{i+j}, F) \in C\\
&\Longrightarrow \text{[Definition } \mathcal{C}[a \to P]\text{, Prädikatenlogik]}\\
&\quad \exists j.\forall i.\exists s_{i+j}, F'.\ a \circ s_{i+j} = t_{i+j} \wedge (s_{i+j}, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[Teilmengenabschluß von C']}\\
&\quad \exists j.\forall i.\exists s_{i+j}, F'.\ a \circ s_{i+j} = t_{i+j} \wedge (s_{i+j}, F') \in C' \wedge\\
&\qquad\quad F = \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[Prädikatenlogik]}\\
&\quad \exists j, F'.\forall i.\exists s_{i+j}.\ a \circ s_{i+j} = t_{i+j} \wedge (s_{i+j}, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[Prädikatenlogik]}\\
&\quad \exists j, F', s.\forall i.\ a \circ s_{i+j} = t_{i+j} \wedge (s_{i+j}, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[}\circ \text{ stetig im zweiten Argument]}\\
&\quad \exists j, F', s.\forall i.\ a \circ \mathsf{lub}_i\, s_{i+j} = \mathsf{lub}_i\, t_{i+j} \wedge (s_{i+j}, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[Induktion]}\\
&\quad \exists j, F', s.\ a \circ \mathsf{lub}_i\, s_{i+j} = \mathsf{lub}_i\, t_{i+j} \wedge (\mathsf{lub}_i\, s_{i+j}, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[Prädikatenlogik]}\\
&\quad \exists j, F', s.\ a \circ s = \mathsf{lub}_i\, t_{i+j} \wedge (s, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[Definition } \mathsf{lub}\text{]}\\
&\quad \exists s, F'.\ a \circ s = \mathsf{lub}_i\, t_i \wedge (s, F') \in C' \wedge\\
&\qquad\quad F \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\}\\
&\Longrightarrow \text{[Definition } \mathcal{C}[a \to P]\text{]}
\end{aligned}
$$

$$(\operatorname{\mathsf{lub}}_i t_i, F) \in C$$

Der Nachweis des Kettenabschlusses von $a \to P$ erfolgt mittels Fallunterscheidung nach $\mathsf{lub}_i\, t_i = \langle\rangle$ geführt:

$\mathsf{lub}_i\, t_i = \langle\rangle$**:** Dann folgt wegen $\forall i.t_i \sqsubseteq \mathsf{lub}_i\, t_i$ auch $\forall i.t_i = \langle\rangle$ und damit

$$\begin{array}{ll} & \forall i.(t, \{(t_i, R)\}) \in C \\ \Longrightarrow & [t_i = \langle\rangle] \\ & \forall i.(t, \{(\langle\rangle, R)\})\} \in C \\ \Longrightarrow & [\text{Prädikatenlogik}] \\ & (t, \{(\langle\rangle, R)\})\} \in C \\ \Longrightarrow & [\mathsf{lub}_i\, t_i = \langle\rangle] \\ & (t, \{(\operatorname{\mathsf{lub}}_i t_i, R)\})\} \in C \end{array}$$

$\mathsf{lub}_i\, t_i \neq \langle\rangle$**:** Dann folgt wegen $\exists i.t_i \neq \langle\rangle$ und $\forall i.t_i \sqsubseteq t_{i+1}$ auch $\exists i.\forall j.t_{i+j} \neq \langle\rangle$, und es folgt

$$\begin{array}{ll} & \forall i.(t, \{(t_i, R)\}) \in C \\ \Longrightarrow & [\exists i.\forall j.t_{i+j} \neq \langle\rangle] \\ & \exists i.\forall j.t_{i+j} \neq \langle\rangle \wedge (t, \{(t_{i+j}, R)\}) \in C \\ \Longrightarrow & [\text{Definition } \mathcal{C}[a \to P]] \\ & \exists i.\forall j.\exists r, F'.\ t = a \circ r \wedge (r, F') \in C' \wedge \\ & \qquad \{(t_{i+j}, R)\} \subseteq \{(\langle\rangle, R) \mid R \subseteq A\} \cup \{(a \circ u, R) \mid (u, R) \in F'\} \\ \Longrightarrow & [\text{Teilmengenabschluß von } C'] \\ & \exists i.\forall j.\exists r, F'.\ t = a \circ r \wedge (r, F') \in C' \wedge \\ & \qquad \{(t_{i+j}, R)\} = \{(a \circ u, R) \mid (u, R) \in F'\} \\ \Longrightarrow & [\text{Prädikatenlogik}] \\ & \exists i.\forall j.\exists r, u.\ t = a \circ r \wedge t_{i+j} = a \circ u \wedge \\ & \qquad (r, \{(u, R)\}) \in C' \\ \Longrightarrow & [\text{Prädikatenlogik}] \\ & \exists i, r, u.\forall j.\ t = a \circ r \wedge t_{i+j} = a \circ u_{i+j} \wedge \\ & \qquad (r, \{(u_{i+j}, R)\}) \in C' \\ \Longrightarrow & [\text{Induktion}] \\ & \exists i, r, u.\forall j.\ t = a \circ r \wedge t_{i+j} = a \circ u_{i+j} \wedge \\ & \qquad (r, \{(\mathsf{lub}_j\, u_{i+j}, R)\}) \in C' \\ \Longrightarrow & [\circ \text{ stetig im zweiten Argument}] \\ & \exists i, r, u.\ t = a \circ r \wedge \mathsf{lub}_j\, t_{i+j} = a \circ \mathsf{lub}_j\, u_{i+j} \wedge \\ & \qquad (r, \{(\mathsf{lub}_j\, u_{i+j}, R)\}) \in C' \\ \Longrightarrow & [\text{Definition } \mathcal{C}[a \to P]] \\ & \exists i.(t, \{(\operatorname{\mathsf{lub}}_j t_{i+j}, R)\}) \in C \\ \Longrightarrow & [\text{Definition } \mathsf{lub}_i] \\ & (t, \{(\operatorname{\mathsf{lub}}_i t_i, R)\}) \in C \end{array}$$

Nachweise für $P_1 \parallel P_2$

Im folgenden wird abkürzend $C_1 = \mathcal{C}[P_1]$, $C_2 = \mathcal{C}[P_2$ sowie $C = \mathcal{C}[P_1 \parallel P_2]$ verwendet.

Die Nichttrivialität von $P_1 \parallel P_2$ folgt aus

$$
\begin{array}{l}
\Longrightarrow \text{[Induktion]} \\
\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_1 \wedge (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_2 \\
\Longrightarrow \text{[Definition } ©\text{]} \\
\quad (A_1 © \langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_1 \wedge (A_2 © \langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_2 \\
\Longrightarrow \text{[Prädikatenlogik, Definition } ©\text{]} \\
\quad (A_1 © \langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_1 \wedge (A_2 © \langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_2 \wedge \\
\quad \{(\langle\rangle, \emptyset)\} \subseteq \{(t, R_1 \cup R_2) \mid \; t \sqsubseteq \langle\rangle \wedge \\
\qquad\qquad\qquad\qquad (A_1 © t, R_1) \in \{(\langle\rangle, \emptyset)\} \wedge (A_2 © t, R_2) \in \{(\langle\rangle, \emptyset)\}\} \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 © P_2]\text{]} \\
\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C
\end{array}
$$

Der Präfixabschluß von $P_1 \parallel P_2$ folgt aus

$$
\begin{array}{l}
\quad (r \circ s, F) \in C \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \parallel P_2]\text{]} \\
\quad \exists F_1, F_2. \\
\qquad (A_1 © (r \circ s), F_1) \in C_1 \wedge (A_2 © (r \circ s), F_2) \in C_2 \wedge \\
\qquad F \subseteq \{(t, R_1 \cup R_2) \mid t \sqsubseteq r \circ s \wedge (A_1 © t, R_1) \in F_1 \wedge (A_2 © t, R_2) \in F_2\} \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad \exists F_1, F_2. (A_1 © (r \circ s), F_1) \in C_1 \wedge (A_2 © (r \circ s), F_2) \in C_2 \\
\Longrightarrow \text{[Induktion]} \\
\quad \exists F_1, F_2. (A_1 © r, \{(A_1 © r, \emptyset)\} \in C_1 \wedge (A_2 © r, \{(A_2 © r, \emptyset)\} \in C_2 \\
\Longrightarrow \text{[Prädikatenlogik, Definition } \sqsubseteq\text{]} \\
\quad (A_1 © r, \{(A_1 © r, \emptyset)\} \in C_1 \wedge (A_2 © r, \{(A_2 © r, \emptyset)\} \in C_2 \wedge \\
\quad \{(r, \emptyset)\} \subseteq \{(t, R_1 \cup R_2) \mid \; t \sqsubseteq r \wedge \\
\qquad\qquad\qquad\qquad (A_1 © t, R_1) \in \{(A_1 © r, \emptyset)\} \wedge \\
\qquad\qquad\qquad\qquad (A_2 © t, R_2) \in \{(A_2 © r, \emptyset)\}\} \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \parallel P_2]\text{]} \\
\quad (r, \{(r, \emptyset)\}) \in C
\end{array}
$$

Die Spurbeschränkung von $P_1 \parallel P_2$ folgt aus

$$
\begin{array}{l}
\quad (t, \{r, R)\}) \in C \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \parallel P_2]\text{]} \\
\quad \exists F_1, F_2. \\
\qquad (A_1 © t, F_1) \in C_1 \wedge (A_2 © t, F_2) \in C_2 \wedge \\
\qquad \{(r, R)\} \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 © r, R_1) \in F_1 \wedge (A_2 © r, R_2) \in F_2\}
\end{array}
$$

$\Longrightarrow$ [Prädikatenlogik]
$\quad \exists R_1, R_2, F_1, F_2.\ (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$\qquad (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2 \wedge R = R_1 \cup R_2 \wedge$
$\qquad r \sqsubseteq t$
$\Longrightarrow$ [Prädikatenlogik]
$\quad r \sqsubseteq t$

Der Teilmengenabschluß von $\mathcal{C}[P_1 \parallel P_2]$ folgt aus

$\quad (t, F) \in C \wedge G \subseteq F$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 © P_2]$]
$\quad \exists F_1, F_2.\ G \subseteq F \wedge (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$\qquad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\quad \exists F_1, F_2.\ (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$\qquad G \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\quad (t, G) \in C$

Der Fairnessteilmengenabschluß von $P_1 \parallel P_2$ folgt mittels

$\quad (t, F) \in C \wedge (r, R) \in F \wedge S \subseteq R$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\quad \exists F_1, F_2.\ (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$\qquad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\} \wedge$
$\qquad (r, R) \in F \wedge S \subseteq R$
$\Longrightarrow$ [Prädikatenlogik]
$\quad \exists R_1, R_2, F_1, F_2.$
$\quad (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$\quad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\} \wedge$
$\quad r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2 \wedge R = R_1 \cup R_2 \wedge S \subseteq R$
$\Longrightarrow$ [Prädikatenlogik]
$\quad \exists R_1, R_2, F_1, F_2.$
$\quad (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$\quad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\} \wedge$
$\quad r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2 \wedge R = R_1 \cup R_2 \wedge$
$\quad (S \cap R_1) \subseteq R_1 \wedge (S \cap R_2) \subseteq R_2 \wedge (S \cap R_1) \cup (S \cap R_2) = S$
$\Longrightarrow$ [Induktion]
$\quad \exists R_1, R_2, F_1, F_2.$
$\quad (A_1 ©t, F_1) \in C_1 \wedge (A_1 ©r, S \cap R_1) \in F_1 \wedge$
$\quad (A_2 ©t, F_2) \in C_2 \wedge (A_2 ©r, S \cap R_2) \in F_2 \wedge$
$\quad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\} \wedge$
$\quad r \sqsubseteq t \wedge (S \cap R_1) \cup (S \cap R_2) = S$

$\Longrightarrow$ [Prädikatenlogik]
$\exists F_1, F_2.\ (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$F \cup \{(r, S)\} \subseteq \{(t, R_1 \cup R_2) \mid (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$(t, F \cup \{(r, S)\}) \in C$

Der Nachweis der Erweiterbarkeit erfolgt mittels Fallunterscheidung nach $b \in \alpha P_1$ und $b \in \alpha P_2$. Im folgenden wird nur der Fall $b \in \alpha P_1 \cap \alpha P_2$ betrachtet. Die Fälle $b \in \alpha P_1 \backslash \alpha P_2$ bzw. $b \in \alpha P_2 \backslash \alpha P_1$ ergeben sich durch Wegfallen der jeweiligen Alternative bei Anwendung der Induktionsvoraussetzung.

$(t, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\exists F_1, F_2.\ (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Induktion]
$\exists F_1, F_2.\ (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\} \wedge$
$((A_1 ©t, F_1 \cup \{(A_1 ©t, \{a\})\}) \in C_1) \vee (A_1 ©t \circ a, F_1) \in C_1) \wedge$
$((A_2 ©t, F_2 \cup \{(A_2 ©t, \{a\})\}) \in C_2) \vee (A_2 ©t \circ a, F_2) \in C_2)$
$\Longrightarrow$ [Prädikatenlogik]
$\exists F_1, F_2.$
$(A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\} \wedge$
$((A_1 ©t, F_1 \cup \{(A_1 ©t, \{a\})\}) \in C_1 \vee$
$(A_2 ©t, F_2 \cup \{(A_2 ©t, \{a\})\}) \in C_2 \vee$
$((A_1 ©t \circ a, F_1) \in C_1 \wedge (A_2 ©t \circ a, F_2) \in C_2))$
$\Longrightarrow$ [Präfixabschluß von C_1, C_2]
$\exists F_1, F_2.$
$(A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\} \wedge$
$(\ ((A_1 ©t, F_1 \cup \{(A_1 ©t, \{a\})\}) \in C_1 \wedge (A_2 ©t, F_2 \cup \{(A_2 ©t, \emptyset)\}) \in C_2) \vee$
$((A_1 ©t, F_1 \cup \{(A_1 ©t, \emptyset)\}) \in C_1 \wedge (A_2 ©t, F_2 \cup \{(A_2 ©t, \{a\})\}) \in C_2) \vee$
$((A_1 ©t \circ a, F_1) \in C_1 \wedge (A_2 ©t \circ a, F_2)))$
$\Longrightarrow$ [Prädikatenlogik]
$(\exists F_1, F_2.\ (A_1 ©t, F_1 \cup \{(A_1 ©t, \{a\})\}) \in C_1 \wedge (A_2 ©t, F_2 \cup \{(A_2 ©t, \emptyset)\}) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}) \vee$
$(\exists F_1, F_2.\ (A_1 ©t, F_1 \cup \{(A_1 ©t, \emptyset)\}) \in C_1 \wedge (A_2 ©t, F_2 \cup \{(A_2 ©t, \{a\})\}) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}) \vee$
$(\exists F_1, F_2.\ (A_1 ©t \circ a, F_1) \in C_1 \wedge (A_2 ©t \circ a, F_2) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\})$
$\Longrightarrow$ [Prädikatenlogik]

$$
\begin{array}{l}
\quad (\exists F_1, F_2.\\
\quad\ (A_1 ©t, F_1 \cup \{(A_1 ©t, \{a\})\}) \in C_1 \wedge (A_2 ©t, F_2 \cup \{(A_2 ©t, \emptyset)\}) \in C_2 \wedge\\
\quad\ F \cup \{(t, \{a\})\} \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\})\\
\quad (\exists F_1, F_2.\\
\quad\ (A_1 ©t, F_1 \cup \{(A_1 ©t, \emptyset)\}) \in C_1 \wedge (A_2 ©t, F_2 \cup \{(A_2 ©t, \{a\})\}) \in C_2 \wedge\\
\quad\ F \cup \{(t, \{a\})\} \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\})\\
\quad (\exists F_1, F_2.\\
\quad\ (A_1 ©t \circ a, F_1) \in C_1 \wedge (A_2 ©t \circ a, F_2) \in C_2 \wedge\\
\quad\ F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\})\\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \parallel P_2]]\\
\quad (t, R \cup \{(t, \{a\})\}) \in C \vee (t \circ a, F) \in C
\end{array}
$$

Der Fairnesspräfixabschluß von $P_1 \parallel P_2$ folgt für $r \notin A^*$ unmittelbar wegen $r \circ s = r$ aus dem Teilmengenabschluß von $P_1 \parallel P_2$. Für den Fall $r \in A^*$ folgt $A_1 ©r \in A_1^*$ und $A_2 ©r \in A_2^*$ und damit

$$
\begin{array}{l}
\quad (r \circ s, F) \in C\\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \parallel P_2]]\\
\quad \exists F_1, F_2.\ (A_1 ©(r \circ s), F_1) \in C_1 \wedge (A_2 ©(r \circ s), F_2) \in C_2 \wedge\\
\qquad\qquad F \subseteq \{(t, R_1 \cup R_2) \mid t \sqsubseteq r \circ s \wedge (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}\\
\Longrightarrow [\text{Definition } \circ, ©,\ A_1 ©r \in A_1^*, A_2 ©r \in A_2^*]\\
\quad \exists F_1, F_2.\ ((A_1 ©r) \circ (A_1 ©s), F_1) \in C_1 \wedge ((A_2 ©r) \circ (A_2 ©s), F_2) \in C_2 \wedge\\
\qquad\qquad F \subseteq \{(t, R_1 \cup R_2) \mid t \sqsubseteq r \circ s \wedge (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}\\
\Longrightarrow [\text{Induktion}]\\
\quad \exists F_1, F_2.\ ((A_1 ©r) \circ (A_1 ©s), F_1) \in C_1 \wedge ((A_2 ©r) \circ (A_2 ©s), F_2) \in C_2 \wedge\\
\qquad\qquad (A_1 ©r, \emptyset) \in F_1 \wedge (A_2 ©r, \emptyset) \in F_2 \wedge\\
\qquad\qquad F \subseteq \{(t, R_1 \cup R_2) \mid t \sqsubseteq r \circ s \wedge (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}\\
\Longrightarrow [\text{Definition } \sqsubseteq, \circ]\\
\quad \exists F_1, F_2.\ (A_1 ©(r \circ s), F_1) \in C_1 \wedge (A_2 ©(r \circ s), F_2) \in C_2 \wedge\\
\qquad\qquad r \sqsubseteq r \circ s \wedge (A_1 ©r, \emptyset) \in F_1 \wedge (A_2 ©r, \emptyset) \in F_2 \wedge\\
\qquad\qquad F \subseteq \{(t, R_1 \cup R_2) \mid t \sqsubseteq r \circ s \wedge (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}\\
\Longrightarrow [\text{Prädikatenlogik}]\\
\quad \exists F_1, F_2.\\
\quad (A_1 ©(r \circ s), F_1) \in C_1 \wedge (A_2 ©(r \circ s), F_2) \in C_2 \wedge\\
\quad\ F \cup \{(r, \emptyset)\} \subseteq \{(t, R_1 \cup R_2) \mid t \sqsubseteq r \circ s \wedge (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}\\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \parallel P_2]]\\
\quad (t, F \cup \{(r, \emptyset)\}) \in C
\end{array}
$$

Der Nachweis des Fairnessereignisabschlusses von $P_1 \parallel P_2$ erfolgt mittels Fallunterscheidung nach $a \in \alpha P_1$ und $a \in \alpha P_2$. Im folgenden wird nur der Fall $a \in \alpha P_1 \backslash \alpha P_2$ betrachtet. Die beiden anderen Fälle $a \in \alpha P_1 \cap \alpha P_2$ bzw. $a \in \alpha P_2 \backslash \alpha P_1$ ergeben sich analog.

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r \circ a, R) \in F
\end{array}
$$

$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\exists F_1, F_2.(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$(r \circ a, R) \in F \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists F_1, F_2, R_1, R_2.(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$r \circ a \sqsubseteq t \wedge (A_1©(r \circ a), R_1) \in F_2 \wedge (A_2©(r \circ a), R_2) \in F_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Teilmengenabschluß von P_1, P_2]
$\exists F_1, F_2.(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$r \sqsubseteq t \wedge (A_1©(r \circ a), \emptyset) \in F_1 \wedge (A_2©r, \emptyset) \in F_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Definition ©, $r \in A^*$, $a \in A_1 \backslash A_2$]
$\exists F_1, F_2.(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$r \sqsubseteq t \wedge ((A_1©r) \circ a, \emptyset) \in F_1 \wedge (A_2©r, \emptyset) \in F_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Induktion]
$\exists F_1, F_2.(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$r \sqsubseteq t \wedge (A_1©r, \{a\}) \in F_1 \wedge (A_2©r, \emptyset) \in F_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists F_1, F_2.(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$F \cup \{(r, \{a\})\} \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$(t, F \cup \{(r, R \cup \{a\})\}) \in C$

Der Fairnessrefusalabschluß von $P_1 \parallel P_2$ folgt unmittelbar für $r \notin A^*$ wegen $r \circ s = r$. Im Fall $r \in A^*$ folgt die Behauptung mittels

$(t, F) \in C \wedge (r \circ s, R) \in F$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\exists F_1, F_2.(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge (r \circ s, R) \in F \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists R_1, R_2, F_1, F_2.$
$(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$r \circ s \sqsubseteq t \wedge R = R_1 \cup R_2 \wedge (A_1©(r \circ s), R_1) \in F_1 \wedge (A_2©(r \circ s), R_2) \in F_2$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\circ$,©, $A_1©r \in A_1^*$,$A_2©r \in A_2^*$]
$\exists R_1, R_2, F_1, F_2.$
$(A_1©t, F_1) \in C_1 \wedge (A_2©t, F_2) \in C_2 \wedge$
$r \sqsubseteq t \wedge R = R_1 \cup R_2 \wedge ((A_1©r) \circ (A_1©s), R_1) \in F_1 \wedge ((A_2©r) \circ (A_2©s), R_2) \in F_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1©r, R_1) \in F_1 \wedge (A_2©r, R_2) \in F_2\}$

$\Longrightarrow$ [Induktion]
$\exists R_1, R_2, F_1, F_2.(A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$r \sqsubseteq t \wedge R = R_1 \cup R_2 \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists F_1, F_2.(A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$F \cup \{(r, R)\} \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$(t, F \cup \{(r, R)\}) \in C$

Der Fairnessvereinigungsabschluß von $P_1 \parallel P_2$ folgt aus

$(t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\exists F_1, F_2.\ (A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$(r, R_1) \in F \wedge (r, R_2) \in F \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists F_1, F_2, R_{11}, R_{12}, R_{21}, R_{22}.$
$(A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge r \sqsubseteq t \wedge$
$(A_1 ©r, R_{11}) \in F_1 \wedge (A_2 ©r, R_{12}) \in F_2 \wedge R_1 = R_{11} \cup R_{12} \wedge$
$(A_1 ©r, R_{21}) \in F_1 \wedge (A_2 ©r, R_{22}) \in F_2 \wedge R_2 = R_{21} \cup R_{22} \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Induktion]
$\exists F_1, F_2, R_{11}, R_{12}, R_{21}, R_{22}.$
$(A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge r \sqsubseteq t \wedge$
$(A_1 ©r, R_{11} \cup R_{21}) \in F_1 \wedge (A_1 ©r, R_{12} \cup R_{12}) \in F_2 \wedge$
$R_1 = R_{11} \cup R_{12} \wedge R_2 = R_{21} \cup R_{22} \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists F_1, F_2.$
$(A_1 ©t, F_1) \in C_1 \wedge (A_2 ©t, F_2) \in C_2 \wedge$
$F \cup \{(r, R_1 \cup R_2)\} \subseteq \{(t, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 ©t, R_1) \in F_1 \wedge (A_2 ©t, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$(t, F \cup \{(r, R_1 \cup R_2)\}) \in C$

Die Spurkettenvollständigkeit von $P_1 \parallel P_2$ folgt trivialerweise für endliche Ketten. Anderenfalls gilt $\forall i.t_i \in A^*$ und es folgt

$\forall i.(t_i, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\forall i.\exists F_1, F_2.(A_1 ©t_i, F_1) \in C_1 \wedge (A_2 ©t_i, F_2) \in C_2 \wedge$
$F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t_i \wedge (A_1 ©r, R_1) \in F_1 \wedge (A_2 ©r, R_2) \in F_2\}$

$\Longrightarrow$ [Prädikatenlogik]
$\quad \forall i.\exists F_1, F_2.(A_1 \copyright t_i, F_1) \in C_1 \wedge (A_2 \copyright t_i, F_2) \in C_2 \wedge$
$\qquad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t_0 \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [Teilmengenabschluß von C_1, C_2]
$\quad \forall i.\exists F_1, F_2.(A_1 \copyright t_i, F_1) \in C_1 \wedge (A_2 \copyright t_i, F_2) \in C_2 \wedge$
$\qquad F = \{(r, R_1 \cup R_2) \mid r \sqsubseteq t_0 \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [A_1 endlich, A_2 endlich, $A_1 \copyright t_i \in A_1^*$, $A_2 \copyright t_i \in A_2^*$, Hilfsätze B.5.2, B.5.3 B.5.5]
$\quad \exists s, F_1, F_2.\forall i.(A_1 \copyright s_i, F_1) \in C_1 \wedge (A_2 \copyright s_i, F_2) \in C_2 \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \wedge$
$\qquad F = \{(r, R_1 \cup R_2) \mid r \sqsubseteq t_0 \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [Induktion]
$\quad \exists s, F_1, F_2.(\mathsf{lub}_i(A_1 \copyright s_i), F_1) \in C_1 \wedge (\mathsf{lub}_i(A_2 \copyright s_i), F_2) \in C_2 \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \wedge$
$\qquad F = \{(r, R_1 \cup R_2) \mid r \sqsubseteq t_0 \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [$\copyright$ stetig]
$\quad \exists s, F_1, F_2.(A_1 \copyright (\mathsf{lub}_i\, s_i), F_1) \in C_1 \wedge (A_2 \copyright (\mathsf{lub}_i\, s_i), F_2) \in C_2 \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \wedge$
$\qquad F = \{(r, R_1 \cup R_2) \mid r \sqsubseteq t_0 \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\quad \exists F_1, F_2.(A_1 \copyright (\mathsf{lub}_i\, t_i), F_1) \in C_1 \wedge (A_2 \copyright (\mathsf{lub}_i\, t_i), F_2) \in C_2 \wedge$
$\qquad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t_0 \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\mathsf{lub}_i\, t_i$]
$\quad \exists F_1, F_2.(A_1 \copyright (\mathsf{lub}_i\, t_i), F_1) \in C_1 \wedge (A_2 \copyright (\mathsf{lub}_i\, t_i), F_2) \in C_2 \wedge$
$\qquad F \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq \mathsf{lub}_i\, t_i \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\quad (\underset{i}{\mathsf{lub}}\, t_i, F) \in C$

Der Kettenabschluß von $P_1 \parallel P_2$ folgt unmittelbar für endliche Ketten, also für Ketten mit $\exists i.t_i = \mathsf{lub}_i\, t_i$. Im anderen Fall gilt $\forall i.t_i \sqsubset \mathsf{lub}_i\, t_i$ und damit $t_i \in A^*$ und es folgt

$\quad \forall i.(t, \{(t_i, R)\}) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \parallel P_2]$]
$\quad \forall i.\exists F_1, F_2.(A_1 \copyright t, F_1) \in C_1 \wedge (A_2 \copyright t, F_2) \in C_2 \wedge$
$\qquad \{(t_i, R)\} \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [Teilmengenabschluß]
$\quad \forall i.\exists F_1, F_2.(A_1 \copyright t, F_1) \in C_1 \wedge (A_2 \copyright t, F_2) \in C_2 \wedge$
$\qquad \{(t_i, R)\} = \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\quad \forall i.\exists R_1, R_2.(A_1 \copyright t, \{(A_1 \copyright t_i, R_1)\}) \in C_1 \wedge (A_2 \copyright t, \{(A_2 \copyright t_i, R_2)\}) \in C_2 \wedge$
$\qquad R = R_1 \cup R_2$
$\Longrightarrow$ [A_1 endlich, A_2 endlich, $A_1 \copyright t_i \in A_1^*$, $A_2 \copyright t_i \in A_2^*$, Hilfsätze B.5.2, B.5.3 B.5.5]
$\quad \exists s, R_1, R_2.\forall i.(A_1 \copyright t, \{(A_1 \copyright s_i, R_1)\}) \in C_1 \wedge (A_2 \copyright t, \{(A_2 \copyright s_i, R_2)\}) \in C_2 \wedge$
$\qquad R = R_1 \cup R_2 \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i$
$\Longrightarrow$ [Induktion]

$$
\begin{array}{l}
\quad \exists s, R_1, R_2.\ (A_1 \copyright t, \{(\mathsf{lub}_i(A_1 \copyright s_i), R_1)\}) \in C_1 \wedge (A_2 \copyright t, \{(\mathsf{lub}_i(A_2 \copyright s_i), R_2)\}) \in C_2 \wedge \\
\qquad\qquad R = R_1 \cup R_2 \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \\
\Longrightarrow [\copyright \text{ stetig}] \\
\quad \exists s, R_1, R_2.(A_1 \copyright t, \{(A_1 \copyright (\mathsf{lub}_i\, s_i), R_1)\}) \in C_1 \wedge (A_2 \copyright t, \{(A_2 \copyright (\mathsf{lub}_i\, s_i), R_2)\}) \in C_2 \wedge \\
\qquad\qquad R = R_1 \cup R_2 \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists R_1, R_2.(A_1 \copyright t, \{(A_1 \copyright (\mathsf{lub}_i\, t_i), R_1)\}) \in C_1 \wedge (A_2 \copyright t, \{(A_2 \copyright (\mathsf{lub}_i\, t_i), R_2)\}) \in C_2 \wedge \\
\qquad\qquad R = R_1 \cup R_2 \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists F_1, F_2.(A_1 \copyright t, F_1) \in C_1 \wedge (A_2 \copyright t, F_2) \in C_2 \wedge \\
\qquad\qquad \{(\mathsf{lub}_i\, t_i, R)\} \subseteq \{(r, R_1 \cup R_2) \mid r \sqsubseteq t \wedge (A_1 \copyright r, R_1) \in F_1 \wedge (A_2 \copyright r, R_2) \in F_2\} \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \parallel P_2]] \\
\quad (t, \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \in C
\end{array}
$$

Nachweise für $P_1 \vartriangle P_2$

Im folgenden wird abkürzend $C_1 = \mathcal{C}[P]$, $C = \mathcal{C}[P_2]$ und $C = \mathcal{C}[P_1 \vartriangle P_2]$, sowie $A_1 = \alpha P$ verwendet.

Die Nichttrivialität von $P_1 \vartriangle P_2$ folgt aus

$$
\begin{array}{l}
\Longrightarrow [\text{Nichttrivialität von } P] \\
\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_1 \\
\Longrightarrow [\text{Prädikatenlogik, Definition } \mathcal{C}[P_1 \vartriangle P_2]] \\
\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C
\end{array}
$$

Die Präfixabgeschlossenheit von $P_1 \vartriangle P_2$ folgt aus

$$
\begin{array}{l}
\quad (r \circ s, F) \in C \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \vartriangle P_2]] \\
\quad (r \circ s, F) \in C_1 \vee \\
\quad (\exists u, v, F_1, F_2.u \in A_1^* \wedge v \neq \langle\rangle \wedge r \circ s = u \circ v \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2) \\
\Longrightarrow [\text{Definition } \circ] \\
\quad (r \circ s, F) \in C_1 \vee \\
\quad (\exists u_1, u_2, v, F_1, F_2.\ u_1 \circ u_2 \in A_1^* \wedge v \neq \langle\rangle \wedge r = u_1 \wedge s = u_2 \circ v \wedge \\
\qquad\qquad (u_1 \circ u_2, F_1) \in C_1 \wedge (v, F_2) \in C_2) \vee \\
\quad (\exists u, v_1, v_2, F_1, F_2.\ u \in A_1^* \wedge v_1 \neq \langle\rangle \wedge r = u \circ v_1 \wedge s = v_2 \wedge \\
\qquad\qquad (u, F_1) \in C_1 \wedge (v_1 \circ v_2, F_2) \in C_2) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad (r \circ s, F) \in C_1 \vee \\
\quad (\exists u_2, v, F_1.(r \circ u_2, F_1) \in C_1) \vee \\
\quad (\exists u, v_1, v_2, F_2.u \in A_1^* \wedge v_1 \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge r = u \circ v_1 \wedge (v_1 \circ v_2, F_2) \in C_2)
\end{array}
$$

$\Longrightarrow$ [Induktion]
$(r, \emptyset) \in C_1 \vee$
$(r, \emptyset) \in C_1 \vee$
$(\exists u, v_1, F_1.u \in A_1^* \wedge v_1 \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge r = u \circ v_1 \wedge (v_1, \emptyset) \in C_2)$
$\Longrightarrow$ [Prädikatenlogik]
$(r, \emptyset) \in C_1 \vee$
$(\exists u, v_1, F_1.u \in A_1^* \wedge v_1 \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge r = u \circ v_1 \wedge (v_1, \emptyset) \in C_2)$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]
$(r, \emptyset) \in \mathcal{C}[P_1 \vartriangle P_2] \vee$
$(r, \emptyset) \in \mathcal{C}[P_1 \vartriangle P_2]$
$\Longrightarrow$ [Prädikatenlogik]
$(r, \emptyset) \in \mathcal{C}[P_1 \vartriangle P_2]$

Die Spurbeschränkung von $P_1 \vartriangle P_2$ folgt aus

$(t, \{(r, R)\}) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]
$(t, \{(r, R)\}) \in C_1 \vee$
$(\exists u, v, F_1, F_2.\ u \in A_1^* \wedge v \neq \langle\rangle \wedge t = u \circ v \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge$
$\{(r, R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup$
$\{(u \circ v, R) \mid (v, R) \in F_2\})$
$\Longrightarrow$ [Prädikatenlogik]
$(t, \{(r, R)\}) \in C_1 \vee$
$(\exists s, u, v, R_1, R_2, F_1, F_2.$
$u \in A_1^* \wedge v \neq \langle\rangle \wedge t = u \circ v \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge$
$((r, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2 \wedge R = R_1 \cup R_2) \vee (r = u \circ s \wedge (s, R) \in F_2))$
$\Longrightarrow$ [Prädikatenlogik]
$(t, \{(r, R)\}) \in C_1 \vee$
$(\exists u, v, R_1, F_1.t = u \circ v \wedge (u, F_1) \in C_1 \wedge (r, R_1) \in F_1) \vee$
$(\exists s, u, v, F_2.t = u \circ v \wedge (v, F_2) \in C_2 \wedge r = u \circ s \wedge (s, R) \in F_2)$
$\Longrightarrow$ [Teilmengenabschluß von C_1,C_2]
$(t, \{(r, R)\}) \in C_1 \vee$
$(\exists u, v, R_1.t = u \circ v \wedge (u, \{(r, R_1)\}) \in C_1) \vee$
$(\exists s, u, v.t = u \circ v \wedge (v, \{(s, R)\}) \in C_2 \wedge r = u \circ s)$
$\Longrightarrow$ [Induktion]
$(r \sqsubseteq t) \vee$
$(\exists u, v.t = u \circ v \wedge r \sqsubseteq u) \vee$
$(\exists s, u, v.t = u \circ v \wedge s \sqsubseteq v \wedge r = u \circ s)$
$\Longrightarrow$ [Definition $\circ$,$\sqsubseteq$]
$(r \sqsubseteq t) \vee (r \sqsubseteq t) \vee (r \sqsubseteq t)$
$\Longrightarrow$ [Prädikatenlogik]
$r \sqsubseteq t$

Der Teilmengenabschluß von $P_1 \triangle P_2$ folgt mittels

$$
\begin{array}{l}
\quad (t,F) \in C \wedge G \subseteq F \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \triangle P_2]] \\
\quad ((t,F) \in C_1 \wedge G \subseteq F) \vee \\
\quad (\exists r,s,F_1,F_2.G \subseteq F \wedge r \in A_1^* \wedge s \neq \langle\rangle \wedge (r,F_1) \in C_1 \wedge (s,F_2) \in C_2 \wedge t = r \circ s \wedge \\
\qquad\qquad F \subseteq \{(u, R_1 \cup R_2) \mid (u,R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\quad \{(r \circ v, R) \mid (v,R) \in F_2\}) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad ((t,F) \in C_1 \wedge G \subseteq F) \vee \\
\quad (\exists r,s,F_1,F_2.r \in A_1^* \wedge s \neq \langle\rangle \wedge (r,F_1) \in C_1 \wedge (s,F_2) \in C_2 \wedge t = r \circ s \wedge \\
\qquad\qquad G \subseteq \{(u, R_1 \cup R_2) \mid (u,R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\quad \{(r \circ v, R) \mid (v,R) \in F_2\}) \\
\Longrightarrow [\text{Induktion}] \\
\quad (t,G) \in C_1 \vee \\
\quad (\exists r,s,F_1,F_2.r \in A_1^* \wedge s \neq \langle\rangle \wedge (r,F_1) \in C_1 \wedge (s,F_2) \in C_2 \wedge t = r \circ s \wedge \\
\qquad\qquad G \subseteq \{(u, R_1 \cup R_2) \mid (u,R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\quad \{(r \circ v, R) \mid (v,R) \in F_2\}) \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \triangle P_2]] \\
\quad (t,G) \in C
\end{array}
$$

Der Nachweis des Fairnessteilmengenabschlusses von $P_1 \triangle P_2$ erfolgt mittels Fallunterscheidung nach $(t,F) \in C_1 \wedge (w,R) \in F$. In diesem Fall folgt die Aussage unmittelbar mittels Induktion:

$$
\begin{array}{l}
\quad (t,F) \in C_1 \wedge (w,R) \in F \wedge S \subseteq R \\
\Longrightarrow [\text{Induktion}] \\
\quad (t, F \cup \{(w,S)\}) \in C_1 \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \triangle P_2]] \\
\quad (t, \cup F\{(w,S)\}) \in C_1
\end{array}
$$

Anderenfalls folgt

$$
\begin{array}{l}
\quad (t,F) \in C \wedge (w,R) \in F \wedge S \subseteq R \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \triangle P_2]] \\
\quad ((t,F) \in C_1 \wedge (w,R) \in F \wedge S \subseteq R) \vee \\
\quad (\exists r,s,F. \\
\qquad (w,R) \in F \wedge S \subseteq R \wedge r \in A_1^* \wedge s \neq \langle\rangle \wedge (r,F_1) \in C_1 \wedge (s,F_2) \in C_2 \wedge t = r \circ s \wedge \\
\qquad F \subseteq \{(u, R_1 \cup R_2) \mid (u,R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v,R) \in F_2\}) \\
\Longrightarrow [\neg((t,F) \in C_1 \wedge (w,R) \in F)] \\
\quad \exists r,s,F. \\
\qquad (w,R) \in F \wedge S \subseteq R \wedge r \in A_1^* \wedge s \neq \langle\rangle \wedge (r,F_1) \in C_1 \wedge (s,F_2) \in C_2 \wedge t = r \circ s \wedge \\
\qquad F \subseteq \{(u, R_1 \cup R_2) \mid (u,R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v,R) \in F_2\}
\end{array}
$$

$\Longrightarrow$ [Prädikatenlogik]
$\exists r, s, F.$
$((\exists R_1, R_2.(w, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2 \wedge R = R_1 \cup R_2) \vee$
$(\exists v.(v, R) \in F_2 \wedge w = r \circ v)) \wedge$
$S \subseteq R \wedge r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$(\exists r, s, R_1, R_2, S_1, S_2, F.$
$(w, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2 \wedge R = R_1 \cup R_2 \wedge S_1 \subseteq R_1 \wedge S_2 \subseteq R_2 \wedge S = S_1 \cup S_2 \wedge$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}) \vee$
$(\exists r, s, v, F.$
$(v, R) \in F_2 \wedge w = s \circ v \wedge r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge S \subseteq R \wedge$
$(s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\})$
$\Longrightarrow$ [Induktion]
$(\exists r, s, S_1, S_2, F.$
$S = S_1 \cup S_2 \wedge$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1 \cup \{(w, S_1)\})) \in C_1 \wedge (s, F_2 \cup \{(\langle\rangle, S_2)\}) \in C_2 \wedge t = r \circ s \wedge$
$F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}) \vee$
$(\exists r, s, v, F.$
$w = r \circ u \wedge r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2 \cup \{(u, S)\}) \in C_2 \wedge t = r \circ s \wedge$
$F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\})$
$\Longrightarrow$ [Prädikatenlogik]
$(\exists r, s, F.\ r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$F \cup \{(w, S)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup$
$\{(r \circ v, R) \mid (v, R) \in F_2\}) \vee$
$(\exists r, s, F.\ r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge \wedge t = r \circ s \wedge$
$F \cup \{(w, S)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup$
$\{(r \circ v, R) \mid (v, R) \in F_2\})$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \triangle P_2]$]
$(t, F \cup \{(w, S)\}) \in C$

Die Erweiterbarkeit von $P_1 \triangle P_2$ folgt mittels

$(t, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \triangle P_2]$]
$(t, F) \in C_1 \vee$
$(\exists r, s, F_1, F_2.$
$r \in A^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge t = r \circ s \wedge (s, F_2) \in C_2 \wedge$
$F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\})$
$\Longrightarrow$ [Induktion]

$$
\begin{array}{l}
\quad ((t \circ a, F) \in C_1 \vee (t, F \cup \{(t, \{a\})\}) \in C_1) \vee \\
\quad (\exists r, s, F_1, F_2. \\
\quad\ r \in A^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge t = r \circ s \wedge \\
\quad\ ((s \circ a, F_2) \in C_2 \vee (s, F_2 \cup \{(s, \{a\})\}) \in C_2) \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad ((t \circ a, F) \in C_1 \vee (t, F \cup \{(t, \{a\})\}) \in C_1) \vee \\
\quad (\exists r, s, F_1, F_2. \\
\quad\ r \in A^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge t = r \circ s \wedge (s \circ a, F_2) \in C_2 \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists r, s, F_1, F_2. \\
\quad\ r \in A^* \wedge (r, F_1) \in C_1 \wedge t = r \circ s \wedge (s, F_2) \in C_2 \wedge (s, \{a\}) \in F_2 \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad ((t \circ a, F) \in C_1 \vee (t, F \cup \{(t, \{a\})\}) \in C_1) \vee \\
\quad (\exists r, s, F_1, F_2. \\
\quad\ r \in A^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge t \circ a = r \circ s \circ a \wedge (s \circ a, F_2) \in C_2 \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists r, s, F_1, F_2. \\
\quad\ r \in A^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge t = r \circ s \wedge (s, F_2) \in C_2 \wedge (s, \{a\}) \in F_2 \wedge \\
\quad\ F \cup \{(t, \{a\})\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\qquad \{(r \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \vartriangle P_2]] \\
\quad ((t \circ a, F) \in C \vee (t, F \cup \{(t, \{a\})\}) \in C) \vee \\
\quad (t \circ a, F) \in C \vee (t, F \cup \{(t, \{a\})\}) \in C \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad (t \circ a, F) \in C \vee (t, F \cup \{(t, \{a\})\}) \in C
\end{array}
$$

Der Fairnessrefusalabschluß von $P_1 \vartriangle P_2$ folgt aus

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r \circ s, R) \in F \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \vartriangle P_2]] \\
\quad ((t, F) \in C_1 \wedge (r \circ s \in F)) \vee \\
\quad (\exists u, v, F_1, F_2.\ (r \circ s, R) \in F \wedge \\
\qquad\qquad\qquad u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\qquad\qquad\qquad F \subseteq\ \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\qquad\qquad \{(u \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Prädikatenlogik]}
\end{array}
$$

$$
\begin{array}{l}
\quad (t,F) \in C_1 \wedge (r \circ s, R) \in F) \vee \\
\quad (\exists u, v, F_1, F_2.\ ((\exists R_1, R_2 . R = R_1 \cup R_2 \wedge (r \circ s, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in C_2) \vee \\
\qquad\qquad\qquad (\exists w . (w, R) \in F_2 \wedge u \circ w = r \circ s)) \wedge \\
\qquad\qquad\qquad u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\qquad\qquad\qquad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\qquad\quad \{(u \circ v, R_2) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad ((t,F) \in C_1 \wedge (r \circ s, R) \in F) \vee \\
\quad (\exists u, v, R_1, R_2, F_1, F_2. \\
\quad\ R = R_1 \cup R_2 \wedge (r \circ s, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in C_2 \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists u, v, w, F_1, F_2. \\
\quad\ (w, R) \in F_2 \wedge u \circ w = r \circ s \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Definition } \circ\text{]} \\
\quad ((t,F) \in C_1 \wedge (r \circ s, R) \in F) \vee \\
\quad (\exists u, v, R_1, R_2, F_1, F_2. \\
\quad\ R = R_1 \cup R_2 \wedge (r \circ s, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in C_2 \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists u, v, w_1, w_2, F_1, F_2. \\
\quad\ (\langle\rangle \circ w_2, R) \in F_2 \wedge r \circ w_1 = u \wedge w_1 \circ w_2 = s \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists u, v, w_1, w_2, F_1, F_2. \\
\quad\ (w_1 \circ w_2, R) \in F_2 \wedge u \circ w_1 = r \wedge w_2 = s \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Induktion]}
\end{array}
$$

$$
\begin{array}{l}
\quad (t, F \cup \{(r,R)\}) \in C_1) \vee \\
\quad (\exists u, v, R_1, R_2, F_1, F_2. \\
\quad\ R = R_1 \cup R_2 \wedge (r, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in C_2 \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists u, v, w_1, F_1, F_2. \\
\quad\ (\langle\rangle, R) \in F_2 \wedge r \circ w_1 = u \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists u, v, w_1, F_1, F_2. \\
\quad\ (w_1, R) \in F_2 \wedge u \circ w_1 = r \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Fairnesspräfixabschluß]} \\
\quad (t, F \cup \{(r,R)\}) \in C_1) \vee \\
\quad (\exists u, v, R_1, R_2, F_1, F_2. \\
\quad\ R = R_1 \cup R_2 \wedge (r, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in C_2 \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists u, v, F_1, F_2. \\
\quad\ (\langle\rangle, R) \in F_2 \wedge (r, \emptyset) \in F_1 \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\quad (\exists u, v, w_1, F_1, F_2. \\
\quad\ (w_1, R) \in F_2 \wedge u \circ w_1 = r \wedge \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad (t, F \cup \{(r,R)\}) \in C_1) \vee \\
\quad (\exists u, v, F_1, F_2. \\
\quad\ u \in A_1^* \wedge v \neq \langle\rangle \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\ F \cup \{(r,R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad \{(u \circ v, R) \mid (v, R) \in F_2\}) \vee \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \vartriangle P_2]] \\
\quad (t, F \cup \{(r,R)\}) \in C
\end{array}
$$

Die Fairnessvereinigung von $P_1 \vartriangle P_2$ folgt mittels

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \vartriangle P_2]]
\end{array}
$$

$$
\begin{array}{l}
((t,F) \in C_1 \wedge (r,R_1) \in F \wedge (r,R_2) \in F) \vee \\
(\exists u, v, F_1, F_2. \\
\quad u \in A_1^* \wedge v \neq \langle\rangle \wedge (u,F_1) \in C_1 \wedge (v,F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u,R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(u \circ v, R) \mid (v,R) \in F_2\} \wedge \\
\quad (r,R_1) \in F \wedge (r,R_2) \in F)
\end{array}
$$

Für den weiteren Nachweis werden die Fälle der Disjunktion separat behandelt. Im Fall $(t,F) \in C_1 \wedge (r,R_1) \in F \wedge (r,R_2) \in F$ folgt die Aussage mittels

$$
\begin{array}{l}
\quad (t,F) \in C_1 \wedge (r,R_1) \in F \wedge (r,R_2) \in F \\
\Longrightarrow [\text{Induktion}] \\
\quad (t, F \cup \{(r, R_1 \cup R_2)\}) \in C_1 \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \vartriangle P_2]] \\
\quad (t, F \cup \{(r, R_1 \cup R_2)\}) \in C
\end{array}
$$

Der Nachweis für die Alternative wird mittels Fallunterscheidung geführt. Wegen $(t,F) \in C$, $t = u \circ v$ und $(r,R_1) \in F$ folgt aus der Spurbeschränkung $r \sqsubseteq u \circ v$ und damit - entsprechend der Definition von $\sqsubseteq$ - die Fälle $r \sqsubseteq u$ sowie $u \sqsubseteq r$. Für den Fall $r \sqsubseteq u$ gilt

$$
\begin{array}{l}
\quad \exists u, v, F_1. F_2. \\
\quad\quad (r,R_1) \in F \wedge (r,R_2) \in F \wedge \\
\quad\quad u \in A_1^* \wedge (u,F_1) \in C_1 \wedge (v,F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\quad F \subseteq \{(u', R_1 \cup R_2) \mid (u',R_1) \in F_1 \wedge (v',R_2) \in F_2\} \cup \\
\quad\quad\quad\quad \{(u \circ v', R) \mid (v',R) \in F_2\} \\
\Longrightarrow [\text{Prädikatenlogik, } r \sqsubseteq u] \\
\quad \exists u, v, F_1, F_2, R_{11}, R_{12}, R_{21}, R_{22}. \\
\quad\quad (r,R_{11}) \in F_1 \wedge (r,R_{21}) \in F_1 \wedge (\langle\rangle, R_{12}) \in F_2 \wedge (\langle\rangle, R_{22}) \in F_2 \wedge \\
\quad\quad R_1 = R_{11} \cup R_{12} \wedge R_2 = R_{21} \cup R_{22} \wedge \\
\quad\quad u \in A_1^* \wedge (u,F_1) \in C_1 \wedge (v,F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\quad F \subseteq \{(u', R_1 \cup R_2) \mid (u',R_1) \in F_1 \wedge (v',R_2) \in F_2\} \cup \\
\quad\quad\quad\quad \{(u \circ v', R) \mid (v',R) \in F_2\} \\
\Longrightarrow [\text{Induktion}] \\
\quad \exists u, v, F_1, F_2, R_{11}, R_{12}, R_{21}, R_{22}. \\
\quad\quad (r, R_{11} \cup R_{21}) \in F_1 \wedge (\langle\rangle, R_{12} \cup R_{22}) \in F_2 \wedge \\
\quad\quad R_1 = R_{11} \cup R_{12} \wedge R_2 = R_{21} \cup R_{22} \wedge \\
\quad\quad u \in A_1^* \wedge (u,F_1) \in C_1 \wedge (v,F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\quad F \subseteq \{(u', R_1 \cup R_2) \mid (u',R_1) \in F_1 \wedge (v',R_2) \in F_2\} \cup \\
\quad\quad\quad\quad \{(u \circ v', R) \mid (v',R) \in F_2\} \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists u, v, F_1, F_2. \\
\quad\quad u \in A_1^* \wedge (u,F_1) \in C_1 \wedge (v,F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad\quad F \cup \{(r, R_1 \cup R_2)\} \subseteq \{(u', R_1 \cup R_2) \mid (u',R_1) \in F_1 \wedge (v',R_2) \in F_2\} \cup \\
\quad\quad\quad\quad\quad\quad\quad\quad \{(u \circ v', R) \mid (v',R) \in F_2\}
\end{array}
$$

$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]
$(F \cup \{(r, R_1 \cup R_2)\}) \in C$

Für den Fall $u \sqsubseteq r$ gilt

$$
\begin{array}{l}
\quad \exists u, v, F_1.F_2. \\
\quad (r, R_1) \in F \wedge (r, R_2) \in F \wedge \\
\quad u \in A_1^* \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad F \subseteq \{(u', R_1 \cup R_2) \mid (u', R_1) \in F_1 \wedge (v', R_2) \in F_2\} \cup \\
\qquad \{(u \circ v', R) \mid (v', R) \in F_2\} \\
\Longrightarrow \text{[Prädikatenlogik, } u \sqsubseteq r\text{]} \\
\quad \exists u, v, v', F_1, F_2. \\
\quad r = u \circ v' \wedge (v', R_1) \in F_2 \wedge (v', R_2) \in F_2 \wedge \\
\quad u \in A_1^* \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad F \subseteq \{(u', R_1 \cup R_2) \mid (u', R_1) \in F_1 \wedge (v', R_2) \in F_2\} \cup \\
\qquad \{(u \circ v', R) \mid (v', R) \in F_2\} \\
\Longrightarrow \text{[Induktion]} \\
\quad \exists u, v, v', F_1, F_2. \\
\quad r = u \circ v' \wedge (v', R_1 \cup R_2) \in F_2 \wedge \\
\quad u \in A_1^* \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad F \subseteq \{(u', R_1 \cup R_2) \mid (u', R_1) \in F_1 \wedge (v', R_2) \in F_2\} \cup \\
\qquad \{(u \circ v', R) \mid (v', R) \in F_2\} \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad \exists u, v, v', F_1, F_2. \\
\quad u \in A_1^* \wedge (u, F_1) \in C_1 \wedge (v, F_2) \in C_2 \wedge t = u \circ v \wedge \\
\quad F \cup \{(r, R_1 \cup R_2)\} \subseteq \{(u', R_1 \cup R_2) \mid (u', R_1) \in F_1 \wedge (v', R_2) \in F_2\} \cup \\
\qquad \{(u \circ v', R) \mid (v', R) \in F_2\} \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \vartriangle P_2]\text{]} \\
\quad (t, F \cup \{(r, R_1 \cup R_2)\}) \in C
\end{array}
$$

Der Nachweis der Spurkettenvollständigkeit von $P_1 \vartriangle P_2$ erfolgt mittels Fallunterscheidung nach $\forall i.(t_i, F) \in C_1$.

$\forall i.(t_i, F) \in C_1$**:** Dann folgt

$$
\begin{array}{l}
\quad \forall i.(t_i, F) \in C \\
\Longrightarrow \text{[Definition } \mathcal{C}[P_1 \vartriangle P_2]\text{]} \\
\quad \forall i.(t_i, F) \in C_1 \vee \\
\qquad (\exists r, s, F_1, F_2 : r \in A^* \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t_i = r \circ s) \\
\Longrightarrow [\forall i.(t_i, F) \in C_1] \\
\quad \forall i.(t_i, F) \in C_1 \\
\Longrightarrow \text{[Induktion]} \\
\quad (\mathop{\mathsf{lub}}_i t_i, F) \in C_1
\end{array}
$$

$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]
$(\mathop{\mathsf{lub}}_i t_i, F) \in C$

$\exists i.(t_i, F) \notin C_1$**:** Dann folgt

$$\exists r, s, F_1, F_2.\ r \in A^* \wedge (r, F_1) \in C_1 \wedge \\ s \neq \langle\rangle \wedge (s, F_2) \in C_2 \wedge t_i = r \circ s$$

Wegen der Anforderung A.18 an die Unterbrechungsaktionen von $P_1 \vartriangle P_2$ folgt für u, v mit

$$\exists F_1, F_2.\ u \in A^* \wedge (u, F_1) \in C_1 \wedge \\ v \neq \langle\rangle \wedge (v, F_2) \in C_2 \wedge t_{i+j} = u \circ v$$

die Aussage

$$r = u \wedge s \sqsubseteq v \tag{B.3}$$

denn es gilt

$\Longrightarrow$ [$(t_i)_{i\in\mathbb{N}}$ Kette]
$t_i \sqsubseteq t_{i+j}$
$\Longrightarrow$ [Definition r, s, u, v]
$r \circ s \sqsubseteq u \circ v$
$\Longrightarrow$ [Definition $\sqsubseteq$]
$\exists w.r \circ s \circ w = u \circ v$
$\Longrightarrow$ [$s \neq \langle\rangle$, $v \neq \langle\rangle$]
$\exists a, b, s', u', w.r \circ a \circ s' \circ w = u \circ b \circ v' \wedge s = a \circ s' \wedge u = b \circ u'$
$\Longrightarrow$ [$\exists F.(s, F) \in C_2$, $\exists F.(v, F) \in C_2$, Anforderung A.18]
$\exists a, b, s', u', w.r \circ a \circ s' \circ w = u \circ b \circ v' \wedge s = a \circ s' \wedge u = b \circ u' \wedge$
$a \notin A_1 \wedge b \notin A_1$
$\Longrightarrow$ [$\exists F.(r, F) \in C_1$, $\exists F.(u, F) \in C_1$]
$\exists a, b, s', u', w.r \circ a \circ s' \circ w = u \circ b \circ v' \wedge s = a \circ s' \wedge u = b \circ u' \wedge$
$a \notin A_1 \wedge b \notin A_1 \wedge r \in A_1^* \wedge u \in A_1^*$
$\Longrightarrow$ [Definition $\circ$]
$\exists a, b, s', u', w.r = u \wedge a = b \wedge s' \circ w = \circ v' \wedge s = a \circ s' \wedge u = b \circ u'$
$\Longrightarrow$ [Definition $\sqsubseteq$]
$r = u \wedge s \sqsubseteq u$

Wegen der Anforderung A.18 und $u \neq \langle\rangle$ folgt weiterhin

$$(t_{i+j}, F) \notin C_1 \tag{B.4}$$

Damit gilt insgesamt

$\forall j.(t_j, F) \in C$
$\Longrightarrow$ [Prädikatenlogik]
$\forall j.(t_{i+j}, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]

$$
\begin{aligned}
&\forall j.(t_{i+j}, F) \in C_1 \vee \\
&\quad \exists u, v, F_1, F_2. u \in A^* \wedge (u, F_1) \in C_1 \wedge v \neq \langle\rangle \wedge (v, F_2) \in C_2 \wedge t_{i+j} = u \circ v \wedge \\
&\qquad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad\quad \{(u \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow\ & [\text{Aussage B.4}] \\
&\forall j. \exists u, v, F_1, F_2. \\
&\quad u \in A^* \wedge (u, F_1) \in C_1 \wedge v \neq \langle\rangle \wedge (v, F_2) \in C_2 \wedge t_{i+j} = u \circ v \wedge \\
&\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad \{(u \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow\ & [\text{Aussage B.3}] \\
&\forall j. \exists v, F_1, F_2. \\
&\quad (r, F_1) \in C_1 \wedge v \neq \langle\rangle \wedge (v, F_2) \in C_2 \wedge t_{i+j} = r \circ v \wedge \\
&\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad \{(r \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
&\exists v, F_1, F_2. \\
&\quad \forall j. (r, F_1(j)) \in C_1 \wedge v_j \neq \langle\rangle \wedge (v_j, F_2(j)) \in C_2 \wedge t_{i+j} = r \circ v_j \wedge \\
&\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad \{(r \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow\ & [A_1 \text{ endlich, } A_2 \text{ endlich, } t_i \in A^*, \text{ Spurbeschränkung, Hilfsätze B.5.2, B.5.3, B.5.5}] \\
&\exists v, w, F_1, F_2. \\
&\quad \forall j. (r, F_1) \in C_1 \wedge v_j \neq \langle\rangle \wedge \mathsf{lub}_j\, v_j = \mathsf{lub}_j\, w_j \wedge \\
&\quad (w_j, F_2) \in C_2 \wedge t_{i+j} = r \circ v_j \wedge \\
&\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad \{(r \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow\ & [v_j \text{ Kette, } \circ \text{ stetig im zweiten Argument, } = \text{zulässig}] \\
&\exists v, F_1, F_2. \\
&\quad \forall j. (r, F_1) \in C_1 \wedge \mathsf{lub}_j\, v_j \neq \langle\rangle \wedge \mathsf{lub}_j\, v_j = \mathsf{lub}_j\, w_j \wedge \\
&\quad (w_j, F_2) \in C_2 \wedge \mathsf{lub}_j\, t_{i+j} = r \circ \mathsf{lub}_j\, v_j \wedge \\
&\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad \{(r \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
&\exists w, F_1, F_2. \\
&\quad \forall j. (r, F_1) \in C_1 \wedge \mathsf{lub}_j\, w_j \neq \langle\rangle \wedge \\
&\quad (w_j, F_2) \in C_2 \wedge \mathsf{lub}_j\, t_{i+j} = r \circ \mathsf{lub}_j\, w_j \wedge \\
&\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad \{(r \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow\ & [\text{Induktion}] \\
&\exists w, F_1, F_2. \\
&\quad (r, F_1) \in C_1 \wedge \mathsf{lub}_j\, w_j \neq \langle\rangle \wedge \\
&\quad (\mathsf{lub}_j\, w_j, F_2) \in C_2 \wedge \mathsf{lub}_j\, t_{i+j} = r \circ \mathsf{lub}_j\, w_j \wedge \\
&\quad F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
&\qquad \{(r \circ v, R) \mid (v, R) \in F_2\}
\end{aligned}
$$

$\Longrightarrow$ [Prädikatenlogik]
$\exists w, F_1, F_2.$
$(r, F_1) \in C_1 \wedge w \neq \langle\rangle \wedge (w, F_2) \in C_2 \wedge \mathsf{lub}_j\, t_j = r \circ w \wedge$
$F \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \{(r \circ v, R) \mid (v, R) \in F_2\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$, $r \in A_1^*$]
$(\underset{j}{\mathsf{lub}}\, t_j, F) \in C$

Der Nachweis der Kettenvollständigkeit von $P_1 \vartriangle P_2$ erfolgt mittels Fallunterscheidung nach $\forall i.(t, \{(t_i, R)\}) \in C_1$:

$\forall i.(t, \{(t_i, R)\}) \in C_1$**:** Dann folgt

$\Longrightarrow$ $[\forall i.(t, \{(t_i, R)\}) \in C_1]$
$\forall i.(t, \{(t_i, R)\}) \in C_1$
$\Longrightarrow$ [Induktion]
$(t, \{(\underset{i}{\mathsf{lub}}\, t_i, R)\}) \in C_1$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]
$(t, \{(\underset{i}{\mathsf{lub}}\, t_i, R)\}) \in C$

$\exists i.(t, \{(t_i, R)\}) \notin C_1$**:** Damit folgt wegen des Teilmengen- und Fairnessrefusalabschlusses für dieses i auch

$$(t, \{(t_{i+j}, R)\}) \notin C_1 \tag{B.5}$$

denn

$(t, \{(t_i, R)\}) \notin C_1$
$\Longrightarrow$ [Teilmengenabschluß]
$(t, \{(t_i, R)\} \cup \{(t_{i+j}, R)\}) \notin C_1$
$\Longrightarrow$ [Fairnessrefusalabschluß]
$(t, \{(t_{i+j}, R)\}) \notin C_1$

Mit Aussage B.5 folgt damit für obiges i

$(t, \{(t_{i+j}, R)\}) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]
$((t, \{(t_{i+j}, R)\}) \in C_1) \vee$
$(\exists r, s, F_1, F_2.r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$\{(t_{i+j}, R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup$
$\{(r \circ v, R) \mid (v, R) \in F_2\})$
$\Longrightarrow$ [Aussage B.5]
$\exists r, s, F_1, F_2.r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$\{(t_{i+j}, R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup$
$\{(r \circ v, R) \mid (v, R) \in F_2\})$

Mit Aussage B.3 folgt für u, v mit

$\exists F_1, F_2.\ u \in A^* \wedge (u, F_1) \in C_1 \wedge$
$v \neq \langle\rangle \wedge (v, F_2) \in C_2 \wedge t = u \circ v$

weiterhin $r = u \wedge s \sqsubseteq v$ und damit

$$r = u \wedge s = v \tag{B.6}$$

Wegen der Spurbeschränkung gilt weiterhin $\forall j.t_j \sqsubseteq t$ und damit folgen die Fälle $\forall j.t_j \sqsubseteq r$ und $\exists j.r \sqsubset t_j$. Eine Fallunterscheidung nach diesen Fällen ergibt:

$\forall j.t_j \sqsubseteq r$: Dann folgt

$$\exists k.\, \mathop{\mathsf{lub}}_j t_j = t_{i+k} \tag{B.7}$$

wegen

$$\begin{array}{l}
\quad \forall j.t_j \sqsubseteq r \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \forall j.t_j \in \{t \mid t \sqsubseteq r\} \\
\Longrightarrow [r \in A^*, \text{Hilfsatz B.5.1, Hilfsatz B.5.4}] \\
\quad \exists k. \forall j.t_{j+k} = t_k \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists k. \forall j.t_{j+k} = t_{i+k} \\
\Longrightarrow [\text{Definition } \mathsf{lub}] \\
\quad \exists k.\, \mathop{\mathsf{lub}}_j t_j = t_{i+k}
\end{array}$$

und es folgt

$$\begin{array}{l}
\exists r, s, F_1, F_2. \\
\quad r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge \\
\quad \{(t_{i+j}, R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\qquad \{(r \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow [\text{Aussage B.7}] \\
\quad \exists r, s, F_1, F_2. \\
\quad r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge \\
\quad \{(\mathsf{lub}_j\, t_j, R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\qquad \{(r \circ v, R) \mid (v, R) \in F_2\}) \\
\Longrightarrow [\text{Definition } \mathcal{C}[P_1 \vartriangle P_2]] \\
\quad (t, \{(\mathop{\mathsf{lub}}_j t_j, R)\}) \in C
\end{array}$$

$\exists j.r \sqsubset t_j$: Dann folgt für dieses j mittels der Ketteneigenschaft von $(t_k)_{k \in \mathbb{N}}$ auch $r \sqsubset t_{i+j+k}$ und es gilt

$$\begin{array}{l}
\forall k. \exists r, s, F_1, F_2. \\
\quad r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge \\
\quad \{(t_{i+k}, R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\qquad \{(r \circ v, R) \mid (v, R) \in F_2\} \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \forall k. \exists r, s, F_1, F_2. \\
\quad r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge \\
\quad \{(t_{i+j+k}, R)\} \subseteq \{(u, R_1 \cup R_2) \mid (u, R_1) \in F_1 \wedge (\langle\rangle, R_2) \in F_2\} \cup \\
\qquad\qquad\qquad \{(r \circ v, R) \mid (v, R) \in F_2\}
\end{array}$$

$\Longrightarrow$ [Definition $\sqsubset$, Spurbeschränkung, $r \sqsubset t_{i+j+k}$]
$\forall k.\exists r, s, F_1, F_2.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, F_1) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$\{(t_{i+j+k}, R)\} \subseteq \{(r \circ v, R) \mid (v, R) \in F_2\}$
$\Longrightarrow$ [Teilmengenabschluß]
$\forall k.\exists r, s, F_2.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, \emptyset) \in C_1 \wedge (s, F_2) \in C_2 \wedge t = r \circ s \wedge$
$\{(t_{i+j+k}, R)\} = \{(r \circ v, R) \mid (v, R) \in F_2\}$
$\Longrightarrow$ [Prädikatenlogik]
$\forall k.\exists r, s, v.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, \emptyset) \in C_1 \wedge (s, \{(v, R)\}) \in C_2 \wedge t = r \circ s \wedge$
$t_{i+j+k} = r \circ v$
$\Longrightarrow$ [Prädikatenlogik, Aussage B.6]
$\exists r, s, v.\forall k.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, \emptyset) \in C_1 \wedge (s, \{(v_k, R)\}) \in C_2 \wedge t = r \circ s \wedge$
$t_{i+j+k} = r \circ v_k$
$\Longrightarrow$ [Induktion]
$\exists r, s, v.\forall k.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, \emptyset) \in C_1 \wedge (s, \{(\mathsf{lub}_k\, v_k, R)\}) \in C_2 \wedge t = r \circ s \wedge$
$t_{i+j+k} = r \circ v_k$
$\Longrightarrow$ [$\circ$ stetig im zweiten Argument, $=$ zulässig]
$\exists r, s, v.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, \emptyset) \in C_1 \wedge (s, \{(\mathsf{lub}_k\, v_k, R)\}) \in C_2 \wedge t = r \circ s \wedge$
$\mathsf{lub}_k\, t_{i+j+k} = r \circ \mathsf{lub}_k\, v_k$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r, s, v.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, \emptyset) \in C_1 \wedge (s, \{v, R)\}) \in C_2 \wedge t = r \circ s \wedge$
$\mathsf{lub}_k\, t_k = r \circ v$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r, s, v.$
$r \in A_1^* \wedge s \neq \langle\rangle \wedge (r, \emptyset) \in C_1 \wedge (s, \{v, R)\}) \in C_2 \wedge t = r \circ s \wedge$
$\{(\mathsf{lub}_k\, t_k, R)\} \subseteq \{(r \circ v, S) \mid (v, S) \in \{(v, R)\}\}$
$\Longrightarrow$ [Definition $\mathcal{C}[P_1 \vartriangle P_2]$]
$(t, \{(\underset{k}{\mathsf{lub}}, R)\}) \in C$

Nachweise für $f(P)$

Im folgenden wird abkürzend $C = \mathcal{C}[f(P)]$ und $C' = \mathcal{C}[P]$ verwendet.

Die Nichttrivialität von $f(P)$ folgt mittels

$\Longrightarrow$ [Induktion]
$(\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C'$

$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$(f^*(\langle\rangle), \{(f^*(\langle\rangle), \bar{f}(\emptyset))\}) \in C$
$\Longrightarrow$ [Definition f^*, $\bar{f}$]
$(\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C'$

Der Präfixabschluß von $f(P)$ folgt mittels

$(r \circ s, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$\exists t', F'.r \circ s = f^*(t') \wedge (t', F') \in C' \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F')\}$
$\Longrightarrow$ [f^* surjektiv]
$\exists r', s', t', F'.r \circ s = f^*(t') \wedge (t', F') \in C' \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F')\} \wedge$
$r = f^*(r') \wedge s = f^*(s')$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', s', t', F'.r \circ s = f^*(t') \wedge (t', F') \in C' \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F')\} \wedge$
$r = f^*(r') \wedge r \circ s = f^*(r') \circ f^*(s')$
$\Longrightarrow$ [Definition f^*]
$\exists r', s', t', F'.r \circ s = f^*(t') \wedge (t', F') \in C' \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F')\} \wedge$
$r = f^*(r') \wedge r \circ s = f^*(r' \circ s')$
$\Longrightarrow$ [f^* injektiv]
$\exists r', s', t', F'.r \circ s = f^*(t') \wedge (t', F') \in C' \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F')\} \wedge$
$r = f^*(r') \wedge t' = r' \circ s'$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', s', F'.(r' \circ s', F') \in C' \wedge r = f^*(r')$
$\Longrightarrow$ [Induktion]
$\exists r'.(r', \emptyset) \in C' \wedge r = f^*(r')$
$\Longrightarrow$ [Definition $\bar{f}(\emptyset)$]
$\exists r'.(r', \emptyset) \in C' \wedge r = f^*(r') \wedge \emptyset = \bar{f}(\emptyset)$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$(r, \emptyset) \in C$

Die Spurbeschränkung von $f(P)$ folgt mittels

$(t, \{(r, R)\}) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$\exists t', F'.(t', F') \in C' \wedge f^*(t') = t \wedge \{(r, R)\} \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Teilmengenabschluß]
$\exists t', F'.(t', F') \in C' \wedge f^*(t') = t \wedge \{(r, R)\} = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', t', R'.(t', \{(r', R')\}) \in C' \wedge f^*(t') = t \wedge r = f^*(r') \wedge R = \bar{f}(R')$
$\Longrightarrow$ [Induktion]

$$
\begin{array}{ll}
 & \exists r', t'.r' \sqsubseteq t' \wedge r = f^*(r') \wedge f^*(t') = t \\
\Longrightarrow & [f^* \text{ monoton}] \\
 & r \sqsubseteq t
\end{array}
$$

Der Teilmengenabschluß von $f(P)$ folgt mittels

$$
\begin{array}{ll}
 & (t, F) \in C \wedge G \subseteq F \\
\Longrightarrow & [\text{Definition } \mathcal{C}[f(P)]] \\
 & \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge G \subseteq F \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge G \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[f(P)]] \\
 & (t, G) \in C
\end{array}
$$

Der Fairnessteilmengenabschluß von $f(P)$ folgt mittels

$$
\begin{array}{ll}
 & (t, F) \in C \wedge (r, R) \in F \wedge S \subseteq R \\
\Longrightarrow & [\text{Definition } \mathcal{C}[f(P)]] \\
 & \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
 & \qquad (r, R) \in F \wedge S \subseteq R \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & \exists r', t', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
 & \qquad\qquad (r', R') \in F' \wedge r = f^*(r') \wedge R = \bar{f}(R') \wedge S \subseteq R \\
\Longrightarrow & [\bar{f} \text{ surjektiv}] \\
 & \exists r', t', R', S', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
 & \qquad\qquad (r', R') \in F' \wedge r = f^*(r') \wedge R = \bar{f}(R') \wedge S = \bar{f}(S') \wedge S \subseteq R \\
\Longrightarrow & [\text{Definition } \bar{f}] \\
 & \exists r', t', R', S', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
 & \qquad\qquad (r', R') \in F' \wedge r = f^*(r') \wedge R = \bar{f}(R') \wedge S = \bar{f}(S') \wedge S' \subseteq R' \\
\Longrightarrow & [\text{Induktion}] \\
 & \exists r', t', S', F'.(t', F' \cup \{(r, S')\}) \in C' \wedge t = f^*(t') \wedge \\
 & \qquad\qquad F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge r = f^*(r') \wedge S = \bar{f}(S') \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \cup \{(r, S)\} \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \\
\Longrightarrow & [\text{Definition } \mathcal{C}[f(P)]] \\
 & (t, F \cup \{(r, S)\}) \in C
\end{array}
$$

Der Nachweis der Erweiterbarkeit von $f(P)$ folgt mittels

$$
\begin{array}{ll}
 & (t, F) \in C \\
\Longrightarrow & [\text{Definition } \mathcal{C}[f(P)]]
\end{array}
$$

$$
\begin{aligned}
& \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [f \text{ surjektiv}] \\
& \exists a', t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge f(a') = a \wedge F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [\text{Induktion}] \\
& \exists a', t', F'.((t', F' \cup \{(t', \{a'\})\}) \in C' \vee (t' \circ a', F') \in C') \wedge \\
& \qquad t = f^*(t') \wedge f(a') = a \wedge F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists a', t', F'.((t', F' \cup \{(t', \{a'\})\}) \in C' \wedge t = f^*(t') \wedge f(a') = a \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}) \vee \\
& \qquad ((t' \circ a', F') \in C' \wedge t = f^*(t') \wedge f(a') = a \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}) \\
\Longrightarrow\ & [\text{Definition } f^*, \text{ Definition } \bar{f}] \\
& \exists a', t', F'.((t', F' \cup \{(t', \{a'\})\}) \in C' \wedge t = f^*(t') \wedge \bar{f}(\{a'\}) = \{a\} \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}) \vee \\
& \qquad ((t' \circ a', F') \in C' \wedge t \circ a = f^*(t' \circ a') \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}) \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists a', t', F'.((t', F') \in C' \wedge t = f^*(t') \wedge \\
& \qquad F \cup \{(t, \{a\})\} \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}) \vee \\
& \qquad ((t', F') \in C' \wedge t \circ a = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}) \\
\Longrightarrow\ & [\text{Definition } \mathcal{C}[f(P)]] \\
& (t, F \cup \{(t, \{a\})\}) \in C \vee (t \circ a, F) \in C
\end{aligned}
$$

Der Nachweis des Fairnesspräfixabschlusses von $f(P)$ folgt mittels

$$
\begin{aligned}
& (r \circ s, F) \in C \\
\Longrightarrow\ & [\text{Definition } \mathcal{C}[f(P)]] \\
& \exists t', F'.(t', F') \in C' \wedge r \circ s = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [f^* \text{ surjektiv}] \\
& \exists r', s', t', F'.(t', F') \in C' \wedge r \circ s = f^*(t') \wedge r = f^*(r') \wedge s = f^*(s') \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists r', s', t', F'.(t', F') \in C' \wedge r \circ s = f^*(t') \wedge r = f^*(r') \wedge r \circ s = f^*(r') \circ f^*(s') \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [\text{Definition } f^*] \\
& \exists r', s', t', F'.(t', F') \in C' \wedge r \circ s = f^*(t') \wedge r = f^*(r') \wedge r \circ s = f^*(r' \circ s') \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists r', s', F'.(r' \circ s', F') \in C' \wedge r = f^*(r') \wedge r \circ s = f^*(r' \circ s') \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longrightarrow\ & [\text{Induktion}] \\
& \exists r', s', F'.(r' \circ s', F' \cup \{(r', \emptyset)\}) \in C' \wedge r = f^*(r') \wedge r \circ s = f^*(r' \circ s') \wedge \\
& \qquad F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}
\end{aligned}
$$

$\Longrightarrow$ [Definition $\bar{f}$]
$\exists r', s', F'.(r' \circ s', F' \cup \{(r', \bar{f}(\emptyset))\}) \in C' \wedge r = f^*(r') \wedge r \circ s = f^*(r' \circ s') \wedge$
$F \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', s', F'.(r' \circ s', F') \in C' \wedge r \circ s = f^*(r' \circ s') \wedge$
$F \cup \{(r, \emptyset)\} \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', t', F'.(t', F') \in C' \wedge r \circ s = f^*(t') \wedge$
$F \cup \{(r, \emptyset)\} \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$(r \circ s, F \cup \{(r, \emptyset)\}) \in C$

Der Fairnessereignisabschluß von $f(P)$ folgt mittels

$(t, F) \in C \wedge (r \circ a, R) \in F$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$\exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r \circ a, R) \in F$
$\Longrightarrow$ [Prädikatenlogik]
$\exists t', u', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$\wedge (u', R') \in F' \wedge r \circ a = f^*(u') \wedge R = \bar{f}(R')$
$\Longrightarrow$ [f^* surjektiv, f surjektiv]
$\exists a', r', t', u', R', F'.$
$(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(u', R') \in F' \wedge r \circ a = f^*(u') \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge a = f(a')$
$\Longrightarrow$ [Prädikatenlogik]
$\exists a', r', t', u', R', F'.$
$(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(u', R') \in F' \wedge r \circ a = f^*(u') \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge a = f(a) \wedge$
$r \circ a = f^*(r') \circ f(a')$
$\Longrightarrow$ [Definition f^*]
$\exists a', r', t', u', R', F'.$
$(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(u', R') \in F' \wedge r \circ a = f^*(u') \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge a = f(a) \wedge$
$r \circ a = f^*(r' \circ a')$
$\Longrightarrow$ [f^* injektiv]
$\exists a', r', t', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r' \circ a', R') \in F' \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge a = f(a)$
$\Longrightarrow$ [Induktion]
$\exists a', r', t', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r', \{a'\}) \in F' \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge a = f(a)$
$\Longrightarrow$ [Definition $\bar{f}$]

$$
\begin{array}{l}
\quad \exists a', r', t', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\qquad\qquad (r', \{a'\}) \in F' \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge \{a\} = \bar{f}(\{a\}) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \cup \{(r, \{a\})\} \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \\
\Longrightarrow [\text{Definition } \mathcal{C}[f(P)]] \\
\quad (t, F \cup \{(r, \{a\})\}) \in C
\end{array}
$$

Der Fairnessrefusalabschluß von $f(P)$ folgt mittels

$$
\begin{array}{l}
\quad (t, F) \in C \wedge (r \circ s, R) \in F \\
\Longrightarrow [\text{Definition } \mathcal{C}[f(P)]] \\
\quad \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\qquad (r \circ s, R) \in F \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists t', u', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\qquad\qquad (u', R') \in F' \wedge r \circ s = f^*(u') \wedge R = \bar{f}(R') \\
\Longrightarrow [f^* \text{ surjektiv}] \\
\quad \exists s', r', t', u', R', F'. \\
\quad (t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\quad (u', R') \in F' \wedge r \circ s = f^*(u') \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge s = f^*(s') \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists s', r', t', u', R', F'. \\
\quad (t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\quad (u', R') \in F' \wedge r \circ s = f^*(u') \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge r \circ s = f^*(r') \circ f^*(s') \\
\Longrightarrow [\text{Definition } f^*] \\
\quad \exists s', r', t', u', R', F'. \\
\quad (t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\quad (u', R') \in F' \wedge r \circ s = f^*(u') \wedge R = \bar{f}(R') \wedge r = f^*(r') \wedge r \circ s = f^*(r' \circ s') \\
\Longrightarrow [f^* \text{ injektiv}] \\
\quad \exists s', r', t', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\qquad\qquad (r' \circ s', R') \in F' \wedge R = \bar{f}(R') \wedge r = f^*(r') \\
\Longrightarrow [\text{Induktion}] \\
\quad \exists r', t', R', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge \\
\qquad\qquad (r', R') \in F' \wedge R = \bar{f}(R') \wedge r = f^*(r') \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \cup \{(r, R)\} \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \\
\Longrightarrow [\text{Definition } \mathcal{C}[f(P)]] \\
\quad (t, F \cup \{(r, R)\}) \in C
\end{array}
$$

Der Fairnessvereinigungsabschluß von $f(P)$ folgt mittels

$$
\quad (t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F
$$

$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$\exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r, R_1) \in F \wedge (r, R_2) \in F$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', t', R_1', R_2', F'.$
$(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r', R_1') \in F' \wedge (r', R_2') \in F' \wedge r = f^*(r') \wedge R_1 = \bar{f}(R_1') \wedge R_2 = \bar{f}(R_2')$
$\Longrightarrow$ [Induktion]
$\exists r', t', R_1', R_2', F'.$
$(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r', R_1' \cup R_2') \in F' \wedge r = f^*(r') \wedge R_1 = \bar{f}(R_1') \wedge R_2 = \bar{f}(R_2')$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', t', R_1', R_2', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r', R_1' \cup R_2') \in F' \wedge r = f^*(r') \wedge R_1 \cup R_2 = \bar{f}(R_1') \cup \bar{f}(R_2')$
$\Longrightarrow$ [Definition $\bar{f}$]
$\exists r', t', R_1', R_2', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$(r', R_1' \cup R_2') \in F' \wedge r = f^*(r') \wedge R_1 \cup R_2 = \bar{f}(R_1' \cup R_2')$
$\Longrightarrow$ [Prädikatenlogik]
$\exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge F \cup \{(r, R_1 \cup R_2)\} \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$(t, F \cup \{(r, R_1 \cup R_2)\}) \in C$

Der Spurkettenabschluß von $f(P)$ folgt mittels

$\forall i.(t_i, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$\forall i.\exists t', F'.(t', F') \in C' \wedge t_i = f^*(t') \wedge F \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Teilmengenabschluß]
$\forall i.\exists t', F'.(t', F') \in C' \wedge t_i = f^*(t') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [f^* injektiv, $\bar{f}$ injektiv]
$\exists F'.\forall i.\exists t'.(t', F') \in C' \wedge t_i = f^*(t') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists t', F'.\forall i.(t_i', F') \in C' \wedge t_i = f^*(t_i') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [$(t_i)_{i \in \mathbb{N}}$ Kette]
$\exists t', F'.\forall i.(t_i', F') \in C' \wedge t_i = f^*(t_i') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$t_i \sqsubseteq t_{i+1}$
$\Longrightarrow$ [Hilfsatz B.5.9]
$\exists t', F'.\forall i.(t_i', F') \in C' \wedge t_i = f^*(t_i') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\} \wedge$
$t_i' \sqsubseteq t_{i+1}'$
$\Longrightarrow$ [Induktion]
$\exists t', F'.\forall i.(\mathsf{lub}_i\, t_i', F') \in C' \wedge t_i = f^*(t_i') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [$=$ zulässig]
$\exists t', F'.(\mathsf{lub}_i\, t_i', F') \in C' \wedge \mathsf{lub}_i\, t_i = \mathsf{lub}_i\, f^*(t_i') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$

$\Longrightarrow$ [f^* stetig]
$\exists t', F'.(\mathsf{lub}_i\, t'_i, F') \in C' \wedge \mathsf{lub}_i\, t_i = f^*(\mathsf{lub}_i\, t'_i) \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Prädikatenlogik]
$\exists t', F'.(t', F') \in C' \wedge \mathsf{lub}_i\, t_i = f^*(t') \wedge F = \{(f^*(r), \bar{f}(R) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$(\underset{i}{\mathsf{lub}}\, t_i, F) \in C$

Der Kettenabschluß von $f(P)$ folgt mittels

$\forall i.(t, \{(t_i, R)\}) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$\forall i.\exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge \{(t_i, R)\} \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Teilmengenabschluß]
$\forall i.\exists t', F'.(t', F') \in C' \wedge t = f^*(t') \wedge \{(t_i, R)\} = \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\}$
$\Longrightarrow$ [Prädikatenlogik]
$\forall i.\exists r', t', R'.(t', \{(r', R')\}) \in C' \wedge t = f^*(t') \wedge t_i = f^*(r') \wedge R = \bar{f}(R')$
$\Longrightarrow$ [f^* injektiv, $\bar{f}$ injektiv]
$\exists t', R'.\forall i.\exists r'.(t', \{(r', R')\}) \in C' \wedge t = f^*(t') \wedge t_i = f^*(r') \wedge R = \bar{f}(R')$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', t', R'.\forall i.(t', \{(r'_i, R')\}) \in C' \wedge t = f^*(t') \wedge t_i = f^*(r'_i) \wedge R = \bar{f}(R')$
$\Longrightarrow$ [$(t_i)_{i\in\mathbb{N}}$ Kette]
$\exists r', t', R'.\forall i.(t', \{(r'_i, R')\}) \in C' \wedge t = f^*(t') \wedge t_i = f^*(r'_i) \wedge R = \bar{f}(R') \wedge$
$\qquad t_i \sqsubseteq t_{i+1}$
$\Longrightarrow$ [Hilfsatz B.5.9]
$\exists r', t', R'.\forall i.(t', \{(r'_i, R')\}) \in C' \wedge t = f^*(t') \wedge t_i = f^*(r'_i) \wedge R = \bar{f}(R') \wedge$
$\qquad r'_i \sqsubseteq r'_{i+1}$
$\Longrightarrow$ [Induktion]
$\exists r', t', R'.\forall i.(t', \{(\mathsf{lub}_i\, r'_i, R')\}) \in C' \wedge t = f^*(t') \wedge t_i = f^*(r'_i) \wedge R = \bar{f}(R')$
$\Longrightarrow$ [$=$ zulässig]
$\exists r', t', R'.(t', \{(\mathsf{lub}_i\, r'_i, R')\}) \in C' \wedge t = f^*(t') \wedge \mathsf{lub}_i\, t_i = \mathsf{lub}_i\, f^*(r'_i) \wedge R = \bar{f}(R')$
$\Longrightarrow$ [f^* stetig]
$\exists r', t', R'.(t', \{(\mathsf{lub}_i\, r'_i, R')\}) \in C' \wedge t = f^*(t') \wedge \mathsf{lub}_i\, t_i = f^*(\mathsf{lub}_i\, r'_i) \wedge R = \bar{f}(R')$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', t', R'.(t', \{(r', R')\}) \in C' \wedge t = f^*(t') \wedge \mathsf{lub}_i\, t_i = f^*(r') \wedge R = \bar{f}(R')$
$\Longrightarrow$ [Prädikatenlogik]
$\exists r', t', R'.(t', \{(r', R')\}) \in C' \wedge t = f^*(t') \wedge$
$\qquad \{(\mathsf{lub}_i\, t_i, R)\} \subseteq \{(f^*(r), \bar{f}(R) \mid (r, R) \in \{(r', R')\}\}$
$\Longrightarrow$ [Definition $\mathcal{C}[f(P)]$]
$(t, (\underset{i}{\mathsf{lub}}\, t_i, R)\}) \in C$

Nachweise für $\mu X.F(X)$

Im folgenden wird abkürzend $C = \mathcal{C}[\mu X.F(X)]$ und $C_i = \mathcal{C}[F^i(\mathsf{CHAOS}_A)]$ verwendet.

Die Nichttrivialität von $\mu X.F(X)$ folgt mittels

$\Longrightarrow$ [Induktion]
$\quad \forall i.(\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in C$

Der Präfixabschluß von $\mu X.F(X)$ folgt mittels

$\quad (r \circ s, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$\quad \forall i.(r \circ s, F) \in C_i$
$\Longrightarrow$ [Induktion]
$\quad \forall i.(r, \emptyset) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$\quad (r, \emptyset) \in C$

Die Spurbeschränkung von $\mu X.F(X)$ folgt mittels

$\quad (t, \{(r, R)\}) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$\quad \forall i.(t, \{(r, R)\}) \in C_i$
$\Longrightarrow$ [Induktion]
$\quad \forall i.r \sqsubseteq t$
$\Longrightarrow$ [Prädikatenlogik]
$\quad r \sqsubseteq t$

Der Teilmengenabschluß von $\mu X.F(X)$ folgt mittels

$\quad (t, F) \in C \wedge G \subseteq F$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$\quad (\forall i.(t, F) \in C_i) \wedge G \subseteq F$
$\Longrightarrow$ [Prädikatenlogik]
$\quad \forall i.((t, F) \in C_i \wedge G \subseteq F)$
$\Longrightarrow$ [Induktion]
$\quad \forall i.(t, G) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$\quad (t, G) \in C$

Der Fairnessteilmengenabschluß von $\mu X.F(X)$ folgt mittels

$\quad (t, F) \in C \wedge (r, R) \in F \wedge S \subseteq R$

$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(\forall i.(t,F) \in C_i) \wedge (r,R) \in F \wedge S \subseteq R$
$\Longrightarrow$ [Prädikatenlogik]
$\forall i.((t,F) \in C_i \wedge (r,R) \in F \wedge S \subseteq R)$
$\Longrightarrow$ [Induktion]
$\forall i.(t, F \cup \{(r,S)\}) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(t, F \cup \{(r,S)\}) \in C$

Die Erweiterbarkeit von $\mathcal{C}[\mu X.F(X)]$ folgt aus dem Nachweis der äquivalenten Aussage

$$(t,F) \in C \wedge (t \circ a, F) \notin C \Rightarrow (t, F \cup \{(t,\{a\})\}) \in C$$

Diese Aussage folgt mittels

$(t,F) \in C \wedge (t \circ a, F) \notin C$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(\forall i.(t,F) \in C_i) \wedge (\exists j.(t \circ a, F) \notin C_j)$
$\Longrightarrow$ [Monotonie]
$(\forall i.(t,F) \in C_i) \wedge (\exists j.\forall i.(t \circ a, F) \notin C_{j+i})$
$\Longrightarrow$ [Induktion]
$\forall i.((t, F \cup \{(t,\{a\})\}) \in C_i \vee (t \circ a, F) \in C_i) \wedge (\exists j.\forall i.(t \circ a, F) \notin C_{j+i})$
$\Longrightarrow$ [Prädikatenlogik]
$\exists j.\forall i.(t, F \cup \{(t,\{a\})\}) \in C_{j+i}$
$\Longrightarrow$ [Monotonie]
$\exists j.\forall i.(t, F \cup \{(t,\{a\})\}) \in C_i$
$\Longrightarrow$ [Prädikatenlogik]
$\forall i.(t, F \cup \{(t,\{a\})\}) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(t, F \cup \{(t,\{a\})\}) \in C$

Der Fairnesspräfixabschluß von $\mathcal{C}[\mu X.F(X)]$ folgt mittels

$(r \circ s, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$\forall i.(r \circ s, F) \in C_i$
$\Longrightarrow$ [Induktion]
$\forall i.(r \circ s, F \cup \{(r,\emptyset)\}) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(r \circ s, F \cup \{(r,\emptyset)\}) \in C$

Der Fairnessereignisabschluß von $\mathcal{C}[\mu X.F(X)]$ folgt mittels

$$(t,F) \in C \wedge (r \circ a, R) \in F$$

$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(\forall i.(t, F) \in C_i) \wedge (r \circ a, R) \in F$
$\Longrightarrow$ [Prädikatenlogik]
$\forall i.((t, F) \in C_i \wedge (r \circ a, R) \in F)$
$\Longrightarrow$ [Induktion]
$\forall i.(t, F \cup \{(r, \{a\})\}) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(t, F \cup \{(r, \{a\})\}) \in C$

Der Fairnessrefusalabschluß von $\mathcal{C}[\mu X.F(X)]$ folgt mittels

$(t, F) \in C \wedge (r \circ s, R) \in F$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(\forall i.(t, F) \in C_i) \wedge (r \circ s, R) \in F$
$\Longrightarrow$ [Prädikatenlogik]
$\forall i.((t, F) \in C_i \wedge (r \circ s, R) \in F)$
$\Longrightarrow$ [Induktion]
$\forall i.(t, F \cup \{(r, R)\}) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(t, F \cup \{(r, R)\}) \in C$

Der Fairnessvereinigungsabschluß von $\mathcal{C}[\mu X.F(X)]$ folgt mittels

$(t, F) \in C \wedge (r, R_1) \in F \wedge (r, R_2) \in F$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(\forall i.(t, F) \in C_i) \wedge (r, R_1) \in F \wedge (r, R_2) \in F$
$\Longrightarrow$ [Prädikatenlogik]
$\forall i.((t, F) \in C_i \wedge (r, R_1) \in F \wedge (r, R_2) \in F)$
$\Longrightarrow$ [Induktion]
$\forall i.(t, F \cup \{(r, R_1 \cup R_2)\}) \in C_i$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X.F(X)]$]
$(t, F \cup \{(r, R_1 \cup R_2)\}) \in C$

Der Spurkettenabschluß von $\mathcal{C}[\mu X.F(X)]$ folgt mittels

$\forall i.(t_i, F) \in C$
$\Longrightarrow$ [Definition $\mathcal{C}[\mu X : A.F(X)]$]
$\forall i.\forall j.(t_i, F) \in C_j$
$\Longrightarrow$ [Prädikatenlogik]
$\forall j.\forall i.(t_i, F) \in C_j$
$\Longrightarrow$ [Induktion]

$$
\begin{array}{l}
\quad \forall j.(\mathop{\mathsf{lub}}_i t_i, F) \in C_j \\
\Longrightarrow [\text{Definition } \mathcal{C}[\mu X : A.F(X)]] \\
\quad (\mathop{\mathsf{lub}}_i t_i, F) \in C
\end{array}
$$

Der Kettenabschluß von $\mathcal{C}[\mu X : A.F(X)]$ folgt mittels

$$
\begin{array}{l}
\quad \forall i.(t, \{(t_i, R)\}) \in C \\
\Longrightarrow [\text{Definition } \mathcal{C}[\mu X : A.F(X)]] \\
\quad \forall j.\forall i.(t, \{(t_i, R)\}) \in C_j \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \forall j.(t, \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \in C_j \\
\Longrightarrow [\text{Definition } \mathcal{C}[\mu X : A.F(X)]] \\
\quad (t, \{(\mathop{\mathsf{lub}}_i t_i, R)\}) \in C
\end{array}
$$

B.2.4 Abstraktion zur unendlichen Failuresemantik

Das unendliche Failuremodell stellt eine Abstraktion des komplexen Failuremodells dar. Aus intuitiver Sicht verzichtet das unendliche Failuremodell auf die explizite Beschreibung der Aktionen vor dem Ende des Ablaufs. Im unendlichen Failuremodell werden ausschließlich die Eigenschaften der Aktionen nach dem Ende des Ablaufs beschrieben, also das mögliche Blockieren nach diesem Ablauf. Formal ergibt sich ein Failurepaar (t, R) der unendlichen Failuresemantik aus einem Element (t, F) der komplexen Failuresemantik durch Einschränkung der Relation F auf t. Die unendliche Failuredarstellung eines Prozesses des komplexen Failuremodells ergibt sich damit zu

$$\{(t, R) \mid \exists F.(t, F) \in C \wedge (t, R) \in F\} \tag{B.8}$$

Der folgende Satz faßt diese Verträglichkeitsbeziehung zusammen.

Satz B.2.1 (Verträglichkeit) Für alle Prozeßterme P gebildet mittels

- CHAOS_A
- STOP_A
- $a \rightarrow P$
- $P_1 \sqcap P_2$
- $P_1 \parallel P_2$
- $f(P)$
- $P_1 \triangle P_2$
- $\mu X : A.F(X)$

gilt

$$(t, R) \in \mathcal{I}[P] \Leftrightarrow (t, \{(t, R\}) \in \mathcal{C}[P]$$

•

Beweis B.2.1 (Satz B.2.1) Der Beweis wird mittels Induktion über den Aufbau von P entsprechend den im Satz angegebenen Konstruktionsoperatoren geführt.

CHAOS_A**:** Die Behauptung folgt mittels

$$\begin{array}{ll} & (t, \{(t, R)\}) \in \mathcal{C}[\mathsf{CHAOS}_A] \\ \Longleftrightarrow & [\text{Definition } \mathcal{C}[\mathsf{CHAOS}_A]] \\ & t \in A^\omega \wedge R \subseteq A \\ \Longleftrightarrow & [\text{Definition } \mathcal{I}[\mathsf{CHAOS}_A]] \\ & (t, R) \in \mathsf{CHAOS}_A \end{array}$$

STOP_A**:** Die Behauptung folgt mittels

$$\begin{array}{ll} & (t, \{(t, R)\}) \in \mathcal{C}[\mathsf{STOP}_A] \\ \Longleftrightarrow & [\text{Definition } \mathcal{C}[\mathsf{STOP}_A]] \\ & t = \langle\rangle \wedge R \subseteq A \\ \Longleftrightarrow & [\text{Definition } \mathcal{I}[\mathsf{STOP}_A]] \\ & (t, R) \in \mathcal{I}[\mathsf{STOP}_A] \end{array}$$

$a \to P$**:** Die Behauptung folgt mittels

$$\begin{array}{ll} & (t, \{(t, R)\}) \in \mathcal{C}[a \to P] \\ \Longleftrightarrow & [\text{Definition } \mathcal{C}[a \to P]] \\ & (t = \langle\rangle \wedge R \subseteq A \backslash \{a\}) \vee \\ & (\exists s, F'.t = a \circ s \wedge (s, R) \in F' \wedge (s, F') \in \mathcal{C}[P]) \\ \Longleftrightarrow & [\text{Prädikatenlogik}] \\ & (t = \langle\rangle \wedge R \subseteq A \backslash \{a\}) \vee \\ & (\exists s.t = a \circ s \wedge (\exists F'.(s, F') \in \mathcal{C}[P] \wedge (s, R) \in F')) \\ \Longleftrightarrow & [\text{Teilmengenabschluß}] \\ & (t = \langle\rangle \wedge R \subseteq A \backslash \{a\}) \vee \\ & (\exists s.t = a \circ s \wedge (s, \{(s, R)\}) \in \mathcal{C}[P]) \\ \Longleftrightarrow & [\text{Induktion}] \\ & (t = \langle\rangle \wedge R \subseteq A \backslash \{a\}) \vee (\exists s.t = a \circ s \wedge (s, R) \in \mathcal{I}[P]) \\ \Longleftrightarrow & [\text{Definition } \mathcal{I}[a \to P]] \\ & (t, R) \in \mathcal{I}[a \to P] \end{array}$$

$P_1 \sqcap P_2$**:** Die Behauptung folgt mittels

$$\begin{array}{ll} & (t, \{(t, R)\}) \in \mathcal{C}[P_1 \sqcap P_2] \\ \Longleftrightarrow & [\text{Definition } \mathcal{C}[P_1 \sqcap P_2]] \\ & (t, \{(t, R)\}) \in \mathcal{C}[P_1] \vee (t, \{(t, R)\}) \in \mathcal{C}[P_2]) \\ \Longleftrightarrow & [\text{Induktion}] \\ & (t, R) \in \mathcal{I}[P_1] \vee (t, R) \in \mathcal{I}[P_2] \\ \Longleftrightarrow & [\text{Definition } \mathcal{I}[P_1 \sqcap P_2]] \\ & (t, R) \in \mathcal{I}[P_1 \sqcap P_2] \end{array}$$

$P_1 \parallel P_2$**:** Sei im folgenden $A_1 = \alpha P_1$ und $A_2 = \alpha P_2$. Die Behauptung folgt dann mittels

$$
\begin{array}{ll}
& (t, \{(t, R)\}) \in \mathcal{C}[P_1 \parallel P_2] \\
\Longleftrightarrow & [\text{Definition } \mathcal{C}[P_1 \parallel P_2]] \\
& \exists R_1, R_2.\ R = R_1 \cup R_2 \wedge \\
& \qquad \exists F_1, F_2.\ ((A_1 ©t, F_1) \in \mathcal{C}[P_1] \wedge (A_1 ©t, R_1) \in F_1 \wedge \\
& \qquad\qquad (A_2 ©t, F_2) \in \mathcal{C}[P_2] \wedge (A_2 ©t, R_2) \in F_2) \\
\Longleftrightarrow & [\text{Prädikatenlogik}] \\
& \exists R_1, R_2.\ R = R_1 \cup R_2 \wedge \\
& \qquad \exists F_1.((A_1 ©t, F_1) \in \mathcal{C}[P_1] \wedge (A_1 ©t, R_1) \in F_1) \wedge \\
& \qquad \exists F_2.((A_2 ©t, F_2) \in \mathcal{C}[P_2] \wedge (A_2 ©t, R_2) \in F_2) \\
\Longleftrightarrow & [\text{Teilmengenabschluß}] \\
& \exists R_1, R_2.\ R = R_1 \cup R_2 \wedge \\
& \qquad (A_1 ©t, \{(A_1 ©t, R_1)\}) \in \mathcal{C}[P_1] \wedge (A_2 ©t, \{(A_2 ©t, R_2)\}) \in \mathcal{C}[P_2] \\
\Longleftrightarrow & [\text{Induktion}] \\
& \exists R_1, R_2. R = R_1 \cup R_2 \wedge (A_1 ©t, R_1) \in \mathcal{I}[P_1] \wedge (A_2 ©t, R_2) \in \mathcal{I}[P_2] \\
\Longleftrightarrow & [\text{Definition } \mathcal{I}[P_1 \parallel P_2]] \\
& (t, R) \in \mathcal{I}[P_1 \parallel P_2]
\end{array}
$$

$f(P)$**:** Die Behauptung folgt mittels

$$
\begin{array}{ll}
& (t, \{(t, R)\}) \in \mathcal{C}[f(P)] \\
\Longleftrightarrow & [\text{Definition } \mathcal{C}[f(P)]] \\
& \exists t', F'.(t', F') \in \mathcal{C}[P] \wedge t = f^*(t') \wedge \{(t, R)\} \subseteq \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longleftrightarrow & [\text{Teilmengenabschluß}] \\
& \exists t', F'.(t', F') \in \mathcal{C}[P] \wedge t = f^*(t') \wedge \{(t, R)\} = \{(f^*(r), \bar{f}(R)) \mid (r, R) \in F'\} \\
\Longleftrightarrow & [\text{Prädikatenlogik}] \\
& \exists t', R'.(t', \{(t', R')\}) \in \mathcal{C}[P] \wedge t = f^*(t') \wedge R = \bar{f}(R') \\
\Longleftrightarrow & [\text{Induktion}] \\
& \exists t', R'.(t', R') \in \mathcal{I}[P] \wedge t = f^*(t') \wedge R = \bar{f}(R') \\
\Longleftrightarrow & [\text{Definition } \mathcal{I}[f(P)]] \\
& (t, R) \in \mathcal{I}[f(P)]
\end{array}
$$

$P_1 \triangle P_2$**:** Die Behauptung folgt mittels

$$
\begin{array}{ll}
& (t, \{(t, R)\}) \in \mathcal{C}[P_1 \triangle P_2] \\
\Longleftrightarrow & [\text{Definition } \mathcal{C}[P_1 \triangle P_2]] \\
& (t, \{(t, R)\}) \in \mathcal{C}[P_1] \vee \\
& (\exists r, s, F_1, F_2.\ s \in (\alpha P_1)^* \wedge (s, F_1) \in \mathcal{C}[P_1] \wedge \\
& \qquad\qquad r \neq \langle\rangle \wedge (r, F_2) \in \mathcal{C}[P_2] \wedge (r, R) \in F_2 \wedge t = s \circ r) \\
\Longleftrightarrow & [\text{“}\Longrightarrow\text{”: Präfixabschluß von } P_1, \text{ “}\Longleftarrow\text{”: } F_1 = \{(s, \emptyset)\}] \\
& (t, \{(t, R)\}) \in \mathcal{C}[P_1] \vee \\
& (\exists r, s, F_2.\ s \in (\alpha P_1)^* \wedge (s, \{(s, \emptyset)\}) \in \mathcal{C}[P_1] \wedge \\
& \qquad\qquad r \neq \langle\rangle \wedge (r, F_2) \in \mathcal{C}[P_2] \wedge (r, R) \in F_2 \wedge t = s \circ r) \\
\Longleftrightarrow & [\text{Teilmengenabschluß}]
\end{array}
$$

$$
\begin{array}{l}
\quad (t, \{(t,R)\}) \in \mathcal{C}[P_1] \vee \\
\quad (\exists r, s.\ s \in (\alpha P_1)^* \wedge (s, \{(s, \emptyset)\}) \in \mathcal{C}[P_1] \wedge \\
\qquad\qquad r \neq \langle\rangle \wedge (r, \{(r,R)\}) \in \mathcal{C}[P_2] \wedge t = s \circ r) \\
\Longleftrightarrow \text{[Induktion]} \\
\quad (t,R) \in \mathcal{I}[P_1] \vee \\
\quad (\exists r, s.\ s \in (\alpha P_1)^* \wedge (s, \emptyset) \in \mathcal{I}[P_1] \wedge \\
\qquad\qquad r \neq \langle\rangle \wedge (r,R) \in \mathcal{I}[P_2] \wedge t = s \circ r) \\
\Longleftrightarrow \text{[Definition } \mathcal{I}[P_1 \vartriangle P_2]] \\
\quad (t,R) \in \mathcal{I}[P_1 \vartriangle P_2]
\end{array}
$$

$\mu X : A.F(X)$**:** Die Behauptung folgt mittels

$$
\begin{array}{l}
\quad (t, \{(t,R)\}) \in \mathcal{C}[\mu X : A.F(X)]) \\
\Longleftrightarrow \text{[Definition } \mathcal{C}[\mu X : A.F(X)]] \\
\quad \forall i.(t, \{(t,R)\}) \in \mathcal{C}[F^i(\mathsf{CHAOS}_A)] \\
\Longleftrightarrow \text{[Induktion]} \\
\quad \forall i.(t,R) \in \mathcal{I}[F^i(\mathsf{CHAOS}_A)] \\
\Longleftrightarrow \text{[Definition } \mathcal{I}[\mu X.F(X)]] \\
\quad (t,R) \in \mathcal{I}[\mu X.F(X)]
\end{array}
$$

□

B.2.5 Komplexes und unendliches Failuremodell

In Abschnitt B.2.4 wurde gezeigt, daß das unendliche Failuremodell für die in Satz B.2.1 genannten Prozeßterme eine Abstraktion der komplexen Failuresemantik darstellt. Damit übertragen sich auch die Abschlußeigenschaften der Prozesse der komplexen Failuresemantik auf die unendliche Failuresemantik durch diesen Abstraktionsschritt in entsprechende einfachere Abschlußeigenschaften:

Satz B.2.2 (Abschlußeigenschaften von $\mathcal{I}$) Für die in Satz B.2.1 beschriebenen Prozeßterme P gilt

1. $(\langle\rangle, \emptyset) \in \mathcal{I}[P]$
2. $(t, R_1) \in \mathcal{I}[P] \wedge R_2 \subseteq R_1 \Rightarrow (t, R_2) \in \mathcal{I}[P]$
3. $(r \circ s, R) \in \mathcal{I}[P] \Rightarrow (r, \emptyset) \in \mathcal{I}[P]$
4. $(t,R) \in \mathcal{I}[P] \Rightarrow (t, R \cup \{a\}) \in \mathcal{I}[P] \vee (t \circ a, \emptyset) \in \mathcal{I}[P]$

•

Beweis B.2.2 (Satz B.2.2) Aussage 1 folgt mittels

$$
\begin{array}{l}
\Longrightarrow \text{[Nichttrivialität von } \mathcal{C}[P]] \\
\quad (\langle\rangle, \{(\langle\rangle, \emptyset)\}) \in \mathcal{C}[P] \\
\Longrightarrow \text{[Verträglichkeit]}
\end{array}
$$

$(\langle\rangle, \emptyset) \in \mathcal{I}[P]$

Aussage 2 folgt mittels

$(t, R_1) \in \mathcal{I}[P] \wedge R_2 \subseteq R_1$
$\Longrightarrow$ [Verträglichkeit]
$(t, \{(t, R_1)\}) \in \mathcal{C}[P] \wedge R_2 \subseteq R_1$
$\Longrightarrow$ [Fairnessteilmengenabschluß]
$(t, \{(t, R_2)\}) \in \mathcal{C}[P]$
$\Longrightarrow$ [Verträglichkeit]
$(t, R_2) \in \mathcal{I}[P]$

Aussage 3 folgt mittels

$(r \circ s, R) \in \mathcal{I}[P]$
$\Longrightarrow$ [Verträglichkeit]
$(r \circ s, \{(r \circ s, R)\}) \in \mathcal{C}[P]$
$\Longrightarrow$ [Präfixabschluß]
$(r, \emptyset) \in \mathcal{C}[P]$
$\Longrightarrow$ [Fairnesspräfixabschluß]
$(r, \{(r, \emptyset)\}) \in \mathcal{C}[P]$
$\Longrightarrow$ [Verträglichkeit]
$(r, \emptyset) \in \mathcal{I}[P]$

Aussage 4 folgt mittels

$(t, R) \in \mathcal{I}[P]$
$\Longrightarrow$ [Verträglichkeit]
$(t, \{(t, R)\}) \in \mathcal{C}[P]$
$\Longrightarrow$ [Erweiterbarkeit]
$(t, \{(t, R)\} \cup \{(t, \{a\})\}) \in \mathcal{C}[P] \vee (t \circ a, \{(t, R)\}) \in \mathcal{C}[P]$
$\Longrightarrow$ [Fairnessvereinigungsabschluß]
$(t, \{(t, R)\} \cup \{(t, \{a\})\} \cup \{(t, R \cup \{a\})\}) \in \mathcal{C}[P] \vee (t \circ a, \{(t, R)\}) \in \mathcal{C}[P]$
$\Longrightarrow$ [Teilmengenabschluß]
$(t, \{(t, R \cup \{a\})\}) \in \mathcal{C}[P] \vee (t \circ a, \emptyset) \in \mathcal{C}[P]$
$\Longrightarrow$ [Fairnesspräfixabschluß]
$(t, \{(t, R \cup \{a\})\}) \in \mathcal{C}[P] \vee (t \circ a, \{(t \circ a, \emptyset)\}) \in \mathcal{C}[P]$
$\Longrightarrow$ [Verträglichkeit]
$(t, R \cup \{a\}) \in \mathcal{I}[P] \vee (t \circ a, \emptyset) \in \mathcal{I}[P]$

□

B.3 Terminierbarkeit

Darüberhinaus erfüllen die oben beschriebenen Prozesse der komplexen unendlichen Failuresemantik eine weitere Eigenschaft, nämlich die Terminierbarkeit endlicher Abläufe. Formal wird die Aussage im folgenden Satz formuliert.

Satz B.3.1 (Terminierbarkeit) Für $r \in A^*$ und $B \subseteq A$ und die in Abschnitt B.2.2 definierten Prozeßterme P gilt:

$$(r, F) \in \mathcal{C}[P] \Rightarrow \exists s.s \in B^\omega \wedge (r \circ s, \{(r \circ s, B)\}) \in \mathcal{C}[P] \tag{B.9}$$

•

Prozesse haben also die Eigenschaft, daß sich endliche Abläufe r mittels Aktionen aus dem Teilalphabet B um einen Ablauf s ergänzen lassen und der Prozeß nach dem Gesamtablauf $r \circ s$ in B terminiert, also alle Aktionen aus B zurückweisen kann.

Beweis B.3.1 (Satz B.3.1) Im folgenden wird dabei die Tatsache ausgenutzt, daß das Alphabet A eines Prozesses endlich ist, also $A = \{a_0, \ldots, a_m\}$. Für $B \subseteq A$ wird die Mächtigkeit $\mid B \mid = n$ angenommen, und damit $B = \{a_0, \ldots, a_{n-1}\}$.

Die Aussage B.3.1 folgt aus den bisher nachgewiesenen Eigenschaften der Prozesse. Der Beweis erfolgt dabei konstruktiv. Dazu wird - basierend auf (r, F) - die folgende Kette $(s_i, F_i)_{i \in \mathbb{N}}$ konstruiert:

$$\begin{array}{lcl} s_0 & = & \langle\rangle \\ F_0 & = & \{(r, \emptyset)\} \\ s_{i+1} & = & \begin{cases} s_i \circ a_{i \bmod n} & \text{falls } (r \circ s_i \circ a_{i \bmod n}, F_i) \in C \\ s_i & \text{falls } (r \circ s_i \circ a_{i \bmod n}, F_i) \notin C \end{cases} \\ F_{i+1} & = & F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\} \cup \{(u, R \cup \{a_{i \bmod n}\}) \mid (u, R) \in F_i\} \end{array}$$

wobei B eine Teilmenge des endlichen Alphabet des Prozesses mit $B = \{a_0, \ldots, a_n\}$ ist.

Offensichtlich ist $(s_i)_{i \in \mathbb{N}}$ eine Kette hinsichtlich der Präfixrelation $\sqsubseteq$ als Ordnungsrelation, denn laut Definition gilt

$$\begin{array}{l} \quad s_i = s_{i+1} \vee \exists a.s_i \circ a = s_{i+1} \\ \Longrightarrow [\text{Definition } \sqsubseteq] \\ \quad s_i \sqsubseteq s_{i+1} \end{array}$$

Damit folgt mittels Induktion

$$\forall i, j \in \mathbb{N}.i \leq j \Rightarrow s_i \sqsubseteq s_j$$

Im folgenden wird daher mit $s = \mathsf{lub}_{i \in \mathbb{N}}\, s_i$ die wegen der Kettenvollständigkeit von A^ω existierende kleinste obere Schranke von $(s_i)_{i \in \mathbb{N}}$ bezeichnet.

Entsprechend bildet $(F_i)_{i \in \mathbb{N}}$ eine Kette hinsichtlich der Inklusionsordnung $\subseteq$ als Ordnungsrelation, denn laut Definition gilt

$$\begin{array}{l} \quad F_i \cup \{(s_i, \{a_{i \bmod n}\})\} = F_{i+1} \\ \Longrightarrow [X \subseteq X \cup Y] \\ \quad F_i \subseteq F_{i+1} \end{array}$$

und damit mittels Induktion

$$\forall i, j \in \mathbb{N}.i \leq j \Rightarrow F_i \subseteq F_j$$

Mittels Induktion läßt sich zeigen, daß

$$\forall i.(r \circ s_i, F_i) \in C$$

Die Induktionsfälle sind dabei

$i = 0$**:** Entsprechend der Prämisse der Aussage B.9 gilt $(r, F) \in C$ und damit folgt

$$\begin{array}{ll} & (r, F) \in C \\ \Longrightarrow & [\text{Teilmengenabschluß}] \\ & (r, \emptyset) \in C \\ \Longrightarrow & [\text{Präfixabschluß}] \\ & (r, \{(r, \emptyset)\}) \in C \\ \Longrightarrow & [\text{Definition } \circ] \\ & (r \circ \langle\rangle, \{(r, \emptyset)\}) \in C \\ \Longrightarrow & [\text{Definition } s_0, F_0] \\ & (s_0, F_0) \in C \end{array}$$

$i > 0$**:** Der Nachweis erfolgt mittels Fallunterscheidung nach $s_{i+1} = s_i$. Im Fall $s_{i+1} = s_i$ gilt

$$\begin{array}{ll} & s_{i+1} = s_i \\ \Longrightarrow & [\text{Induktionsvoraussetzung}] \\ & s_{i+1} = s_i \wedge (r \circ s_i, F_i) \in C \\ \Longrightarrow & [\text{Konstruktion } s_{i+1}] \\ & (r \circ s_i, F_i) \in C \wedge (r \circ s_i \circ a_{i \bmod n}, F_i) \notin C \\ \Longrightarrow & [\text{Erweiterbarkeit}] \\ & (r \circ s_i, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\}) \in C \\ \Longrightarrow & [r \circ s_i \in A^*, \text{ Fairnessrefusalabschluß}] \\ & (r \circ s_i, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\} \cup \{(u, \{a_{i \bmod n}\}) \mid u \sqsubseteq r \circ s_i\} \in C \\ \Longrightarrow & [\text{Spurbeschränkung, Fairnessvereinigungsabschluß}] \\ & (r \circ s_i, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\} \cup \{(u, \{a_{i \bmod n}\}) \mid u \sqsubseteq r \circ s_i\} \cup \\ & \qquad \{(u, R \cup \{a_{i \bmod n}\}) \mid (u, R) \in F_i\}) \in C \\ \Longrightarrow & [\text{Teilmengenabschluß}] \\ & (r \circ s_i, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\} \cup \{(u, R \cup \{a_{i \bmod n}\}) \mid (u, R) \in F_i\} \in C \\ \Longrightarrow & [\text{Definition } F_{i+1}] \\ & (r \circ s_i, F_{i+1}) \in C \\ \Longrightarrow & [s_{i+1} = s_i] \\ & (r \circ s_{i+1}, F_{i+1}) \in C \end{array}$$

Im Fall $s_{i+1} \neq s_i$ gilt $s_{i+1} = s_i \circ a_{i \bmod n}$ und es folgt

$$\begin{array}{ll} & s_{i+1} \neq s_i \\ \Longrightarrow & [\text{Konstruktion } s_{i+1}] \\ & (r \circ s_i \circ a_{i \bmod n}, F_i) \in C \\ \Longrightarrow & [\text{Fairnessereignisabschluß}] \\ & (r \circ s_i \circ a_{i \bmod n}, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\}) \in C \\ \Longrightarrow & [r \circ s_i \circ a_{i \bmod n} \in A^*, \text{ Fairnessrefusalabschluß}] \end{array}$$

$$
\begin{array}{l}
\quad (r \circ s_i, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\} \cup \{(u, \{a_{i \bmod n}\}) \mid u \sqsubseteq r \circ s_i\}) \in C \\
\Longrightarrow [(r \circ s_i, F_i) \in C, \text{ Spurbeschränkung, Fairnessvereinigungsabschluß}] \\
\quad (r \circ s_i, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\} \cup \{(u, \{a_{i \bmod n}\}) \mid u \sqsubseteq r \circ s_i\} \cup \\
\qquad \{(u, R \cup \{a_{i \bmod n}\}) \mid (u, R) \in F_i\}) \in C \\
\Longrightarrow [\text{Teilmengenabschluß}] \\
\quad (r \circ s_i, F_i \cup \{(r \circ s_i, \{a_{i \bmod n}\})\} \cup \{(u, R \cup \{a_{i \bmod n}\}) \mid (u, R) \in F_i\}) \in C \\
\Longrightarrow [\text{Definition } F_{i+1}] \\
\quad (r \circ s_i \circ a_{i \bmod n}, F_{i+1}) \in C \\
\Longrightarrow [s_{i+1} = s_i \circ a_{i \bmod n}] \\
\quad (r \circ s_{i+1}, F_{i+1}) \in C
\end{array}
$$

Insgesamt folgt damit $(r \circ s_{i+1}, F_{i+1}) \in C$

Wegen der Ketteneigenschaft von $(F_i)_{i \in \mathbb{N}}$ sowie der Teilmengenabgeschlossenheit von C folgt weiterhin

$$\forall i, j.i \leq j \Rightarrow (r \circ s_j, F_i) \in C \tag{B.10}$$

denn es gilt

$$
\begin{array}{l}
\quad i \leq j \\
\Longrightarrow [\forall i.(s_i, F_i) \in C] \\
\quad i \leq j \wedge (r \circ s_j, F_j) \in C \\
\Longrightarrow [(F_i)_{i \in \mathbb{N}} \text{ Kette}] \\
\quad F_i \subseteq F_j \wedge (r \circ s_j, F_j) \in C \\
\Longrightarrow [\text{Teilmengenabschluß von } C] \\
\quad (r \circ s_j, F_i) \in C
\end{array}
$$

Es bleibt, zu zeigen, daß der in B faire Ablauf s auch zur Termination in B führt.Dazu wird zunächst gezeigt, daß die Aktionen aus B jeweils bis zum Teilablauf s_i des Ablaufs s fair behandelt werden. Die folgende Hilfsbehauptung zeigt, daß in jedem Schritt vom Teilablauf s_i zum Teilablauf s_{i+1} eine weitere Aktion aus B fair behandelt wird:

$$(r \circ s_i, \{a_{k \bmod n} \mid i \leq k \leq i + j\}) \in F_{i+j+1} \tag{B.11}$$

Der Nachweis wird mittels Induktion über j geführt:

$j = 0$: Die Aussage folgt unmittelbar aus der Definition von F_i wegen

$$
\begin{array}{l}
\quad (r \circ s_i, \{a_{i \bmod n}\}) \in F_{i+1} \\
\Longrightarrow [j = 0] \\
\quad (r \circ s_i, \{a_{j \bmod n} \mid i \leq k \leq i + j\}) \in F_{i+j+1}
\end{array}
$$

$j > 0$: Die Aussage folgt mittels

$$
\begin{array}{l}
\quad (r \circ s_i, \{a_{k \bmod n} \mid i \leq k \leq i + j\}) \in F_{i+j+1} \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \{(r \circ s_i, \{a_{k \bmod n} \mid i \leq k \leq i + j\})\} \subseteq F_{i+j+1} \\
\Longrightarrow [\text{Definition } F_{i+j+1+1}] \\
\quad \{(r \circ s_i, \{a_{k \bmod n} \mid i \leq k \leq i + j\} \cup \{a_{i+j+1}\})\} \subseteq F_{i+j+1+1}
\end{array}
$$

$$
\begin{array}{l}
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad \{(r \circ s_i, \{a_{k \bmod n} \mid i \leq k \leq i+j+1\})\} \subseteq F_{i+j+1+1} \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad (r \circ s_i, \{a_{k \bmod n} \mid i \leq k \leq i+j+1\}) \in F_{i+j+1+1}
\end{array}
$$

Damit läßt sich nun zeigen, daß s einen hinsichtlich der Elemente aus B fairen Ablauf des Prozesses darstellt: zu jedem Teilablauf s_i existiert eine Fairnessmenge, nämlich F_{i+n}, die $(r \circ s_i, B)$ enthält:

$$\forall i.(r \circ s_i, B) \in F_{i+n} \tag{B.12}$$

Die Behauptung folgt sofort aus Aussage B.11 bei Wahl von $j = n$, da $\{a_{k \bmod n} \mid i \leq k \leq i+n-1\} = \{a_0, \ldots, a_{n-1}\} = B$. Wegen der Spurkettenvollständigkeit folgt aus Aussage B.10

$$(r \circ s, \{(r \circ s_i, B)\}) \in C \tag{B.13}$$

denn es gilt

$$
\begin{array}{l}
\quad \forall i, j.i \leq j \Rightarrow (r \circ s_j, F_i) \in C \\
\Longrightarrow \text{[Prädikatenlogik]} \\
\quad \forall i, j.(r \circ s_{i+j+n}, F_{i+n}) \in C \\
\Longrightarrow \text{[Aussage B.12]} \\
\quad \forall i, j.(r \circ s_{i+j+n}, F_{i+n}) \in C \wedge \{(r \circ s_i, B)\} \subseteq F_{i+n} \\
\Longrightarrow \text{[Teilmengenabschluß]} \\
\quad \forall i, j.(r \circ s_{i+j+n}, \{(r \circ s_i, B)\}) \in C \\
\Longrightarrow \text{[Spurkettenabschluß]} \\
\quad \forall i.(\mathop{\mathsf{lub}}_j (r \circ s_{i+j+n}), \{(r \circ s_i, B)\}) \in C \\
\Longrightarrow \text{[Definition lub]} \\
\quad \forall i.(\mathop{\mathsf{lub}}_j (r \circ s_j), \{(r \circ s_i, B)\}) \in C \\
\Longrightarrow \text{[Definition } s\text{]} \\
\quad \forall i.(r \circ s, \{(r \circ s_i, B)\}) \in C
\end{array}
$$

Entsprechend der Kettenvollständigkeit von C folgt aus Aussage B.13 damit

$$(r \circ s, \{(r \circ s, B)\}) \in C \tag{B.14}$$

denn es gilt

$$
\begin{array}{l}
\quad \forall i.(r \circ s, \{(r \circ s_i, B)\}) \in C \\
\Longrightarrow \text{[Kettenvollständigkeit]} \\
\quad (r \circ s, \{(\mathop{\mathsf{lub}}_i (r \circ s_i), B)\}) \in C \\
\Longrightarrow \text{[Definition } s\text{]} \\
\quad (r \circ s, \{(r \circ s, B)\}) \in C
\end{array}
$$

Aus Aussage B.14 folgt schließlich die Behauptung. □

Aus der Terminierbarkeit der Prozesse von $\mathcal{C}$ folgt auch eine entsprechende Aussage für die Prozesse von $\mathcal{I}$:

Satz B.3.2 (Terminierbarkeit) Für die in Satz B.2.1 beschriebenen Prozeßterme gilt

$$r \in A^* \wedge (r, \emptyset) \in \mathcal{I}[P] \wedge B \subseteq A \Rightarrow \exists s.s \in B^\omega \wedge (r \circ s, B) \in \mathcal{I}[P]$$

•

Beweis B.3.2 (Satz B.3.2) Die Aussage folgt aus Aussage B.9 mittels

$$\begin{array}{ll} & r \in A^* \wedge (r, \emptyset) \in \mathcal{I}[P] \wedge B \subseteq A \\ \Longrightarrow & \text{[Verträglichkeit,Teilmengenabschluß]} \\ & r \in A^* \wedge (r, \emptyset) \in \mathcal{C}[P] \wedge B \subseteq A \\ \Longrightarrow & \text{[Aussage B.9]} \\ & \exists s.s \in B^\omega \wedge (r \circ s, \{(r \circ s, B)\}) \in \mathcal{C}[P] \\ \Longrightarrow & \text{[Verträglichkeit]} \\ & \exists s.s \in B^\omega \wedge (r \circ s, B) \in \mathcal{I}[P] \end{array}$$

□

B.4 Eigenschaften der unendlichen Failuresemantik

In Abschnitt B.2.5 wurde gezeigt, daß die Prozeßeigenschaften der endlichen Failuresemantik sich auch für die in B.2.1 beschriebenen Prozeßterme in der unendlichen Failuresemantik nachweisen lassen. Zusätzlich wurde für diese Prozesse in Abschnitt B.3 mit der Terminierbarkeit eine weitere Eigenschaft nachgewiesen.

Um diese Eigenschaften auf alle in Definition A.3.3 beschriebenen Systeme übertragen zu können, wird unter Ausnutzung des Ergebnisses von Satz B.1.1 in Abschnitt B.4.1 deren Gültigkeit auch für die Abstraktion $P\backslash H$ gezeigt. Aufbauend auf den bisherigen Ergebnissen werden dann in Abschnitt B.4.2 auch die weiteren in Definition A.3.1 gezeigten Eigenschaften der Prozesse der unendlichen Failuresemantik nachgewiesen.

B.4.1 Eigenschaften von $P\backslash H$

Aufbauend auf der Terminierbarkeitsaussage für Prozesse lassen sich nun die Prozeßeigenschaften auch für den Abstraktionsoperator $P\backslash H$ nachweisen. Dazu wird gezeigt, daß sich die Aussagen

- der Nichttrivialität
- des Präfixabschlusses
- des Teilmengenabschlusses

- der Erweiterbarkeit
- der Terminierbarkeit

auch auf Prozeßterme der Form $P\backslash H$ übertragen lassen, wobei P ein Prozeßterm in der in Satz B.2.1 beschriebenen Form ist. Zusammen mit den in Abschnitt B.1 erarbeiteten Ergebnissen folgen damit diese Eigenschaften für alle in Definition A.3.3 eingeführten Systeme sequentieller Prozesse.

Nichttrivialität von $P\backslash H$

Im folgenden wird abkürzend $I = \mathcal{I}[P\backslash H]$ sowie $I' = \mathcal{I}[P]$ verwendet.

$\Longrightarrow$ [Nichttrivialität von P]
$(\langle\rangle, \emptyset) \in I'$
$\Longrightarrow$ [Terminierbarkeit von P]
$\exists s.s \in H^\omega \wedge (\langle\rangle \circ s, H) \in I'$
$\Longrightarrow$ [Definition $\mathcal{I}[P\backslash H]$]
$\exists s.s \in H^\omega \wedge ((A\backslash H)©(\langle\rangle \circ s), H\backslash H) \in I$
$\Longrightarrow$ [Definition ©, Prädikatenlogik]
$(\langle\rangle, \emptyset) \in I$

Präfixabschluß von $P\backslash H$

Im folgenden wird abkürzend $P = \mathcal{I}[P\backslash H]$ und $I' = \mathcal{I}[P]$ verwendet. Ohne Beschränkung der Allgemeinheit wird hier nur der Fall $r \in (A\backslash H)^*$ betrachtet. Anderenfalls folgt mit

$(r \circ s, R) \in I$
$\Longrightarrow$ [Definition $\circ$, $r \in (A\backslash H)^\infty$]
$(r, R) \in I$

die Behauptung direkt aus dem Teilmengenabschluß von $P\backslash H$. Mit obiger Annahme folgt

$(r \circ s, R) \in I$
$\Longrightarrow$ [Definition $\mathcal{I}[P\backslash H]$]
$\exists t.(A\backslash H)©t = r \circ s \wedge (t, R \cup H) \in I'$
$\Longrightarrow$ [Hilfsatz B.5.7, Definition ©, $r \in (A\backslash H)^*$]
$\exists r', s'.r' \in A^* \wedge (A\backslash H)©r' = r \wedge (A\backslash H)©s' = s \wedge (r' \circ s', R \cup H) \in I'$
$\Longrightarrow$ [Präfixabschluß von I']
$\exists r'.r' \in A^* \wedge (A\backslash H)©r' = r \wedge (r', \emptyset) \in I'$
$\Longrightarrow$ [Terminierbarkeit von P]
$\exists r', s'.r' \in A^* \wedge (A\backslash H)©r' = r \wedge s' \in H^\omega \wedge (r' \circ s', H) \in I'$
$\Longrightarrow$ [Definition $\mathcal{I}[P\backslash H]$]

$$
\begin{array}{ll}
 & \exists r', s'.r' \in A^* \wedge (A\backslash H)©r' = r \wedge s' \in H^\omega \wedge ((A\backslash H)©(r' \circ s'), H\backslash H) \in I \\
\Longrightarrow & \text{[Definition ©, } \circ\text{]} \\
 & \exists r', s'.(A\backslash H)©r' = r \wedge s' \in H^\omega \wedge (((A\backslash H)©r') \circ ((A\backslash H)©s'), \emptyset) \in I \\
\Longrightarrow & \text{[Definition ©, Prädikatenlogik]} \\
 & (r, \emptyset) \in I
\end{array}
$$

Teilmengenabschluß von $P\backslash H$

Der Teilmengenabschluß von $P\backslash H$ folgt direkt aus dem Teilmengenabschluß von P. Im folgenden wird abkürzend $I = \mathcal{I}[P\backslash H]$ und $I' = \mathcal{I}[P]$ verwendet.

$$
\begin{array}{ll}
 & (t, R) \in I \wedge S \subseteq R \\
\Longrightarrow & \text{[Definition } \mathcal{I}[P\backslash H]\text{]} \\
 & \exists t'.(A\backslash H)©t' = t \wedge (t', R \cup H) \in I' \wedge S \subseteq R \\
\Longrightarrow & \text{[Teilmengenabschluß von } I'\text{]} \\
 & \exists t'.(A\backslash H)©t' = t \wedge (t', S \cup H) \in I' \\
\Longrightarrow & \text{[Definition } \mathcal{I}[P\backslash H] \\
 & (t, S) \in I
\end{array}
$$

Erweiterbarkeit von $P\backslash H$

Für die Erweiterbarkeit von $P\backslash H$ reicht es, ohne Beschränkung der Allgemeinheit den Fall $t \in (A\backslash H)^*$ zu betrachten, wobei $A = \alpha P$. Anderenfalls gilt

$$
\begin{array}{ll}
 & (t, R) \in I \wedge t \in A^\infty \\
\Longrightarrow & \text{[Definition } \circ\text{]} \\
 & (t, R) \in I \wedge t \circ a = t \\
\Longrightarrow & \text{[Prädikatenlogik]} \\
 & (t \circ a, R) \in I \\
\Longrightarrow & \text{[Teilmengenabschluß von I, } \emptyset \subseteq R\text{]} \\
 & (t \circ a, \emptyset) \in I
\end{array}
$$

Im folgenden wird abkürzend $I = \mathcal{I}[P\backslash H]$ und $I' = \mathcal{I}[P]$ verwendet.

$$
\begin{array}{ll}
 & (t, R) \in I \wedge (t \circ a, \emptyset) \notin I \\
\Longrightarrow & \text{[Definition } \mathcal{I}[P\backslash H]\text{]} \\
 & (\exists t'.(A\backslash H)©t' = t \wedge (t', R \cup H) \in I') \wedge \\
 & (\forall t'.(A\backslash H)©t' \neq t \circ a \vee (t', H) \notin I') \\
\Longrightarrow & \text{[Hilfsatz B.5.7, Definition ©, } t \in (A\backslash H)^*\text{]} \\
 & (\exists r, s.(A\backslash H)©r = t \wedge r \in A^* \wedge s \in H^\omega \wedge (r \circ s, R \cup H) \in I') \wedge \\
 & (\forall r, s.(A\backslash H)©r \neq t \circ a \vee r \notin A^* \vee s \notin H^\omega \vee (r \circ s, H) \notin I') \\
\Longrightarrow & \text{[Terminierbarkeit]}
\end{array}
$$

$$
\begin{array}{l}
\quad (\exists r, s.(A\backslash H)\copyright r = t \wedge r \in A^* \wedge s \in H^\omega \wedge (r \circ s, R \cup H) \in I') \wedge \\
\quad (\forall r.(A\backslash H)\copyright r \neq t \circ a \vee r \notin A^* \vee (r, \emptyset) \notin I') \\
\Longrightarrow [\text{Definition } \copyright] \\
\quad \exists r, s.\ (A\backslash H)\copyright r = t \wedge r \in A^* \wedge s \in H^\omega \wedge (r \circ s, R \cup H) \in I' \wedge \\
\qquad (\forall s.s \notin H^* \vee (r \circ s \circ a, \emptyset) \notin I') \\
\Longrightarrow [\text{Verträglichkeit}] \\
\quad \exists r, s.\ (A\backslash H)\copyright (r \circ s) = t \wedge r \in A^* \wedge s \in H^\omega \wedge \\
\qquad (\exists F.(r \circ s, F) \in C' \wedge (r \circ s, R \cup H) \in F) \wedge \\
\qquad (\forall s, F.s \notin H^* \vee (r \circ s \circ a, F) \notin C' \vee (r \circ s \circ a, \emptyset) \notin F) \\
\Longrightarrow [\text{Teilmengenabschluß, Fairnesspräfixabschluß von } C'] \\
\quad \exists r, s.\ (A\backslash H)\copyright (r \circ s) = t \wedge r \in A^* \wedge s \in H^\omega \wedge \\
\qquad (r \circ s, \{(r \circ s, R \cup H)\}) \in C' \wedge \\
\qquad (\forall s.s \notin H^* \vee (r \circ s \circ a, \emptyset) \notin C') \\
\Longrightarrow [\text{Fairnessrefusalabschluß, Teilmengenabschluß}] \\
\quad \exists r, s.\ (A\backslash H)\copyright (r \circ s) = t \wedge r \in A^* \wedge s \in H^\omega \wedge \\
\qquad (\forall s'.s' \sqsubseteq s \wedge s' \in H^* \Rightarrow (r \circ s, \{(r \circ s', R \cup H)\}) \in C') \wedge \\
\qquad (\forall s.s \in H^* \Rightarrow (r \circ s \circ a, \emptyset) \notin C') \\
\Longrightarrow [\text{Erweiterbarkeit, Teilmengenabschluß}] \\
\quad \exists r, s.\ (A\backslash H)\copyright (r \circ s) = t \wedge r \in A^* \wedge s \in H^\omega \wedge \\
\qquad (\forall s'.s' \sqsubseteq s \wedge s' \in H^* \Rightarrow (r \circ s, \{(r \circ s', R \cup H \cup \{a\})\}) \in C') \\
\Longrightarrow [\text{Kettenvollständigkeit, Definition } \mathsf{lub}] \\
\quad \exists r, s.\ (A\backslash H)\copyright (r \circ s) = t \wedge r \in A^* \wedge s \in H^\omega \wedge \\
\qquad (r \circ s, \{(r \circ s, R \cup H \cup \{a\})\}) \in C' \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists t'.(A\backslash H)\copyright t' = t \wedge (t', \{(t', R \cup H \cup \{a\})\}) \in C' \\
\Longrightarrow [\text{Verträglichkeit}] \\
\quad \exists t'.(A\backslash H)\copyright t' = t \wedge (t', R \cup H \cup \{a\}) \in I' \\
\Longrightarrow [\text{Definition } \mathcal{I}[P\backslash H]] \\
\quad (t, R \cup \{a\}) \in I
\end{array}
$$

B.4.2 Weitere Eigenschaften

Für die Prozesse der unendlichen Failuresemantik gelten, wie in Definition A.3.1 gezeigt, neben der Terminierbarkeit zusätzliche Eigenschaften gegenüber den Prozessen der endlichen Failuresemantik. Diese Eigenschaften werden im folgenden für die Prozesse der in B.1.1 beschriebenen Prozeßterme gezeigt. Damit gelten diese Eigenschaften für alle Prozesse der in Abschnitt A.3.3 beschriebenen Prozeßterme.

Serialität

Für die Operatoren ohne die Abstraktion folgt die Aussage aus den Eigenschaften der komplexen Failuresemantik und der Verträglichkeitsaussage:

$$
\begin{array}{ll}
 & (t,R) \in \mathcal{I}[P] \wedge \{a\} \copyright t \notin A^* \\
\Longrightarrow & [\text{Verträglichkeit}] \\
 & (t, \{(t,R)\}) \in \mathcal{C}[P] \wedge \{a\} \copyright t \notin A^* \\
\Longrightarrow & [\text{Fairnessrefusalabschluß}] \\
 & (\forall s.s \sqsubseteq t \Rightarrow (t, \{(s,R),(t,R\}) \in \mathcal{C}[P]) \wedge \{a\} \copyright t \notin A^* \\
\Longrightarrow & [\text{Teilmengenabschluß}] \\
 & (\forall s.s \sqsubseteq t \Rightarrow (t, \{(s,R)\}) \in \mathcal{C}[P]) \wedge \{a\} \copyright t \notin A^* \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (\forall s \in A^*.s \sqsubseteq t \Rightarrow (t, \{(s,R)\}) \in \mathcal{C}[P]) \wedge \{a\} \copyright t \notin A^* \\
\Longrightarrow & [\text{Definition } \copyright] \\
 & (\forall s \in A^*.s \sqsubseteq t \Rightarrow (t, \{(s,R)\}) \in \mathcal{C}[P]) \wedge (\forall s \in A^*.s \sqsubseteq t \Rightarrow \exists r.s \circ r \circ a \sqsubseteq t) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & \forall s \in A^*.s \sqsubseteq t \Rightarrow \exists r.(t, \{(s,R),(s \circ r, \{a\})\}) \in \mathcal{C}[P] \\
\Longrightarrow & [\text{Fairnessereignisabschluß}] \\
 & \forall s \in A^*.s \sqsubseteq t \Rightarrow (t, \{(s,R),(s, \{a\})\}) \in \mathcal{C}[P] \\
\Longrightarrow & [\text{Fairnessvereinigungsabschluß}] \\
 & \forall s \in A^*.s \sqsubseteq t \Rightarrow (t, \{(s, R \cup \{a\})\}) \in \mathcal{C}[P] \\
\Longrightarrow & [\text{Kettenabschluß}] \\
 & (t, \{(\underset{s \in A^* \wedge s \sqsubseteq t}{\mathsf{lub}} s, R \cup \{a\})\}) \in \mathcal{C}[P] \\
\Longrightarrow & [\text{Definition } \mathsf{lub}] \\
 & (t, \{(t, R \cup \{a\})\}) \in \mathcal{C}[P] \\
\Longrightarrow & [\text{Verträglichkeit}] \\
 & (t, R \cup \{a\}) \in \mathcal{I}[P]
\end{array}
$$

Damit folgt auch die Aussage für die Abstraktion mittels

$$
\begin{array}{ll}
 & (t,R) \in \mathcal{I}[P \backslash H] \wedge \{a\} \copyright t \notin (A \backslash H)^* \\
\Longrightarrow & [\text{Definition } \mathcal{I}[P \backslash H]] \\
 & \exists t'.(A \backslash H) \copyright t' = t \wedge (t', R \cup H) \in \mathcal{I}[P] \wedge \{a\} \copyright t \notin (A \backslash H)^* \\
\Longrightarrow & [\text{Definition } \copyright] \\
 & \exists t'.(A \backslash H) \copyright t' = t \wedge (t', R \cup H) \in \mathcal{I}[P] \wedge \{a\} \copyright t \notin A^* \\
\Longrightarrow & [\text{Induktion}] \\
 & \exists t'.(A \backslash H) \copyright t' = t \wedge (t', R \cup H \cup \{a\}) \in \mathcal{I}[P] \\
\Longrightarrow & [\text{Definition } \mathcal{I}[P \backslash H], a \notin H] \\
 & (t, R \cup \{a\}) \in \mathcal{I}[P \backslash H]
\end{array}
$$

Die Aussage überträgt sich von P auch auf $f(P)$:

$$(t,R) \in \mathcal{I}[f(P)] \wedge a \copyright t \notin A^*$$

$\Longrightarrow$ [Definition $\mathcal{I}[f(P)]$]
$\quad \exists t', R'.(t', R') \in \mathcal{I}[P] \wedge f^*(t') = t \wedge \bar{f}(R') = R \wedge a \copyright t \notin A^*$
$\Longrightarrow$ [Definition f^*, $\bar{f}$]
$\quad \exists t', R'.(t', R') \in \mathcal{I}[P] \wedge f^*(t') = t \wedge \bar{f}(R') = R \wedge f(a) \copyright f^*(t') \notin \bar{f}(A)^*$
$\Longrightarrow$ [Induktion]
$\quad \exists t', R'.(t', R' \cup \{f(a)\}) \in \mathcal{I}[P] \wedge f^*(t') = t \wedge \bar{f}(R') = R$
$\Longrightarrow$ [Prädikatenlogik, Definition $\bar{f}$]
$\quad \exists t', R'.(t', R') \in \mathcal{I}[P] \wedge f^*(t') = t \wedge \bar{f}(R') = R \cup \{a\}$
$\Longrightarrow$ [Definition $\mathcal{I}[f(P)]$]
$\quad (t, R \cup \{a\}) \in \mathcal{I}[f(P)]$

Damit gilt das Serialitätsaxiom für die genannten Prozesse.

Allgemeine Erweiterbarkeit

Der Nachweis der Erweiterbarkeit kann direkt für die unendliche Failuresemantik mittels Induktion über den strukturellen Aufbau der Prozesse gezeigt, und zwar unter der Verwendung der Terminierungsaussage. Für die Einzelnachweisen werden, wie in den obigen Nachweisen, die Abkürzungen I für $\mathcal{I}[\mathsf{STOP}_A]$, $\mathcal{I}[a \to P]$, $\mathcal{I}[P_1 \sqcap P_2]$ usw. verwendet, sowie I_1 für $\mathcal{I}[P_1]$, I_2 für $\mathcal{I}[P_2]$, I' für $\mathcal{I}[P]$ sowie A für $\alpha(\mathsf{STOP}_A)$, $\alpha(a \to P)$, $\alpha(P_1 \sqcap P_2)$ usw. und A_1 für αP_1 und A_2 für αP_2.

Die Aussage folgt für $\mathcal{I}[\mathsf{CHAOS}_A]$ unmittelbar aus der Definition, da $(\mathsf{lub}_i\, t_i, R \cup \{a\}) \in A^\omega \times \mathbb{P}(A)$ und damit $(\mathsf{lub}_i\, t_i, R \cup \{a\}) \in \mathcal{I}[\mathsf{CHAOS}_A]$ gilt.

Die Aussage folgt für $\mathcal{I}[\mathsf{STOP}_A]$ mittels

$\quad (\underset{i}{\mathsf{lub}}\, t_i, R) \in I$
$\Longrightarrow$ [Definition $\mathcal{I}[\mathsf{STOP}_A]$]
$\quad \underset{i}{\mathsf{lub}}\, t_i = \langle\rangle \wedge R \subseteq A$
$\Longrightarrow$ [Prädikatenlogik]
$\quad \underset{i}{\mathsf{lub}}\, t_i = \langle\rangle \wedge R \cup \{a\} \subseteq A$
$\Longrightarrow$ [Definition $\mathcal{I}[\mathsf{STOP}_A]$]
$\quad (\underset{i}{\mathsf{lub}}\, t_i, R \cup \{a\}) \in I$

Die Aussage wird für $\mathcal{I}[b \to P]$ mittels Fallunterscheidung nach $\forall i.t_i = \langle\rangle$ geführt. Mit $\forall i.t_i = \langle\rangle$ folgt

$\quad (\underset{i}{\mathsf{lub}}\, t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I$
$\Longrightarrow$ [Definition lub_i, $\forall i.t_i = \langle\rangle$]
$\quad (\langle\rangle, R) \in I \wedge (\langle\rangle \circ a, \emptyset) \notin I$
$\Longrightarrow$ [Erweiterbarkeit]

$$
\begin{array}{ll}
 & ((\langle\rangle, R \cup \{a\}) \in I \\
\Longrightarrow & [\text{Definition } \mathsf{lub}_i,\ \forall i.t_i = \langle\rangle] \\
 & (\underset{i}{\mathsf{lub}}\, t_i, R \cup \{a\}) \in I
\end{array}
$$

Mit $\exists i.t_i \neq \langle\rangle$ folgt wegen der Ketteneigenschaft von $(t_i)_{i\in\mathbb{N}}$ auch $\exists i.\forall j.t_{i+j} \neq \langle\rangle$ und es folgt für dieses i

$$
\begin{array}{ll}
 & (\underset{i}{\mathsf{lub}}\, t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (\underset{j}{\mathsf{lub}}\, t_{i+j}, R) \in I \wedge \forall j.(t_{i+j} \circ a, \emptyset) \notin I \\
\Longrightarrow & [\text{Definition } \mathcal{I}[b \to P]] \\
 & (\exists s.\, \underset{j}{\mathsf{lub}}\, t_{i+j} = b \circ s \wedge (s, R) \in I') \wedge (\forall j, s.t_j \circ a \neq b \circ s \vee (s, \emptyset) \notin I') \\
\Longrightarrow & [\text{Definition } \circ,\ \forall j.t_{i+j} \neq \langle\rangle] \\
 & \exists s.(\forall j.t_{i+j} = b \circ s_j) \wedge (\underset{j}{\mathsf{lub}}\, s_j, R) \in I' \wedge (\forall j.(s_j \circ a, \emptyset) \notin I') \\
\Longrightarrow & [\text{Induktion}] \\
 & \exists s.\, \underset{j}{\mathsf{lub}}\, t_{i+j} = b \circ \underset{j}{\mathsf{lub}}\, s_j \wedge (\underset{j}{\mathsf{lub}}\, s_j, R \cup \{a\}) \in I' \\
\Longrightarrow & [\text{Definition } \mathcal{I}[b \to P]] \\
 & (\underset{j}{\mathsf{lub}}\, t_{i+j}, R \cup \{a\}) \in I \\
\Longrightarrow & [\text{Definition } \mathsf{lub}_j] \\
 & (\underset{j}{\mathsf{lub}}\, t_j, R \cup \{a\}) \in I
\end{array}
$$

Die Aussage folgt für $\mathcal{I}[P_1 \sqcap P_2]$ mittels

$$
\begin{array}{ll}
 & (\underset{i}{\mathsf{lub}}\, t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I \\
\Longrightarrow & [\text{Definition } \mathcal{I}[P_1 \sqcap P_2]] \\
 & ((\underset{i}{\mathsf{lub}}\, t_i, R) \in I_1 \vee (\underset{i}{\mathsf{lub}}\, t_i, R) \in I_2) \wedge \forall i.((t_i \circ a, \emptyset) \notin I_1 \wedge (t_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & ((\underset{i}{\mathsf{lub}}\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee ((\underset{i}{\mathsf{lub}}\, t_i, R) \in I_2 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_2) \\
\Longrightarrow & [\text{Induktion}] \\
 & (\underset{i}{\mathsf{lub}}\, t_i, R \cup \{a\}) \in I_1 \vee (\underset{i}{\mathsf{lub}}\, t_i, R \cup \{a\}) \in I_2 \\
\Longrightarrow & [\text{Definition } \mathcal{I}[P_1 \sqcap P_2]] \\
 & (\underset{i}{\mathsf{lub}}\, t_i, R \cup \{a\}) \in I
\end{array}
$$

Die Aussage wird für $\mathcal{I}[P_1 \parallel P_2]$ mittels Fallunterscheidung nach $a \in A_1$ und $a \in A_2$ gezeigt. Im folgenden wird nur der Fall $a \in A_1 \cap A_2$ betrachtet. Die beiden weiteren Fällen folgen entsprechend durch Weglassen jeweils einer Alternative zu $\forall i.(t_i \circ a, \emptyset) \notin I_1$ bzw. $\forall i.(t_i \circ a, \emptyset) \notin I_2$.

$$
(\underset{i}{\mathsf{lub}}\, t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I
$$

$\Longrightarrow$ [Definition $\mathcal{I}[P_1 \parallel P_2]$]
$(\exists R_1, R_2.(A_1 © \,\mathsf{lub}_i\, t_i, R_1) \in I_1 \wedge (A_2 © \,\mathsf{lub}_i\, t_i, R_2) \in I_2 \wedge R = R_1 \cup R_2) \wedge$
$(\forall i.(A_1 © (t_i \circ a), \emptyset) \notin I_1 \vee (A_2 © (t_i \circ a), \emptyset) \notin I_2)$
$\Longrightarrow$ [© stetig]
$(\exists R_1, R_2.(\mathsf{lub}_i(A_1 © t_i), R_1) \in I_1 \wedge (\mathsf{lub}_i(A_2 © t_i), R_2) \in I_2 \wedge R = R_1 \cup R_2) \wedge$
$(\forall i.(A_1 © (t_i \circ a), \emptyset) \notin I_1 \vee (A_2 © (t_i \circ a), \emptyset) \notin I_2)$

Damit folgt entsprechend Hilfsatz B.5.6 die Existenz einer Kette s mit

$$\mathop{\mathsf{lub}}_i s_i = \mathop{\mathsf{lub}}_i t_i \wedge ((\forall i.(s_i \circ a, \emptyset) \notin I_1) \vee (\forall i.(s_i \circ a, \emptyset) \notin I_2))$$

und es folgt schließlich

$\exists s, R_1, R_2.$
$(\mathsf{lub}_i(A_1 © s_i), R_1) \in I_1 \wedge (\mathsf{lub}_i(A_2 © s_i), R_2) \in I_2 \wedge R = R_1 \cup R_2) \wedge$
$(\forall i.((A_1 © s_i) \circ a, \emptyset) \notin I_1 \vee ((A_2 © s_i) \circ a, \emptyset) \notin I_2) \wedge$
$\mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i$
$\Longrightarrow$ [Prädikatenlogik]
$\exists s, R_1, R_2.\ ((\mathsf{lub}_i(A_1 © s_i), R_1) \in I_1 \wedge (\mathsf{lub}_i(A_2 © s_i), R_2) \in I_2 \wedge$
$(\forall i.((A_1 © s_i) \circ a, \emptyset) \notin I_1) \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \wedge R = R_1 \cup R_2) \vee$
$((\mathsf{lub}_i(A_1 © s_i), R_1) \in I_1 \wedge (\mathsf{lub}_i(A_2 © s_i), R_2) \in I_2 \wedge$
$(\forall i.((A_2 © s_i) \circ a, \emptyset) \notin I_1) \wedge \mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \wedge R = R_1 \cup R_2)$
$\Longrightarrow$ [Induktion]
$\exists s, R_1, R_2.$
$((\mathsf{lub}_i(A_1 © s_i), R_1) \cup \{a\}) \in I_1 \wedge (\mathsf{lub}_i(A_2 © s_i), R_2) \in I_2 \wedge$
$\mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \wedge R = R_1 \cup R_2) \vee$
$((\mathsf{lub}_i(A_1 © s_i), R_1) \in I_1 \wedge (\mathsf{lub}_i(A_2 © s_i), R_2 \cup \{a\}) \in I_2 \wedge$
$\mathsf{lub}_i\, s_i = \mathsf{lub}_i\, t_i \wedge R = R_1 \cup R_2)$
$\Longrightarrow$ [Prädikatenlogik]
$\exists R_1, R_2.$
$((\mathsf{lub}_i(A_1 © t_i), R_1) \cup \{a\}) \in I_1 \wedge (\mathsf{lub}_i(A_2 © t_i), R_2) \in I_2 \wedge$
$R = R_1 \cup R_2) \vee$
$((\mathsf{lub}_i(A_1 © t_i), R_1) \in I_1 \wedge (\mathsf{lub}_i(A_2 © t_i), R_2 \cup \{a\}) \in I_2 \wedge$
$R = R_1 \cup R_2)$
$\Longrightarrow$ [© stetig]
$\exists R_1, R_2.$
$((A_1 © \,\mathsf{lub}_i\, t_i, R_1) \cup \{a\}) \in I_1 \wedge (A_1 © \,\mathsf{lub}_i\, t_i, R_2) \in I_2 \wedge R = R_1 \cup R_2) \vee$
$((A_1 © \,\mathsf{lub}_i\, t_i, R_1) \in I_1 \wedge (A_1 © \,\mathsf{lub}_i\, t_i, R_2 \cup \{a\}) \in I_2 \wedge R = R_1 \cup R_2)$
$\Longrightarrow$ [Definition $\mathcal{I}[P_1 \parallel P_2]$]
$(\mathop{\mathsf{lub}}_i t_i, R \cup \{a\}) \in I \vee (\mathop{\mathsf{lub}}_i t_i, R \cup \{a\}) \in I$
$\Longrightarrow$ [Prädikatenlogik]
$(\mathop{\mathsf{lub}}_i t_i, R \cup \{a\}) \in I$

Die Aussage folgt für $\mathcal{I}[f(P)]$ mittels

$$
\begin{aligned}
& (\mathop{\mathsf{lub}}_i t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I \\
\Longrightarrow\ & [\text{Definition } \mathcal{I}[f(P)]] \\
& (\exists t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R) \wedge \\
& (\forall i, s'.f^*(s') = t_i \circ a \Rightarrow (s', \emptyset) \notin I') \\
\Longrightarrow\ & [f^* \text{ distributiv über } \circ] \\
& (\exists t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R) \wedge \\
& (\forall i, s', a'.(f^*(s') = t_i \wedge f(a') = a \Rightarrow (s' \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge \\
& \qquad (\forall i, s', a'.f^*(s') = t_i \wedge f(a') = a \Rightarrow (s' \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [f^* \text{ surjektiv}] \\
& \exists t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge \\
& \qquad (\forall i.\exists r'.f^*(r') = t_i) \wedge \\
& \qquad (\forall i, s', a'.f^*(s') = t_i \wedge f(a') = a \Rightarrow (s' \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [f \text{ surjektiv}] \\
& \exists a', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad (\forall i.\exists r'.f^*(r') = t_i) \wedge \\
& \qquad (\forall i, s'.f^*(s') = t_i \wedge \Rightarrow (s' \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists a', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad (\forall i.\exists r'.f^*(r') = t_i \wedge (r' \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists a', r', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad (\forall i.f^*(r'_i) = t_i \wedge (r'_i \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [(t_i)_{i \in \mathbb{N}} \text{ Kette}] \\
& \exists a', r', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad (\forall i.f^*(r'_i) = t_i \wedge (r'_i \circ a', \emptyset) \notin I') \wedge (\forall i.t_i \sqsubseteq t_{i+1}) \\
\Longrightarrow\ & [\text{Hilfsatz B.5.9}] \\
& \exists a', r', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad (\forall i.f^*(r'_i) = t_i \wedge (r'_i \circ a', \emptyset) \notin I') \wedge (\forall i.r'_i \sqsubseteq r'_{i+1}) \\
\Longrightarrow\ & [= \text{ zulässig}] \\
& \exists a', r', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad (\forall i.\, \mathsf{lub}_i\, f^*(r'_i) = \mathsf{lub}_i\, t_i \wedge (r'_i \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [f^* \text{ stetig}] \\
& \exists a', r', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad f^*(\mathsf{lub}_i\, r'_i) = \mathsf{lub}_i\, t_i \wedge (r'_i \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [f^* \text{ injektiv}] \\
& \exists a', r', t', R'.(t', R') \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
& \qquad \mathsf{lub}_i\, r'_i = t' \wedge (r'_i \circ a', \emptyset) \notin I') \\
\Longrightarrow\ & [\text{Prädikatenlogik}]
\end{aligned}
$$

$$
\begin{array}{l}
\quad \exists a', r', R'.(\mathsf{lub}_i\, r'_i, R') \in I' \wedge f^*(\mathsf{lub}_i\, r'_i) = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \wedge \\
\qquad\qquad (r'_i \circ a', \emptyset) \notin I') \\
\Longrightarrow [\text{Induktion}] \\
\quad \exists a', r', R'.(\mathsf{lub}_i\, r'_i, R' \cup \{a'\}) \in I' \wedge f^*(\mathsf{lub}_i\, r'_i) = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad \exists a', t', R'.(t', R' \cup \{a'\}) \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R') = R \wedge f(a') = a \\
\Longrightarrow [\text{Definition } \bar{f}] \\
\quad \exists a', t', R'.(t', R' \cup \{a'\}) \in I' \wedge f^*(t') = \mathsf{lub}_i\, t_i \wedge \bar{f}(R' \cup \{a'\}) = R \cup \{a\} \\
\Longrightarrow [\text{Definition } \mathcal{I}[f(P)]] \\
\quad (\underset{i}{\mathsf{lub}}\, t_i, R \cup \{a\}) \in I
\end{array}
$$

Die Aussage folgt für $\mathcal{I}[P_1 \triangle P_2]$ mittels

$$
\begin{array}{l}
\quad (\underset{i}{\mathsf{lub}}\, t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I \\
\Longrightarrow [\text{Definition } \mathcal{I}[P_1 \triangle P_2]] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \vee \\
\quad\ (\exists r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2)) \wedge \\
\quad (\forall i.(t_i \circ a, \emptyset) \notin I_1 \wedge \\
\qquad (\forall r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = t_i \circ a \Rightarrow ((r, \emptyset) \notin I_1 \vee (s, \emptyset) \notin I_2))) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad (\forall i, s'.r \circ s' = t_i \circ a \Rightarrow (s', \emptyset) \notin I_2)) \\
\Longrightarrow [\text{Definition } \mathsf{lub},\ (t_i)_{i \in \mathbb{N}} \text{ Kette}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad (\forall i, s'.r \circ s' = t_i \circ a \Rightarrow (s', \emptyset) \notin I_2) \wedge (\exists j. \forall i.r \sqsubseteq t_{i+j})) \\
\Longrightarrow [\text{Definition } \sqsubseteq] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad (\forall i, s'.r \circ s' = t_i \circ a \Rightarrow (s', \emptyset) \notin I_2) \wedge (\exists j. \forall i. \exists s'.r \circ s' = t_{i+j})) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad (\forall i, s'.r \circ s' = t_i \circ a \Rightarrow (s', \emptyset) \notin I_2) \wedge \\
\qquad (\exists j. \forall i. \exists s'.r \circ s' = t_{i+j} \wedge r \circ s' \circ a = t_{i+j} \circ a)) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad (\exists j. \forall i. \exists s'.r \circ s' = t_{i+j} \wedge (s' \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [\text{Prädikatenlogik}]
\end{array}
$$

$$
\begin{array}{l}
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists j, r, s, s'.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad\qquad (\forall i.r \circ s'_i = t_{i+j} \wedge (s'_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [(t_i)_{i\in\mathbb{N}} \text{ Kette}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists j, r, s, s'.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad\qquad \mathsf{lub}_i(r \circ s'_i) = \mathsf{lub}_i\, t_{i+j} \wedge (\forall i.(s'_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [\circ \text{ stetig im zweiten Argument}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists j, r, s, s'.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad\qquad r \circ \mathsf{lub}_i\, s'_i = \mathsf{lub}_i\, t_{i+j} \wedge (\forall i.(s'_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [\text{Definition } \mathsf{lub}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s, s'.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad\qquad r \circ \mathsf{lub}_i\, s'_i = \mathsf{lub}_i\, t_i \wedge (\forall i.(s'_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s, s'.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad\qquad r \circ \mathsf{lub}_i\, s'_i = r \circ s \wedge (\forall i.(s'_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [\text{Definition } \circ,\ r \in A_1^*] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s, s'.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R) \in I_2 \wedge \\
\qquad\qquad \mathsf{lub}_i\, s'_i = s \wedge (\forall i.(s'_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad ((\mathsf{lub}_i\, t_i, R) \in I_1 \wedge \forall i.(t_i \circ a, \emptyset) \notin I_1) \vee \\
\quad (\exists r, s'.r \in A_1^* \wedge \mathsf{lub}_i\, s'_i \neq \langle\rangle \wedge r \circ \mathsf{lub}_i\, s'_i = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (\mathsf{lub}_i\, s'_i, R) \in I_2 \wedge \\
\qquad (\forall i.(s'_i \circ a, \emptyset) \notin I_2)) \\
\Longrightarrow [\text{Induktion}] \\
\quad (\mathsf{lub}_i\, t_i, R \cup \{a\}) \in I_1 \vee \\
\quad (\exists r, s'.r \in A_1^* \wedge \mathsf{lub}_i\, s'_i \neq \langle\rangle \wedge r \circ \mathsf{lub}_i\, s'_i = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (\mathsf{lub}_i\, s'_i, R \cup \{a\}) \in I_2) \\
\Longrightarrow [\text{Prädikatenlogik}] \\
\quad (\mathsf{lub}_i\, t_i, R \cup \{a\}) \in I_1 \vee \\
\quad (\exists r, s.r \in A_1^* \wedge s \neq \langle\rangle \wedge r \circ s = \mathsf{lub}_i\, t_i \wedge (r, \emptyset) \in I_1 \wedge (s, R \cup \{a\}) \in I_2) \\
\Longrightarrow [\text{Definition } \rangle[P_1 \triangle P_2]] \\
\quad (\underset{i}{\mathsf{lub}}\, t_i, R \cup \{a\}) \in I
\end{array}
$$

Die Aussage folgt für $\mathcal{I}[P\backslash H]$ mittels

$$
\begin{array}{l}
\quad (\underset{i}{\mathsf{lub}}\, t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I \\
\Longrightarrow [\text{Präfixabschluß}] \\
\quad (\underset{i}{\mathsf{lub}}\, t_i, R) \in I \wedge (\forall i.(t_i, \emptyset) \in I) \wedge (\forall i.(t_i \circ a, \emptyset) \notin I) \\
\Longrightarrow [\text{Definition } \circ]
\end{array}
$$

$$
\begin{array}{ll}
 & (\mathop{\mathsf{lub}}_i t_i, R) \in I \wedge (\forall i.(t_i, \emptyset) \in I) \wedge (\forall i.t_i \in A^*) \\
\Longrightarrow & [\text{Definition } \mathcal{I}[P\backslash H]] \\
 & (\exists t'.(A\backslash H)©t' = \mathsf{lub}_i\, t_i \wedge (t', R \cup H) \in I') \wedge \\
 & (\forall i, s.(A\backslash H)©s \neq t_i \circ a \vee (s, H) \notin I') \wedge \\
 & (\forall i.t_i \in A^*) \\
\Longrightarrow & [\text{Hilfsatz B.5.7}] \\
 & (\exists t'.(A\backslash H)©t' = \mathsf{lub}_i\, t_i \wedge (t', R \cup H) \in I' \wedge (\forall i.\exists s.(A\backslash H)©s = t_i)) \wedge \\
 & (\forall i, s.(A\backslash H)©s \neq t_i \circ a \vee (s, H) \notin I') \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (\exists t'.(A\backslash H)©t' = \mathsf{lub}_i\, t_i \wedge (t', R \cup H) \in I' \wedge \\
 & \quad (\exists s.\forall i.(A\backslash H)©s_i = t_i)) \wedge \\
 & (\forall i, s.(A\backslash H)©s \neq t_i \circ a \vee (s, H) \notin I') \\
\Longrightarrow & [\sqsubset \text{ fundiert auf } \{s \mid s \sqsubseteq t\}] \\
 & (\exists t'.(A\backslash H)©t' = \mathsf{lub}_i\, t_i \wedge (t', R \cup H) \in I' \wedge \\
 & \quad (\forall i.\exists s.(A\backslash H)©s_i = t_i \wedge s_i \sqsubseteq s_{i+1})) \wedge \\
 & (\forall i, s.(A\backslash H)©s \neq t_i \circ a \vee (s, H) \notin I') \\
\Longrightarrow & [\text{Prädikatenlogik, } (s_i)_{i\in\mathbb{N}} \text{ Kette}] \\
 & (\exists t'.(\forall i.(A\backslash H)©t'_i = t_i \wedge t'_i \in A^*) \wedge \mathsf{lub}_i\, t'_i = \mathsf{lub}_i\, t_i \wedge (\mathsf{lub}_i\, t'_i, R \cup H) \in I') \wedge \\
 & (\forall i, s.(A\backslash H)©s \neq t_i \circ a \vee (s, H) \notin I') \\
\Longrightarrow & [\text{Definition } ©,\ a \notin H] \\
 & (\exists t'.(\forall i.(A\backslash H)©(t'_i \circ a) = t_i \circ a) \wedge \mathsf{lub}_i\, t'_i = \mathsf{lub}_i\, t_i \wedge (\mathsf{lub}_i\, t'_i, R \cup H) \in I') \wedge \\
 & (\forall i, s.(A\backslash H)©s \neq t_i \circ a \vee (s, H) \notin I') \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & \exists t'.(\forall i.(t'_i \circ a, H) \notin I') \wedge \mathsf{lub}_i\, t'_i = \mathsf{lub}_i\, t_i \wedge (\mathsf{lub}_i\, t'_i, R \cup H) \in I') \\
\Longrightarrow & [\text{Teilmengenabschluß}] \\
 & \exists t'.(\forall i.(t'_i \circ a, \emptyset) \notin I') \wedge \mathsf{lub}_i\, t'_i = \mathsf{lub}_i\, t_i \wedge (\mathsf{lub}_i\, t'_i, R \cup H) \in I') \\
\Longrightarrow & [\text{Induktion}] \\
 & \exists t'.(\mathsf{lub}_i\, t'_i = \mathsf{lub}_i\, t_i \wedge (\mathsf{lub}_i\, t'_i, R \cup H \cup \{a\}) \in I') \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & \exists t'.(t' = \mathsf{lub}_i\, t_i \wedge (t', R \cup H \cup \{a\}) \in I') \\
\Longrightarrow & [\text{Definition } \mathcal{I}[P\backslash H]] \\
 & (\mathop{\mathsf{lub}}_i t_i, R \cup \{a\}) \in I
\end{array}
$$

Die Aussage folgt für $\mathcal{I}[\mu X : A.F(X)]$ mittels

$$
\begin{array}{ll}
 & (\mathop{\mathsf{lub}}_i t_i, R) \in I \wedge \forall i.(t_i \circ a, \emptyset) \notin I \\
\Longrightarrow & [\text{Definition } \mathcal{I}[\mu X : A.F(X)]] \\
 & (\forall j.(\mathop{\mathsf{lub}}_i t_i, R) \in I_j) \wedge (\forall i.\forall j.(t_i \circ a, \emptyset) \notin I_j) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & (\forall j.(\mathop{\mathsf{lub}}_i t_i, R) \in I_j) \wedge (\forall j.\forall i.(t_i \circ a, \emptyset) \notin I_j) \\
\Longrightarrow & [\text{Prädikatenlogik}]
\end{array}
$$

$$
\begin{array}{ll}
& (\forall j.(\operatorname{\mathsf{lub}}_i t_i, R) \in I_j \wedge \forall i.(t_i \circ a, \emptyset) \notin I_j) \\
\Longrightarrow & \text{[Induktion]} \\
& \forall j.(\operatorname{\mathsf{lub}}_i t_i, R \cup \{a\}) \in I_j \\
\Longrightarrow & \text{[Definition } \mathcal{I}[\mu X : A.F(X)]\text{]} \\
& (\operatorname{\mathsf{lub}}_i t_i, R \cup \{a\}) \in I
\end{array}
$$

B.5 Lemmata

In den vorangegangenen Abschnitten wurden wiederholt Aussagen verwendet, die nicht spezifisch für die in dieser Arbeit vorgestellen Ansätze gelten, sondern allgemeiner Natur sind. Daher wurden diese wiederholt auftretenden Aussagen zusammen mit ihren Beweisen in den folgenden Abschnitten zusammengefaßt.

Als Beweisbasis werden Prädikatenlogik höherer Stufe und die Mengentheorie nach Zermelo-Fränklin verwendet, wie sie beispielsweise in [Pau94b] beschrieben sind. Schlüsse, die ausschließliche Theoreme aus diesen Logiken verwenden, werden mit folgenden allgemein mit der Begründung *Prädikatenlogik* versehen.

B.5.1 Algebraische Aussagen über Prozesse

Die folgenden Hilfsätze beschreiben algebraische Eigenschaften von Prozessen der unendlichen Failuresemantik. Zur vereinfachten Formulierung werden die Konstruktionsoperatoren direkt auf Prozesse anstatt auf Prozeßterme angewendet. Dabei sind die Operatoren durch die Semantik der entsprechenden syntaktischen Konstruktionsoperatoren definiert.

Satz B.5.1 (Verbergen disjunkter Alphabete) Für zwei Prozesse P und Q mit $\alpha P \cap \alpha Q = \emptyset$ gilt

$$(P \parallel Q)\backslash B = (P\backslash(B \cap \alpha P)) \parallel (Q\backslash(B \cap \alpha Q)) \tag{B.15}$$

•

Beweis B.5.1 (Satz B.5.1) Für den Nachweis der Aussage B.15 wird

$$(t, R) \in (P \parallel Q)\backslash B \Leftrightarrow (t, R) \in (P\backslash(B \cap \alpha P)) \parallel (Q\backslash(B \cap \alpha Q))$$

gezeigt. Dabei werden die Abkürzungen $A_P = \alpha P$, $A_Q = \alpha Q$ und $A = A_P \cup A_Q$ verwendet.

$$
\begin{array}{ll}
& (t, R) \in (P \parallel Q)\backslash B \\
\Longleftrightarrow & \text{[Definition } (P \parallel Q)\backslash B\text{]} \\
& \exists t'.(A\backslash B)\copyright t' = t \wedge (t', R \cup B) \in P \parallel Q \\
\Longleftrightarrow & \text{[Definition } P \parallel Q\text{]}
\end{array}
$$

$$
\begin{array}{ll}
& \exists t'.\ (A\backslash B)\copyright t' = t \wedge \\
& \quad \exists R_P, R_Q.\ (A_P\copyright t', R_P) \in P \wedge (A_Q\copyright t', R_Q) \in Q \wedge \\
& \qquad R_P \cup R_Q = R \cup B \\
\Longleftrightarrow & [A_P \cap A_Q = \emptyset,\ R_P \subseteq A_P,\ R_Q \subseteq A_Q] \\
& \exists t'.\ (A\backslash B)\copyright t' = t \wedge \\
& \quad \exists R_P, R_Q.\ (A_P\copyright t', R_P) \in P \wedge (A_Q\copyright t', R_Q) \in Q \wedge \\
& \qquad R_P = (R \cup B) \cap A_P \wedge R_Q = (R \cup B) \cap A_Q \\
\Longleftrightarrow & [\text{Prädikatenlogik}] \\
& \exists t'.\ (A\backslash B)\copyright t' = t \wedge \\
& \quad (A_P\copyright t', (R \cup B) \cap A_P) \in P \wedge (A_Q\copyright t', (R \cup B) \cap A_Q) \in Q \\
\Longleftrightarrow & [\text{Siehe } * \text{ unten}] \\
& \exists t'_P.(t'_P, (R \cup B) \cap A_P) \in P \wedge (A_P\backslash B)\copyright t'_P = (A_P\backslash B)\copyright t \wedge \\
& \exists t'_Q.(t'_Q, (R \cup B) \cap A_Q) \in Q \wedge (A_Q\backslash B)\copyright t'_Q = (A_Q\backslash B)\copyright t \\
\Longleftrightarrow & [\text{Prädikatenlogik}] \\
& \exists R_P, R_Q.\ \exists t'_P.(t'_P, R_P \cup (B \cap A_P)) \in P \wedge (A_P\backslash B)\copyright t'_P = (A_P\backslash B)\copyright t \wedge \\
& \qquad \exists t'_Q.(t'_Q, R_Q \cup (B \cap A_Q)) \in Q \wedge (A_Q\backslash B)\copyright t'_Q = (A_Q\backslash B)\copyright t \wedge \\
& \qquad R_P = (R \cap A_P) \wedge \\
& \qquad R_Q = (R \cap A_Q) \\
\Longleftrightarrow & [A_P \cap A_Q = \emptyset,\ R \subseteq A_P \cup A_Q,\ R_P \subseteq A_P,\ R_Q \subseteq A_Q] \\
& \exists R_P, R_Q.\ \exists t'_P.(t'_P, R_P \cup (B \cap A_P)) \in P \wedge (A_P\backslash B)\copyright t'_P = (A_P\backslash B)\copyright t \wedge \\
& \qquad \exists t'_Q.(t'_Q, R_Q \cup (B \cap A_Q)) \in Q \wedge (A_Q\backslash B)\copyright t'_Q = (A_Q\backslash B)\copyright t \wedge \\
& \qquad R_P \cup R_Q = R \\
\Longleftrightarrow & [\text{Definition } P\backslash(B \cap A_P), Q\backslash(B \cap A_Q)] \\
& \exists R_P, R_Q.((A_P\backslash B)\copyright t, R_P) \in P \wedge ((A_Q\backslash B)\copyright t, R_Q) \in Q \wedge R_P \cup R_Q = R \\
\Longleftrightarrow & [\text{Definition } (P\backslash(B \cap A_P)) \parallel (Q\backslash(B \cap A_Q))] \\
& (t, R) \in (P\backslash(B \cap A_P)) \parallel (Q\backslash(B \cap A_Q))
\end{array}
$$

Dabei folgt der Schluß $*$ in Richtung $\Longrightarrow$ unmittelbar bei Wahl von $t'_P = A_P\copyright t'$ und $t'_Q = A_Q\copyright t'$ aus den Eigenschaften von $\sqsubseteq$ und $\copyright$, die Richtung $\Longleftarrow$ wegen $A_P \cap A_Q = \emptyset$ unter Anwendung von Hilfsatz B.5.8 auf t'_P und t'_Q. □

Satz B.5.2 (Wiederholtes Verbergen) Für einen Prozeß P gilt

$$(P\backslash B)\backslash C = P\backslash(B \cup C) \tag{B.16}$$

•

Beweis B.5.2 (Satz B.5.2) Für den Nachweis von Aussage B.16 wird die Aussage

$$(t, R) \in (P\backslash B)\backslash C \Leftrightarrow (t, F) \in P\backslash(B \cup C)$$

gezeigt. Dabei werden die Abkürzungen $\bar{A} = A\backslash B$ und $\bar{\bar{A}} = (A\backslash B)\backslash C$ verwendet.

$$
\begin{array}{ll}
& (t, R) \in (P\backslash B)\backslash C \\
\Longleftrightarrow & [\text{Definition } (P\backslash B)\backslash C] \\
& \exists t'.(t', R \cup C) \in P\backslash B \wedge \bar{\bar{A}}\copyright t' = t
\end{array}
$$

$\Longleftrightarrow$ [Definition $(P\backslash B)$]
$\exists t', t''.(t'', (R \cup C) \cup B) \in P \wedge \bar{A}©t'' = t' \wedge \bar{\bar{A}}©t' = t$
$\Longleftrightarrow$ [Prädikatenlogik]
$\exists t''.(t'', (R \cup C) \cup B) \in P \wedge \bar{\bar{A}}©(\bar{A}©t'') = t$
$\Longleftrightarrow$ [Prädikatenlogik, Definition ©]
$\exists t''.(t'', R \cup (C \cup B)) \in P \wedge \bar{\bar{A}}©t'' = t$
$\Longleftrightarrow$ [$t'' = t'$]
$\exists t'.(t', R \cup (B \cup C)) \in P \wedge \bar{\bar{A}}©t' = t$
$\Longleftrightarrow$ [Definition $P\backslash(B \cup C)$]
$(t, F) \in P\backslash(B \cup C)$

□

Satz B.5.3 (Umbenennung und Parallelkomposition) Seien P und Q Prozesse, sowie f eine Umbenennung. Dann gilt

$$f(P \parallel Q) = f(P) \parallel f(Q) \tag{B.17}$$

•

Beweis B.5.3 (Satz B.5.3) Im folgenden wird mit A_P das Alphabet von P, sowie mit A_Q das Alphabet von Q bezeichnet. Entsprechend wird mit $A_{f(P)}$ das Alphabet von $f(P)$, sowie mit $A_{f(Q)}$ das Alphabet von $f(Q)$ bezeichnet. Der Nachweis der Eigenschaft B.17 erfolgt mittels

$f(P \parallel Q)$
$=$ [Definition f]
$\{(f^*(t), \bar{f}(R)) \mid (t, R) \in P \parallel Q\}$
$=$ [Definition $\|$]
$\{(f^*(t), \bar{f}(R)) \mid \exists R_P, R_Q.(A_P©t, R_P) \in P \wedge (A_Q©t, R_Q) \in Q \wedge R_P \cup R_Q = R\}$
$=$ [Definition f^*,$\bar{f}$,$f(P)$,$f(Q)$]
$\{(f^*(t), \bar{f}(R)) \mid \exists R_P, R_Q.\ (\bar{f}(A_P)©f^*(t), \bar{f}(F_P)) \in f(P) \wedge$
$(\bar{f}(A_Q)©f^*(t), \bar{f}(F_Q)) \in f(Q) \wedge$
$\bar{R} = \bar{f}(R_P) \cup \bar{f}(R_Q)\}$
$=$ [Prädikatenlogik]
$\{(t, R) \mid \exists R_P, R_Q.\ (\bar{f}(A_P)©t, R_P) \in f(P) \wedge$
$(\bar{f}(A_Q)©t, R_Q) \in f(Q) \wedge$
$R_P \cup R_Q = R\}$
$=$ [Definition $\|$]
$f(p) \parallel f(Q)$

□

Satz B.5.4 (Umbenennung und Verbergen) Für einen Prozeß P mit $\alpha P = A$, eine Menge von Aktionen $B \subseteq A$ und eine Umbenennung $f : A \to C$ gilt

$$f(P\backslash B) = f(P)\backslash\bar{f}(B) \tag{B.18}$$

•

Beweis B.5.4 (Satz B.5.4) Für den Nachweis der Aussage B.18 wird die Aussage

$$(t, R) \in f(P\backslash B) \Leftrightarrow (t, R) \in f(P)\backslash\bar{f}(B)$$

gezeigt. Im folgenden sei $A = \alpha P$. Dann folgt die Behauptung mittels

$$\begin{array}{ll}
& (t, R) \in f(P\backslash B) \\
\Longleftrightarrow & [\text{Definition } f(P\backslash B)] \\
& \exists t', R'.(t', R') \in P\backslash B \wedge t = f^*(t') \wedge R = \bar{f}(R') \\
\Longleftrightarrow & [\text{Definition } P\backslash B] \\
& \exists t', R'.(t', R') \in P \wedge B \subseteq R' \wedge t = f^*((A\backslash B)©t') \wedge R = \bar{f}(R'\backslash B) \\
\Longleftrightarrow & [\text{Definition } f^*,\ \bar{f}] \\
& \exists t', R'.(t', R') \in P \wedge \bar{f}(B) \subseteq \bar{f}(R') \wedge t = \bar{f}(A\backslash B)©f^*(t') \wedge R = \bar{f}(R')\backslash\bar{f}(B) \\
\Longleftrightarrow & [\text{Definition } f(P)] \\
& \exists t', R'.(t', R') \in f(P) \wedge \bar{f}(B) \subseteq R' \wedge t = \bar{f}(A\backslash B)©t' \wedge R = R'\backslash\bar{f}(B) \\
\Longleftrightarrow & [\text{Definition } \bar{f}] \\
& \exists t', R'.(t', R') \in f(P) \wedge \bar{f}(B) \subseteq R' \wedge t = (\bar{f}(A)\backslash\bar{f}(B))©t' \wedge R = R'\backslash\bar{f}(B) \\
\Longleftrightarrow & [\text{Definition } f(P)\backslash\bar{f}(B)] \\
& (t, R) \in f(P)\backslash\bar{f}(B)
\end{array}$$

□

Satz B.5.5 (Erweiterung der Umbenennung) Seien f und g zwei Umbenennungen mit $f : B \to D$ und $g : C \to D$, sowie P ein Prozeß mit $\alpha P = A$, wobei $A \subseteq B$ und $A \subseteq C$. Gilt

$$\forall a \in A.f(a) = g(a) \tag{B.19}$$

so folgt

$$f(P) = g(P)$$

•

Beweis B.5.5 (Satz B.5.5) Die Aussage folgt mittels

$$\begin{array}{ll}
& f(P) \\
= & [\text{Definition } f(P)] \\
& \{f^*(t), \bar{f}(R)) \mid (t, R) \in P\} \\
= & [t \in A^\omega,\ R \subseteq A, \text{ Eigenschaft B.19}] \\
& \{g^*(t), \bar{g}(R)) \mid (t, R) \in P\} \\
= & [\text{Definition } g(P)] \\
& g(P)
\end{array}$$

□

B.5.2 Hilfsaussagen über endliche Mengen

Im folgenden werden wiederholt Aussagen über endliche Mengen benötigt. Daher werden diese explizite eingeführt. Dabei wird die Endlichkeit einer Menge mit $\mathbb{F}(A)$ bezeichnet.

Neben allgemein verwendeten und daher hier nicht bewiesenen Aussagen wie

- der Endlichkeit von Teilmengen endlicher Mengen

  $\mathbb{F}(A) \wedge B \subseteq A \Rightarrow \mathbb{F}(B)$

- der Endlichkeit von Produkten endlicher Mengen

  $\mathbb{F}(A) \wedge \mathbb{F}(B) \Rightarrow \mathbb{F}(A \times B)$

- der Endlichkeit der Potenzmenge endlicher Mengen

  $\mathbb{F}(A) \Rightarrow \mathbb{F}(\mathbb{P}(A))$

- der Endlichkeit der Bildmenge endlicher Mengen

  $\mathbb{F}(A) \Rightarrow \mathbb{F}(\{f(a) \mid a \in A\})$

werden auch einige Endlichkeitsaussagen bezüglich der in dieser Arbeit verwendeten Operatoren benötigt.

Hilfsatz B.5.1 (Endlichkeit der Präfixmenge) Zu gegebener Alphabetsmenge ist die Menge der Präfizes einer endlichen Spur über A endlich:

$$t \in A^* \Rightarrow \mathbb{F}(\{r \mid r \sqsubseteq t\})$$

△

Beweis B.5.6 (SatzB.5.1) Der Beweis erfolgt mittels Induktion über den Aufbau von t:[4]

$t = \langle\rangle$**:** Damit folgt

$\mathbb{F}(\emptyset)$
$\Longrightarrow$ [Prädikatenlogik]
$\mathbb{F}(\emptyset \cup \{\langle\rangle\})$
$\Longrightarrow$ [Definition $\sqsubseteq$]
$\mathbb{F}(\{r \mid r \sqsubseteq \langle\rangle\})$
$\Longrightarrow$ [$t = \langle\rangle$]
$\mathbb{F}(\{r \mid r \sqsubseteq t\})$

$t = a \circ s$**:** Dann folgt aus der Induktionsvoraussetzung

$\mathbb{F}(\{r \mid r \sqsubseteq s\})$
$\Longrightarrow$ [Endlichkeit der Bildmenge endlicher Mengen]
$\mathbb{F}(\{a \circ r \mid r \sqsubseteq s\})$

[4] Für das entsprechende Induktionsprinzip siehe z.B. [Reg94].

$\Longrightarrow$ [$\mathbb{F}(\{\langle\rangle\})$, Prädikatenlogik]
$\mathbb{F}(\{\langle\rangle\} \cup \{a \circ r \mid r \sqsubseteq s\})$
$\Longrightarrow$ [Definition $\circ$, $\sqsubseteq$]
$\mathbb{F}(\{r \mid r \sqsubseteq a \circ s\})$
$\Longrightarrow$ [$t = a \circ s$]
$\mathbb{F}(\{r \mid r \sqsubseteq t\})$

□

Hilfsatz B.5.2 (Fairnessmengenendlichkeit) Es gilt

$$t \in A^* \wedge \mathbb{F}(B) \Rightarrow \mathbb{F}(\{(B©r, R) \mid r \sqsubseteq t \wedge R \subseteq B\})$$

△

Beweis B.5.7 (Hilfsatz B.5.2) Die Aussage folgt unmittelbar aus der Endlichkeit der Potenzmenge endlicher Mengen, der Endlichkeit von Teilmengen endlicher Mengen, der Endlichkeit der Präfizesmenge endlicher Spuren (Hilfsatz B.5.1), der Endlichkeit von Bildern endlicher Mengen und der Endlichkeit von Produktmengen. □

B.5.3 Hilfsaussagen über Ketten

Im folgenden werden Hilfsaussagen über Folgen c über einem Wertebereich C, also totale Funktionen $c : \mathbb{N} \to C$ bzw. Ketten, also Folgen mit $\forall i, j.i \leq j \Rightarrow c_i \sqsubseteq c_j$[5] bei gegebender Ordnung $\sqsubseteq$ auf C, hergeleitet, die für die Beweise in Abschnitt B.2 verwendet werden.

Ketten mit endlichen Wertebereichen

Hilfsatz B.5.3 zeigt, daß für nichtendliche Folgen (und damit auch Ketten) über endlichen Wertebereichen ein Wert gefunden werden kann, der unbeschränkt oft in der Folge (bzw. Kette) auftritt. Als Beweisprinzip wird die Induktion über den Aufbau beliebiger endlicher Mengen verwendet, wie sie beispielsweise in [Pau94b] vorgestellt wird.

Hilfsatz B.5.3 (Endliche Wertebereiche von Folgen) Für eine Menge C mit $\mathbb{F}(C)$ und ein Folge $(c_i)_{i \in \mathbb{N}}$ gilt

$$(\forall i.c_i \in C) \Rightarrow \exists k.\forall i.\exists j.i \leq j \wedge c_j = k$$

△

Beweis B.5.8 (Hilfsatz B.5.3) Der Beweis erfolgt mittels Induktion über den Aufbau endlicher Mengen:

[5] Für Folgen - und entsprechend auch für Ketten - wird hier die Indexschreibweise c_i statt der Funktionsschreibweise $c(i)$ verwendet.

$C = \emptyset$: Die Behauptung folgt unmittelbar aus

$$
\begin{aligned}
& C = \emptyset \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \nexists i.c_i \in C \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& (\forall i.c_i \in C) \Rightarrow \exists k.\forall i.\exists j.i \leq j \wedge c_j = k
\end{aligned}
$$

$C = A \cup \{a\}$: Der Nachweis der Aussage für den Induktionsschritt erfolgt mittels Fallunterscheidung nach $\forall i.\exists j.i \leq j \wedge c_j = a$. In diesem Fall folgt die Behauptung mit der Wahl von $k = a$. Andernfalls gilt $\exists i.\forall j.i \leq j \Rightarrow c_j \neq a$ und es folgt

$$
\begin{aligned}
& (\forall i.c_i \in C) \wedge (\exists i.\forall j.i \leq j \Rightarrow c_j \neq a) \\
\Longrightarrow\ & [C = A \cup \{a\}] \\
& (\forall i.c_i \in A \vee c_i = a) \wedge (\exists i.\forall j.i \leq j \Rightarrow c_j \neq a) \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists i.\forall j.i \leq j \Rightarrow c_j \in A \\
\Longrightarrow\ & [\text{Definition } +] \\
& \exists i.\forall j.c_{i+j} \in A \\
\Longrightarrow\ & [\text{Induktion}] \\
& \exists i,k.\forall j.\exists l.j \leq l \wedge c_{i+l} = k \\
\Longrightarrow\ & [\text{Prädikatenlogik, Definition } +] \\
& \exists k.\forall j.\exists l.j \leq l \wedge c_l = k
\end{aligned}
$$

□

Für Ketten über endliche Wertebereiche läßt sich eine entsprechend schärfere Aussage herleiten:

Hilfsatz B.5.4 (Endliche Wertebereiche von Ketten) Für eine Menge C mit $\mathbb{F}(C)$ und ein hinsichtlich $\sqsubseteq$ geordnete Kette $(c_i)_{i \in \mathbb{N}}$ gilt

$$(\forall i.c_i \in C) \Rightarrow \exists j.\forall i.c_{i+j} = c_j$$

△

Beweis B.5.9 (Hilfsatz B.5.4) Der Hilfsatz folgt mit wesentlichen aus Hilfsatz B.5.3

$$
\begin{aligned}
& \forall i.c_i \in C \\
\Longrightarrow\ & [\text{Hilfsatz B.5.3}] \\
& \exists k.\forall i.\exists j.i \leq j \wedge c_j = k \\
\Longrightarrow\ & [(c_i)_{i \in \mathbb{N}} \text{ Kette}] \\
& \exists k.\forall i.\exists j.c_i \sqsubseteq c_j \wedge c_j = k \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists k.\forall i.\exists j.c_i \sqsubseteq k \wedge c_j = k \\
\Longrightarrow\ & [\text{Prädikatenlogik}] \\
& \exists k.\forall i.(c_i \sqsubseteq k \wedge \exists j.c_j = k) \\
\Longrightarrow\ & [\text{Prädikatenlogik}]
\end{aligned}
$$

$$
\begin{array}{ll}
& \exists k.((\forall i.c_i \sqsubseteq k) \wedge (\exists j.c_j = k)) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& \exists j.\forall i.c_i \sqsubseteq c_j \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& \exists j.\forall i.c_{i+j} \sqsubseteq c_j \\
\Longrightarrow & [(c_i)_{i\in\mathbb{N}} \text{ Kette}] \\
& \exists j.\forall i.c_{i+j} \sqsubseteq c_j \wedge c_j \sqsubseteq c_{j+i} \\
\Longrightarrow & [\text{Definition } \sqsubseteq] \\
& \exists j.\forall i.c_{i+j} = c_j
\end{array}
$$

□

Teilkettenkonstruktion

Die folgenden beiden Hilfsätze zeigen, daß aus einer Kette durch geeignete Auswahl einer Teilmenge der Kettenelemente wiederum eine Kette entsteht.

Hilfsatz B.5.5 (Teilkettenkonstruktion) Für eine Kette $(t_i)_{i\in\mathbb{N}}$ gilt

$$(\forall i.\exists j.P(t_{i+j})) \Rightarrow (\exists s.(\forall i.P(s_i)) \wedge \operatorname*{lub}_i s_i = \operatorname*{lub}_i t_i) \tag{B.20}$$

△

Beweis B.5.10 (Hilfsatz B.5.5) Im folgenden gelte

$$\forall i.\exists j.P(t_{i+j}) \tag{B.21}$$

Aus B.21 folgt

$$\exists k.P(t_k)$$

Damit wird s konstruktiv angegeben als

$$
\begin{array}{lcl}
s_0 & \stackrel{\text{def}}{=} & t_k \\
s_{i+1} & \stackrel{\text{def}}{=} & \begin{cases} t_{i+k+1} & \text{falls } P(t_{i+k+1}) \\ s_i & \text{falls } \neg P(t_{i+k+1}) \end{cases}
\end{array}
$$

Die Aussage $\forall i.P(s_i)$ folgt mittels Induktion über i:

$i = 0$**:** Dann gilt

$$
\begin{array}{ll}
& P(t_k) \\
\Longrightarrow & [\text{Definition } s] \\
& P(s_0)
\end{array}
$$

$i > 0$**:** Dann gilt

$$
\begin{array}{ll}
& P(t_{i+k+1}) \vee \neg P(t_{i+k+1}) \\
\Longrightarrow & [\text{Definition } s]
\end{array}
$$

$$
\begin{array}{ll}
 & (P(t_{i+k+1}) \wedge s_{i+1} = t_{i+k+1}) \vee (s_{i+1} = s_i) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & P(s_{i+1}) \vee (P(s_{i+1} \Leftrightarrow P(s_i)) \\
\Longrightarrow & [\text{Induktion}] \\
 & P(s_{i+1})
\end{array}
$$

Weiterhin gilt $\forall i.s_i \sqsubseteq t_{k+i}$, wie mittels Induktion folgt:

$i = 0$**:** Dann folgt mittels Definition von s $s_0 = t_k$ und damit $s_0 \sqsubseteq t_{k+0}$.

$i > 0$**:** Dann gilt

$$
\begin{array}{ll}
\Longrightarrow & [\text{Induktion}] \\
 & s_i \sqsubseteq t_{k+i} \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & P(t_{k+i+1}) \vee (s_i \sqsubseteq t_{k+i} \wedge \neg P(t_{k+i+1})) \\
\Longrightarrow & [\text{Definition } s] \\
 & s_{i+1} = t_{k+i+1} \vee (s_i \sqsubseteq t_{k+i} \wedge s_{i+1} = s_i) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & s_{i+1} = t_{k+i+1} \vee s_{i+1} \sqsubseteq t_{k+i} \\
\Longrightarrow & [(t_i)_{i \in \mathbb{N}} \text{ Kette}] \\
 & s_{i+1} = t_{k+i+1} \vee s_{i+1} \sqsubseteq t_{k+i+1} \\
\Longrightarrow & [\text{Definition } \sqsubseteq] \\
 & s_{i+1} \sqsubseteq t_{k+i+1}
\end{array}
$$

Damit folgt $\forall i.s_i \sqsubseteq s_{i+1}$, denn

$$
\begin{array}{ll}
 & P(t_{i+k+1}) \vee \neg P(t_{i+k+1}) \\
\Longrightarrow & [\text{Definition } s] \\
 & s_{i+1} = t_{i+k+1} \vee s_{i+1} = s_i \\
\Longrightarrow & [(t_i)_{i \in \mathbb{N}} \text{ Kette}] \\
 & (s_{i+1} = t_{i+k+1} \wedge t_{i+k} \sqsubseteq t_{i+k+1}) \vee s_{i+1} = s_i \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
 & t_{i+k} \sqsubseteq s_{i+1} \vee s_{i+1} = s_i \\
\Longrightarrow & [s_i \sqsubseteq t_{k+i}] \\
 & s_i \sqsubseteq s_{i+1} \vee s_{i+1} = s_i \\
\Longrightarrow & [\text{Definition } \sqsubseteq] \\
 & s_i \sqsubseteq s_{i+1}
\end{array}
$$

Damit ist $(s_i)_{i \in \mathbb{N}}$ ebenfalls eine Kette und erfüllt wegen B.21 die Eigenschaft

$$\forall i.\exists j.t_i = s_j$$

Damit folgt (vgl. [Reg94]) schließlich

$$\bigsqcup_i t_i = \bigsqcup_i s_i$$

□

Hilfsatz B.5.6 (Alternativkettenkonstruktion) Für eine Kette $(t_i)_{i\in\mathbb{N}}$ gilt

$$(\forall i.P(t_i) \vee Q(t_i)) \Rightarrow (\exists s.\, \underset{i}{\mathsf{lub}}\, s_i = \underset{i}{\mathsf{lub}}\, t_i \wedge (\forall i.P(s_i) \vee \forall i.Q(s_i)) \tag{B.22}$$

△

Beweis B.5.11 (Hilfsatz B.5.6) Ohne Beschränkung der Allgemeinheit wird für den Nachweis von Aussage B.22 die Eigenschaft

$$\forall i.\exists j.P(t_{i+j}) \tag{B.23}$$

angenommen, da anderenfalls wegen $\forall i.P(t_i) \vee Q(t_i)$ die entsprechende Aussage für Q gilt. Damit folgt die Aussage mittels Hilfsatz B.5.5. □

Projektionsfunktion

In den vorangegangenen Beweisen wird wiederholt die Aussage angewendet, daß zu einem endlichen Präfix des Bildes der Projektionsfunktion © auch ein dazu passender endlicher Präfix des Urbildes gefunden werden kann. Dies wird im folgenden Satz formalisiert:

Hilfsatz B.5.7 (Urbildpräfix) Sei A eine Alphabetmenge mit $B \subseteq A$ und $t \in A^\omega$. Dann gilt

$$s \in A^* \wedge s \sqsubseteq B©t \Rightarrow (\exists r \in A^*.r \sqsubseteq t \wedge B©r = s) \tag{B.24}$$

△

Beweis B.5.12 (Hilfsatz B.5.7) Im folgenden wird – da B in der Aussage B.24 fest gewählt ist – $B©r$ abgekürzt zu $f(r)$.

Wegen der Stetigkeit von f gilt

$$\exists u \in A^*.u \sqsubseteq t \wedge s \sqsubseteq f(u) \tag{B.25}$$

da sonst

$$\begin{array}{ll}
& \forall u \in A^*.u \sqsubseteq t \Rightarrow s \not\sqsubseteq f(u) \\
\Longrightarrow & [f(u) \sqsubseteq f(t),\ s \sqsubseteq f(t),\ \text{Definition } \sqsubset] \\
& \forall u \in A^*.u \sqsubseteq t \Rightarrow f(u) \sqsubset s \\
\Longrightarrow & [\text{Definition } \sqsubset] \\
& \forall u \in A^*.\exists a \in A, s \in s'.\ u \sqsubseteq t \Rightarrow f(u) \sqsubseteq s' \wedge s = s' \circ s \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& \exists a \in A, s \in s'.\ s = s' \circ a \wedge (\forall u \in A^*.\ u \sqsubseteq t \Rightarrow f(u) \sqsubseteq s') \\
\Longrightarrow & [f \text{ stetig}, \sqsubseteq \text{ zulässig}] \\
& \exists a \in A, s \in s'.\ s = s' \circ a \wedge f(\mathsf{lub}_{u\in A^*\wedge u\sqsubseteq t}\, u) \sqsubseteq s' \\
\Longrightarrow & [\text{Definition } \mathsf{lub}] \\
& \exists a \in A, s \in s'.\ s = s' \circ a \wedge f(t) \sqsubseteq s' \\
\Longrightarrow & [\text{Definition } \sqsubset]
\end{array}$$

$$
\begin{array}{ll}
& f(t) \sqsubset s \\
\Longrightarrow & [\text{Definition } \sqsubseteq] \\
& s \not\sqsubseteq f(t)
\end{array}
$$

im Widerspruch zur Voraussetzung von B.24 gilt.

Weiterhin folgt aus $s \sqsubseteq f(t)$ und der Monotonie von f

$$\forall r \in A^*.r \sqsubseteq t \Rightarrow (f(r) \sqsubseteq s \vee s \sqsubseteq f(r)) \tag{B.26}$$

Da

$$f(\langle\rangle) \sqsubseteq s$$

gilt, folgt damit aus Eigenschaft B.26, der Eigenschaft B.25 und der Monotonie von f

$$\exists r \in A^*, a \in A.r \circ a \sqsubseteq t \wedge f(r) \sqsubseteq s \wedge s \sqsubseteq f(r \circ a)$$

Damit folgt für dieses $r \in A^*$ und $a \in A$:

$$
\begin{array}{ll}
& f(r) \sqsubseteq s \wedge s \sqsubseteq f(r \circ a) \\
\Longrightarrow & [\text{Definition } f] \\
& f(r) \sqsubseteq s \wedge s \sqsubseteq f(r \circ a) \wedge (f(r \circ a) = f(r) \vee \exists b \in A.f(r \circ a) = f(r) \circ b) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& (f(r) \sqsubseteq s \wedge s \sqsubseteq f(r \circ a) \wedge f(r \circ a) = f(r)) \vee \\
& (f(r) \sqsubseteq s \wedge s \sqsubseteq f(r \circ a) \wedge \exists b \in A.f(r \circ a) = f(r) \circ b) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& (f(r) \sqsubseteq s \wedge s \sqsubseteq f(r)) \vee \\
& (\exists b \in A.f(r) \sqsubseteq s \wedge s \sqsubseteq f(r) \circ b \wedge f(r \circ a) = f(r) \circ b) \\
\Longrightarrow & [\text{Definition } \sqsubseteq] \\
& f(r) = s \vee \\
& (\exists b \in A.(f(r) = s \vee f(r) \circ b = s) \wedge f(r \circ a) = f(r) \circ b) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& f(r) = s \vee \\
& f(r) = s \vee (\exists b \in A.f(r) \circ b = s \wedge f(r \circ a) = f(r) \circ b) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& f(r) = s \vee f(r \circ a) = s
\end{array}
$$

Insgesamt folgt damit

$$
\begin{array}{ll}
& \exists r \in A^*, a \in A.r \circ a \sqsubseteq t \wedge (f(r) = s \vee f(r \circ a) = s) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& (\exists r \in A^*, a \in A.r \circ a \sqsubseteq t \wedge f(r) = s) \vee \\
& (\exists r \in A^*, a \in A.r \circ a \sqsubseteq t \wedge f(r \circ a) = s) \\
\Longrightarrow & [\text{Definition } \sqsubseteq] \\
& (\exists r \in A^*.r \sqsubseteq t \wedge f(r) = s) \vee \\
& (\exists r \in A^*, a \in A.r \circ a \sqsubseteq t \wedge f(r \circ a) = s) \\
\Longrightarrow & [\text{Prädikatenlogik}] \\
& \exists r \in A^*.r \sqsubseteq t \wedge f(r) = s
\end{array}
$$

und damit die Behauptung. □

B.5.4 Verflechten von Spuren

Im in dieser Arbeit vorgestellten Ansatz spielt die Unterscheidung zwischen endlichen, partiellen und unendliche Spuren bzw. Sequenzen von Aktionen keine Rolle. Dies bedeutet, daß nicht unterschieden wird, ob eine Spur durch eine endliche, terminierende Berechnung, eine partielle Berechnung oder durch eine unbeschränkt Aktionen produzierende Berechnung entstanden ist. Diese Unterscheidung ist nicht nötig, da hier Spuren nur als Beobachtungen verstanden werden, nicht aber als Ergebnis einer Berechnung. Im Gegensatz dazu wird in denotationellen Semantiken, die näher an der Charakterisierung ausführbarer Berechnungen und damit Implementierungen liegen, genauer unterschieden; dies ist beispielsweise bei den stromverarbeitenden Funktionen ([BDD+93]) und deren Formalisierung in Logik höherer Stufe (HOLCF, [Reg94]). HOLCF stellt ausschließlich partielle und unendliche Spuren zur Verfügung; endliche Spuren können aber als partielle Ströme mit einem abschließenden Terminierungselement dargestellt werden (vgl. [Ded92]).

Hilfsatz B.5.8 (Verflechtung von Spuren) Für zwei Alphabetmengen A_1 und A_2 gilt

$$\forall t_1 \in A_1^\omega, t_2 \in A_2^\omega . A_2 © t_1 = A_1 © t_2 \Rightarrow \exists t \in (A_1 \cup A_2)^\omega . (A_1 © t = t_1 \wedge A_2 © t = t_2)$$

△

Für den Nachweis der Aussage reicht es, diese Aussage für alle endlichen und unendliche, nicht aber für die partiellen Spuren nachzuweisen.[6] Für diese Klasse von Spuren ist dieser Nachweis sogar leicht durch Konstruktion in Form einer Funktion über t_1 und t_2 zu führen. Die folgende Funktion gibt eine an HOLCF angelehnte Fassung an:

```
merge = fix'
        lambda m. lambda t1,t2.
           if ft't1 == term then t2
           elseif ft't2 == term then t1
           elseif A2'ft't1 and A1'ft't2 then ft't1##m'(rt't1)'(rt't2)
           elseif A2'ft't1 then ft't2##m't1'(rt't2)
           elseif A1'ft't2 then ft't1##m'(rt't1)'t2
           else ft't1##ft't2##m'(rt't1)'(rt't2)
```

wobei

- `fix` den Fixpunkt eines stetigen Funktionals,
- `lambda` die Lambdaabstraktion,
- `f'a` die gecurryte Anwendung der Funktion`f` auf das Argument `a`,
- `ft` die Funktion des ersten Elements einer Spur,

[6] Die Aussage gilt auch für die Menge der partiellen und unendliche Ströme, also z.B. in HOLCF; dort ist der Nachweis jedoch aufwendiger, da keine monotone Funktion angegeben werden kann.

- `rt` die Funktion des Rests einer Spur abzüglich des ersten Elements,
- `##` das Voranstellen eines Elements vor eine Spur
- `A1` bzw. `A2` das charakterisierende Prädikat von a_1 bzw. A_2

bezeichnet. Der Nachweis, daß die Funktion die Anforderung erfüllt, erfolgt durch Induktion über t_1 und t_2 und mittels Fallunterscheidung.

B.5.5 Punktweise und elementweise Erweiterung

Für Alphabetsumbenennungen $f : A \to B$ wurden in den Definitionen 2.2.1 und B.1.1 Erweiterung der Abbildungen auf Spuren mittels f^* bzw. auf Mengen mittels $\bar{f}$. In den vorangegangenen Nachweisen wird dabei von mehreren Aussagen über diese Erweiterungen Gebrauch gemacht, nämlich

- der *Surjektivität* der Erweiterungen von surjektiven Umbenennungen,
- der *Injektivität* der Erweiterungen von injektiven Umbenennungen,
- der *Distributivität* der Erweiterungen über $\circ$ bzw. $\cup$, und
- die *umgekehrte Monotonie* von f^*, wie in Hilfsatz B.5.9 gezeigt.

Während die ersten drei Eigenschaften für endliche Alphabete leicht mittels Induktion nachweisbar sind, wird die vierte Aussage hier nachgewiesen.

Hilfsatz B.5.9 (Umgekehrte Monotonie von f^*) Für $a, b \in A^\omega$ und ein bijektives $f : A \to B$ gilt

$$f^*(a) \sqsubseteq f^*(b) \Rightarrow a \sqsubseteq b \tag{B.27}$$

△

Beweis B.5.13 (Hilfsatz B.5.9) Aussage B.27 folgt mit der Bijektivität von f und der damit, wie oben angesprochen, resultierenden Bijektivität von f^* sowie der Distributivität über $\circ$:

$$\begin{aligned}
& f^*(a) \sqsubseteq f^*(b) \\
\Longrightarrow\ & [\text{Definition } \sqsubseteq] \\
& \exists c. f^*(a) \circ c = f^*(b) \\
\Longrightarrow\ & [f^* \text{ surjektiv}] \\
& \exists c. f^*(a) \circ f^*(c) = f^*(b) \\
\Longrightarrow\ & [f^* \text{ distributiv}] \\
& \exists c. f^*(a \circ c) = f^*(b) \\
\Longrightarrow\ & [f^* \text{ injektiv}] \\
& \exists c. a \circ c = b \\
\Longrightarrow\ & [\text{Definition } \sqsubseteq] \\
& a \sqsubseteq b
\end{aligned}$$

□

Literaturverzeichnis

[Ace92] L. Aceto. *Action Refinement in Process Algebras.* Cambridge University Press, 1992.

[AL91] M. Abadi und L. Lamport. Composing Specifications. In J. de Bakker, W.-P. de Roever und G. Rozenberg, Hrsg., *Stepwise Refinement of Distributed Systems*, Seiten 1–41. Springer, 1991. LNCS 430.

[AWO85] J. A.Bergstra, J. W.Klop und E.-R. Olderog. Readies and failures in the algebra of communicationg processes. Technischer Bericht CS-R8523, University of Amsterdam, Centrum voor Wiskunde en Informatica, 1985.

[Bac93] R. Back. Refinement of Parallel and Reactive Systems. In M. Broy, Hrsg., *Program Design Calculi. Springer NATO ASI Series, Series F: Computer and System Sciences, Vol. 118*, 1993.

[Bag86] R. Bagrodia. A Distributed Algorithm to Implement the General Alternative Command of CSP. In *International Conference on Distributed Systems*, Seiten 422–428. IEEE, May 1986.

[BBB+90] T. Bemmerl, A. Bode, P. Braun, O. Hansen, P. Luksch und R. Wismüller. TOPSYS - Tools for Parallel Systems (User's Overview and User's Manual). TUM- 9047, Institut für Informatik der TU München, 1990.

[BBK87] J. C. Baeten, J. A. Bergstra und J. W. Klop. Ready-Trace Semantics for Concrete Process Algebra with the Priority Operator. *The Computer Journal*, 30(6):498 – 506, 1987.

[BBLT90] T. Bemmerl, A. Bode, T. Ludwig und S. Tritscher. MMK - Multiprocessor Multitasking Kernel (User's Guide and User's Reference Manual). TUM-I 9048, Institut für Informatik der TU München, 1990.

[BCHK94] A. Boudol, I. Castellani, M. Hennessy und A. Kiehn. A Theory of Processes with Localities. *Formal Aspects of Computing*, 6(2):165–200, 1994.

[BDD+92] M. Broy, F. Dederich, C. Dendorfer, M. Fuchs, T. Gritzner und R. Weber. The Design of Distributed Systems - An Introduction to FOCUS. TUM-I 9202, Technische Universität München, Januar 1992.

[BDD+93] M. Broy, F. Dederich, C. Dendorfer, M. Fuchs, T. Gritzner und R. Weber. The Design of Distributed Systems - An Introduction to FOCUS. TUM-I 9202-2,

Technische Universität München, Januar 1993. SFB-Bericht Nr.342/2-2/92 A.

[BDDW91] M. Broy, F. Dederichs, C. Dendorfer und R. Weber. Characterizing the Behaviour of Reactive Systems by Trace Sets. TUM-I 9102, Technische Universität München, Februar 1991.

[Ber93] G. Berry. Synchronous Languages for Reactive Systems: Styles, Semantics, Implementations. In *Symposium on Principles of Programming Languages*. ACM SIGPLAN-SIGACT, 1993.

[Bes93] E. Best, Hrsg. *CONCUR'93 - 4th international Conference on Concurrency Theory*. Springer Verlag, 1993. LCNS 715.

[BHR84] S. D. Brookes, C. A. R. Hoare und A. W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the Association of Computing Machinery*, 31(3):560–599, 1984.

[BM93] Ö. Babaoğlu und K. Marzullo. Consistent Global States. In S. Mullendar, Hrsg., *Distributed Systems*. ACM Press, 1993.

[Bro87a] M. Broy. Semantics of finite or infinite networks of communicating agents. *Distributed Computing*, 2:13–31, 1987.

[Bro87b] M. Broy. Specification and Top-Down Design of Distributed Systems. *Journal of Computer and System Science*, 34:236 – 265, 1987.

[Bro89] M. Broy. Towards a Design Methodology for Distributed Systems. In M. Broy, Hrsg., *Constructive Methods in Computer Science*, Seiten 311–364. Springer Verlag, 1989. LCNS.

[Bro90] M. Broy, Hrsg. *TC 2 Working Conference on Programming Concepts and Methods*, Sea Gallilee, Israel, April 2–5 1990. The International Federation for Information Processing. Preprint.

[Bro91] M. Broy. Formalization of Distributed, Concurrent, Reactive Systems. In E. J. Neuhold und M. Paul, Hrsg., *Formal Descriptions of Programming Concepts*. Springer, 1991.

[Bro92] M. Broy. Operational and Denotational Semantics with Explicit Concurrency. *Fundamenta Informaticae*, 16(3):201–230, März 1992.

[Bro94] M. Broy. Specification and Refinement of a Buffer of Lenght One, 1994. Working material of Marktoberdorf Summer School 1994.

[BS91] R. Back und K. Sere. Deriving an Occam Implementation of Action Systems. In C. Morgan und J. Woodcock, Hrsg., *3rd Refinement Workshop*, Workshop in Computing. Springer Verlag, 1991.

[BS94a] M. Broy und K. Stølen. Specification and Refinement of Finite Dataflow Networks - a Relational Approach. In *Proc. FTRTFT'94. 1994. p. 247-267*, 1994.

[BS94b] M. Broy und K. Stølen. Specification and Refinement of Finite Dataflow Networks - A Relational Approach. Technischer Bericht TUM-I9412, Technische Univerität München, 1994.

[CFR93] T. R. Colburn, J. H. Fetzer und T. L. Rankin, Hrsg. *Program Verification: Fundamental Issues in Computer Science*. Kluwer Academic Publishers, 1993.

[CM89] K. M. Chandy und J. Misra. *Parallel Program Design - A Foundation*. Addison-Wesley, 2. Auflage, Mai 1989.

[ČP93] M. Čubrić und P. Panangaden. Minimal Memory Schedules for Dataflow Networks. In *[Bes93]*, Seiten 368–383, 1993.

[dBKPR91] F. S. de Boer, J. N. Kok, C. Palamidessi und J. J. Rutten. The failure of failures in a paradigm for asynchronous communication. In J. Baeten und J. Groote, Hrsg., *Proceedings of CONCUR'91*, Bd. 527 der Reihe *Lecture Notes in Computer Science*, Seiten 111–126. Springer-Verlag, 1991.

[dBP91a] F. de Boer und C. Palamidessi. Embedding as a tool for Language Comparison: On the CSP hierarchy. Technischer Bericht 91/27, Eindhoven University of Technology, 1991.

[dBP91b] F. S. de Boer und C. Palamidessi. Embedding as a tool for Language Comparison: On the CSP hierarchy. In J. Baeten und J. Groote, Hrsg., *Proceedings of CONCUR'91*, Bd. 527 der Reihe *Lecture Notes in Computer Science*, Seiten 127–141. Springer-Verlag, 1991.

[Ded92] F. Dederichs. *Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen*. Dissertation, Technische Universität München, 1992.

[Den85] J. B. Dennis. Data Flow Computation. In M. Broy, Hrsg., *Data flow and control flow: concepts of distributed programming*, Bd. 14. Springer, 1985. NATO ASI Series F: Computer and Systems Science.

[Dij76] E. W. Dijkstra. *The Discipline of Programming*. Prentice Hall, 1976.

[Dil89] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed Independent Circuits*. ACM Distinguished Dissertations. The MIT Press, 1989.

[DS89] J. Davis und S. Schneider. An Introduction to Timed CSP. PRG- 75, PRG Programming Research Group Oxford, 1989.

[DS92a] J. Davies und S. Schneider. Using Timed CSP to Verify a Protocol over a Fair Medium. Technischer bericht, Programming Research Group Oxford, England, 1992.

[DS92b] J. Davies und S. Schneider. Using Timed CSP to Verify a Protocol over a Fair Medium. In W.R.Cleaveland, Hrsg., *CONCUR'92*, Seiten 355–369. Springer Verlag, 1992. LNCS 630.

[DW89] F. Dederichs und R. Weber. Safety and Liveness from a Methodological Point of View. MIP- 8918, Universität Passau, Juni 1989.

[DW92] C. Dendorfer und R. Weber. Development and Implementation of a Communication Protocol - An Exercise in FOCUS. TUM-I 9205, Technische Universität München, März 1992.

[Fle94] A. Fleischmann. *Distributed Systems: Software Design and Implementation.* Springer, 1994.

[Fra86] N. Francez. *Fairness.* Springer Berlin, 1986.

[Fuc94] M. Fuchs. *Technologieabhängigkeit von Spezifikationen digitaler Hardware.* Dissertation, Technische Universität München, 1994.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, J. Weicheng, R. Macheck und S. Vaidy. *PVM: Parallel Virtual machine: A Users Guide and Tutorial for Networked Parallel Computing.* Scentific and engineering computation series. The MIT Press, 1994.

[GS85] G.N.Buckely und A. Silberschatz. An Effecive Implementation of the Generalized Input-Output Construct of CSP. *Computing Reviews*, 24(6), 1985.

[GS95] R. Grosu und K. Stølen. A Denotational Model for Mobile Point-to-Point Dataflow Networks. TUM-I 9527, Technische Universität München, 1995.

[HJ95] M. G. Hinchey und S. A. Jarvis. *Concurrent Systems: Formal Development in CSP.* International Series in Software Engeneering. McGraw-Hill, 1995.

[Hoa78] C. A. R. Hoare. Communicating Sequnetial Processes. *Journal of the ACM*, 21(8):666–667, 1978.

[Hoa80] C. A. R. Hoare. A Model for Communicating Sequential Processes. In R. M. McKeag und A. M. MacNaghten, Hrsg., *On the Construction of Programs.* Cambridge University Press, 1980.

[Hoa83] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 26(1):100–106, 1983.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.

[Hoa93a] C. A. R. Hoare. Algebra and Models. In M. Broy, Hrsg., *Program Design Calculi. Springer NATO ASI Series, Series F: Computer and System Sciences, Vol. 118*, 1993.

[Hoa93b] C. A. R. Hoare. Mathematics of Programming. In *[CFR93]*. Kluwer Academic Publishers, 1993.

[Hoa94] C. A. Hoare. Mathematical Models for Computing Science, 1994. Working material of Marktoberdorf Summer School 1994.

[Inm84] Inmos Ltd., Englewood Cliffs, New Jersey. *Occam Programming Manual*, 1984.

[JG88] G. Jones und M. Goldsmith. *Programming in occam® 2.* International Series in Computer Science. Prentice Hall, 1988.

[JJH90] H. Jifeng, M. B. Josephs und C. A. R. Hoare. A Theory of Synchrony and Asynchrony. In *[Bro90]*, 1990.

[Jon87] B. Jonsson. *Compositional Verification of Distributed Systems.* Dissertation, Department of Computer Systems, Uppsala University, 1987.

[Jon90] B. Jonsson. A Hierachy of fully abstract Models of I/O-Automata. In B. Rovan, Hrsg., *Mathematical Foundation of Computer Science*, Seiten 347–354. Springer Verlag, 1990. LNCS 452.

[Jos92] M. B. Josephs. Receptive Process Theory. *Acta Informatica*, 29(1):17–31, 1992.

[Kah74] G. Kahn. The Semantics of a Simple Language for Parallel Programming. *Information Processing*, 74:471–475, 1974.

[KM68] K. Kuratowski und A. Mostowski. *Set Theory.* North-Holland Publishing Company, Amsterdam, 1968.

[KP94] K. N. Kumar und P. K. Pandya. On Computational Power of Operators in ICSP with Fairness. In *Proceedings of FST & TCS 14.* Springer, 1994. LNCS 880.

[Krö87] F. Kröger. *Temporal Logics of Programs.* Nummer 7 der Reihe EATCS Monograph. Springer Verlag, 1987.

[Lam89] L. Lamport. A Simple Approach to Specifying Concurrent Systems. *Communications of the ACM*, 32(1):32–47, 1989.

[LL94] L. Lamport und N. Lynch. Distributed Computing: Models and Methods. In J. van Leuwen, Hrsg., *Theoretical Aspects of Computer Science*, Bd. B. The MIT Press, 1994.

[Low93] G. Lowe. Probabilities and Priorities in CSP. PRG- 111, Oxford University Computing Laboratory, Programming Research Group, 1993. Ph.D. Thesis.

[LT89] N. Lynch und M. Tuttle. An Introduction to Input/Output Automata. *CWI Quarterly*, 2(3):219–246, 1989.

[Maz86] A. Mazurkiewicz. Trace Theory. In W. Brauer, W. Reisig und G. Rozenberg, Hrsg., *Petri Nets: Application and Relationship to Other Models of Conucrrency*, Seiten 279–304. Springer, 1986. LNCS 255.

[Mil83] R. Milner. *CCS - A Calculus for Communicating Systems*, Bd. 83 der Reihe *Lecture Notes in Computer Science.* Springer Verlag, 1983.

[MPW92a] R. Milner, J. Parrow und D. Walker. A Calculus of Mobile Processes, Part I. *Communication and Computation*, 1(100):1–40, 1992.

[MPW92b] R. Milner, J. Parrow und D. Walker. A Calculus of Mobile Processes, Part II. *Communication and Computation*, 1(100):41–77, 1992.

[Mul89] S. Mullendar, Hrsg. *Distributed Systems.* Frontier Series. ACM Press, 3.. Auflage, 1989.

[Old85] E.-R. Olderog. Semantics of Concurrent Processes: The Search for Structure and Abstraction - Part I. *EACTS Bulletins*, 28:73–97, 1985.

[Old86] E.-R. Olderog. Semantics of Concurrent Processes: The Search for Structure and Abstraction - Part II. *EACTS Bulletins*, ?:96–117, 1986.

[Old91] E.-R. Olderog. *Nets, Terms and Formulas*. Nummer 23 der Reihe Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1991.

[ORSS92] E.-R. Olderog, S. Rössig, J. Sander und M. Schenke. ProCoS at Oldenburg: The Interface between Specification Language and occam-like Programming Language. Technischer Bericht 3/92, Universität Oldenburg, jun 1992.

[PA85] K. Pingali und D. E. Arvind. Efficient Demand-Driven Evalutaion. *ACM Transations on Programming Languages and Systems*, 7(3), 1985.

[Pan93] P. K. Pandya. An Operational Justification of the Theory of Input-Buffered CSP. TCS- 93/5, Tata Institute of Fundamental Research, August 1993.

[Pau94a] L. C. Paulson. *Isabelle: A Generic Theorem Prover*, Bd. 828 der Reihe *LNCS*. Springer-Verlag, 1994.

[Pau94b] L. C. Paulson. *Isabelle's Object Logics*. Computer Labarotary, University of Cambridge, 1994.

[Pau94c] L. C. Paulson. *The Isabelle Reference Manual*. Computer Labarotary, University of Cambridge, 1994.

[Reg94] F. Regensburger. *HOLCF: Eine konservative Erweiterung von HOL um LCF*. Dissertation, Technische Universität München, 1994.

[Rei85] W. Reisig. *Petri Nets - An Introduction*. Nummer 4 der Reihe EATCS Monograph. Springer Verlag, 1985.

[Ros88] A. W. Roscoe. Two papers on CSP. PRG- 67, Oxford University Computing Laboratory, Programming Research Group, 1988.

[RR88] G. M. Reed und A. W. Roscoe. A Timed Model for Communicating Sequential Processes. In *Theoretical Computer Science*, Bd. 58, Seiten 249–261. North Holland, 1988.

[Rud95] J. Rudolph. Entwicklung einer Liftsteuerung - Fallstudie in FOCUS. Diplomarbeit, Technische Universität München, 1995.

[San96] R. Sandner. Unterstützung von Strukturverfeinerung in FOCUS durch *Isabelle* - Verifikation einer Fertigungszelle. Diplomarbeit, Technische Universität München, 1996.

[Sch93] M. D. Schroeder. A State-of-the-Art Distributed System: Computing with BOB. In S. Mullendar, Hrsg., *Distributed Systems*. ACM Press, 1993.

[Sha92] E. Shapiro. Embeddings Among Concurrent Programming Lanugages. In W.R.Cleaveland, Hrsg., *CONCUR'92*, Seiten 486–503. Springer Verlag, 1992.

[Spi88] P. P. Spies. Das aktuelle Schlagwort: Non-wait-send/Rendezvous. *Informatik-Spektrum*, 11(8):283–288, 1988.

[Stø94] K. Stølen. A Refinement Relation Supporting the Transitions from Unbounded to Bounded Communication Buffers. TUM-I 9435, Technische Universität München, 1994. SFB-Bericht Nr.342/17/94 A.

[vG96] R. J. H. van Glabbeck. Comparative concurrency semantics and refinement of actions. Technischer Bericht 109, Centrum voor Wiskunden en Informatica, 1996. CWI Tracts.

[Wal95] K. Waldschmidt, Hrsg. *Parallelrechner: Architekturen - Systeme - Werkzeuge.* Teubner, 1995.

[Web91] R. Weber. *Eine Methodik für die formale Anforderungsspezifikation verteilter Systeme.* Dissertation, Technische Universität München, Oktober 1991.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages - An Introduction.* Foundations of Computing Science. MIT Press, 1993.

[ZCdR92] J. Zwiers, J. Coenen und W.-P. de Roever. A note on compositional refinement. In C. B. Jones, R. G. Shaw und T. Denvir, Hrsg., 5^{th} *Refinement Workshop.* Springer, 1992.