TECHNISCHE UNIVERSITÄT DRESDEN

Adaptionen von Prozesskalkülen für eine Anwendung in der Systembiologie

Diplomarbeit

Florian Stenger

Betreuender Hochschullehrer: Prof. Dr. Franz Baader
Betreuerin: Dr. Monika Sturm

Eingereicht am 21. Juni 2007

Technische Universität Dresden
Institut für Theoretische Informatik
Lehrstuhl für Automatentheorie

21.12.2006

Aufgabenstellung Diplomarbeit

Adaptionen von Prozesskalkülen für eine Anwendung in der Systembiologie

Bearbeiter: Florian Stenger (Informatik)

Das Forschungsgebiet der Systembiologie verbindet die Biologie mit der Informatik, der Mathematik sowie den System- und Ingenieurwissenschaften und ist inhaltlich darauf gerichtet, Modellierungen von physiologischen Prozessen in Zellen, Zellverbänden und ganzen Organismen zu entwickeln. Dabei ist das Ziel, ein komplexes Verständnis der Lebensprozesse aufzubauen und zu vermitteln. Insbesondere die Theoretische Informatik kann zur Realisierung dieser Zielstellung einen wesentlichen Beitrag leisten. Die Diplomarbeit soll bisheriges Wissen und Erfahrungen in der Anwendung der Theoretischen Informatik in der Systembiologie repräsentieren und für ein konkretes Beispiel aus der Biologie einsetzen.

- Im Ergebnis der Literaturrecherche ist die Systembiologie in ihren Forschungsfeldern zu charakterisieren. Anforderungen an die Theoretische Informatik sind zu formulieren und auf Erfahrungen und Ergebnisse in der Anwendung formaler Beschreibungsmöglichkeiten (z.B. abstrakte Maschinen, Kalküle, Sprachen) zu verweisen.
- Prozesskalküle, die gezielt für eine Modellierung biologischer Prozesse entwickelt wurden, sind zusammenzustellen und in einem geeigneten Rahmen vorzustellen. Grenzen in ihrer Anwendbarkeit sind zu analysieren.
- Ein spezieller Prozesskalkül (entwickelt auf der Basis des Pi-Kalküls) ist für eine Anwendung in der Systembiologie zu entwickeln und darzustellen.
- Zur formalen Beschreibung eines ausgewählten biologischen Prozesses ist dieser Kalkül einzusetzen.
- Bewertung der Ergebnisse

Folgende Literatur wird empfohlen:

- A. Regev. *Computational systems biology: a calculus for biomolecular knowledge.* Ph.D. Thesis, Tel Aviv University, 2002.
- L. Cardelli. *Abstract Machines of Systems Biology.* Transactions on Computational Systems Biology. III, LNBI 3737, pp 145-168, Springer 2005.

- C. Priami. Transactions on Computational Systems Biology VII: v. 7 (Lecture Notes in Computer Science), Springer 2006.

Betreuerin: Dr. M. Sturm
Betreuender Hochschullehrer: Prof. F. Baader
Bearbeitungszeitraum: 21.12.2006 - 21.06.2007
Abschlußleistung: Diplomarbeit, Verteidigung mit Vortrag

Erklärung an Eides statt

Hiermit erkläre ich an Eides statt, diese Diplomarbeit selbstständig und ohne Verwendung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt zu haben.

Florian Stenger
Dresden, den 21. Juni 2007

Danksagung

Ich möchte mich an dieser Stelle herzlich bei allen bedanken, die mich während der Zeit meines Studiums unterstützt haben. An erster Stelle danke ich Frau Dr. Monika Sturm für die hervorragende Betreuung, das mir entgegengebrachte Vertrauen und die vielen interessanten Gespräche zur Thematik. Mein besonderer Dank gilt Herrn Sebastian Voigt für das akribische Korrekturlesen der Arbeit, sowie den regen Gedankenaustausch. Des Weiteren danke ich meiner Familie für die Unterstützung und die große Zuversicht, die sie meinem Studium entgegengebracht hat.

Inhaltsverzeichnis

1 Einführung **1**

2 Systembiologie und Theoretische Informatik **3**
2.1 Zelluläre Maschinen 4
2.1.1 Gen-Maschine 6
2.1.2 Protein-Maschine 7
2.1.3 Membran-Maschine 8
2.2 Theoretische Ansätze in der Systembiologie 10
2.2.1 Eigenschaften von Modellen 10
2.2.2 Einige Modellierungsansätze 13
2.3 Prozesskalküle 13
2.3.1 κ-Kalkül 14
2.3.2 Brane-Kalkül 16
2.3.3 Bio-Ambients 17

3 π-Kalkül **19**
3.1 Syntax 19
3.2 Reduktionssemantik 22
3.3 Transitionssemantik 26
3.4 Anwendungsbeispiel: Genexpression mit positivem Feedback 31

4 Stochastischer π-Kalkül **37**
4.1 Syntax und Semantik 37
4.2 Biochemischer stochastischer π-Kalkül 45
4.3 Anwendungsbeispiel: Genexpression in bSπ 49

5 β-Binder **53**
5.1 Syntax 54
5.2 Semantik 56
5.3 Stochastische β-Binder 59
5.4 Anwendungsbeispiel: Genexpression mit Kompartimenten 63
5.5 Mustermodellierungen biologischer Phänomene 67
5.5.1 Modifikation von Binder-Typen 67
5.5.2 Zellteilung 68
5.5.3 Endocytose und Exocytose 69
5.5.4 Energiehaushalt in Zellen 71

6 Vergleich der Ansätze **75**
6.1 Vergleich relevanter Eigenschaften 75
6.2 Betrachtungen zur Expressivität 78
6.2.1 Übersetzung von Sβ in bSπ 78
6.2.2 Übersetzung von join- und split-Operationen 84
6.2.3 Übersetzung von bSπ in Sβ 85

7 Bewertung und Ausblick **87**

Kapitel 1

Einführung

Warum profitiert die Biologie von der theoretischen Informatik und den Systemwissenschaften? Die Frage ist leicht zu beantworten: Weil Lebewesen als Ganzes und Komponenten von Lebewesen wie Organe, Zellverbände oder sogar bloß einzelne Zellen als Systeme im ingenieurwissenschaftlichen Sinne aufgefasst und durch Modelle und Konzepte der theoretischen Informatik beschrieben, analysiert und simuliert werden können. Um zu verstehen, warum diese Herangehensweise erst in jüngster Zeit auf das Interesse der Wissenschaftler stößt, empfiehlt es sich, einen Blick auf die aktuelle Forschung in der Zellbiologie zu werfen. Tatsächlich sind die Abläufe in einer einzigen lebenden Zelle so komplex, dass man bis vor wenigen Jahren nur davon träumen konnte, sie als Ganzes zu beschreiben. Auch heute sind die Systembiologen noch einen großen Schritt davon entfernt. Allerdings hat sich die Geschwindigkeit, mit der Informationen über die Struktur von Makromolekülen wie der DNA gewonnen werden können, aufgrund neuer Verfahren zur Automatisierung von Experimenten vervielfacht. Solche Makromoleküle, zu denen neben der DNA, die sich zu Genen zusammensetzt, auch Proteine und Zellmembranen gehören, bestimmen das Verhalten einer Zelle maßgeblich. Groß angelegte Projekte wie das *Human Genome Project*, dessen Ziel es ist, die Gene des Menschen in ihrer Gesamtheit zu erfassen oder *Proteomics*, einem Projekt, das auf die Charakterisierung aller in der Natur vorkommenden Proteine abzielt, lassen sich nun unter internationaler Zusammenarbeit effizient umsetzen.

Während sich die Bioinformatik mit der Speicherung und Darstellung der enormen Informationsmengen befasst, die bei solchen Experimenten anfallen, ist die Systembiologie inhaltlich darauf gerichtet, die Informationen zu nutzen, um die verschiedenen Komponenten wie Gene, Proteine und Membranen zunächst als einzelne abgeschlossene Systeme und dann insbesondere deren Interaktion und die Integration in komplexeren Systemen zu verstehen und mit formalen Mitteln nachzubilden. Es existieren bereits einige Ansätze, die jeweils ein einzelnes Gebiet, etwa die Synthese von Proteinen nach einem in DNA gespeicherten Bauplan oder den Transfer von Stoffen durch eine Zellmembran einigermaßen akurat beschreiben. Allerdings erweisen sich diese Ansätze oft als ungeeignet, um andere Aspekte des Zellverhaltens zu erfassen und sind darüber hinaus schlecht mit anderen Formalismen kombinierbar. Das Hauptanliegen der aktuellen Forschung auf dem Gebiet besteht daher darin, ein Konzept zu entwickeln, das universell genug ist, um verschiedenartige molekulare Vorgänge einheitlich zu modellieren und gleichzeitig intuitiv genug in seiner Handhabung ist, um von Biologen eingesetzt zu werden, ohne dass diese sich in den formalen Aspekten der dem Konzept zugrundeliegenden Theorie vertiefen müssten.

Ein Blick in die theoretische Informatik ist dabei alles andere als abwegig, denn hier existieren verschiedene abstrakte Maschinen, Kalküle und Sprachen zur Beschreibung von dynamischen und verteilten Systemen, die den in Lebewesen vorkommenden Systemen nicht unähnlich sind. Diese Arbeit beschäftigt sich im Kern mit einer Gruppe von Kalkülen, die auf dem Konzept der Interaktion ihrer autonomen Komponenten basieren und deren Anpassungsfähigkeit an verschiedene Themengebiete sie in den letzten Jahren zu einem vielversprechenden Vertreter in der Systembiologie gemacht hat. Diese unter dem Begriff der *Prozesskalküle* zusammengefassten Sprachen wurden ursprünglich für die Modellierung und Analyse von verteilten Rechnersystemen und Telekommunikationsnetzen entwickelt und scheinen sich nicht unmittelbar für eine Anwendung auf biologische Systeme zu eignen. Daher gibt es eine ganze Reihe von biologisch motivierten Erweiterungen und Spezialisierungen dieser Sprachen, die in den folgenden Kapiteln vorgestellt und diskutiert werden.

In Kapitel 2 wird zunächst in knapper Form eine Übersicht über das Gebiet der Systembiologie gegeben. Es wird die Sichtweise der theoretischen Informatik auf das Gebiet sowie die Anforderungen an formale Modellierungsansätze herausgearbeitet. Des Weiteren werden einige, in dieser Arbeit nicht schwerpunktmäßig behandelte Konzepte kurz vorgestellt. Das Kapitel 3 stellt einen der bedeutendsten und wohluntersuchtesten Prozesskalküle, den *π-Kalkül*, ausführlich in Syntax und Semantik vor und enthält darüber hinaus die Beispielmodellierung einer Genexpression, die einen relevanten biologischen Vorgang darstellt. Eine Erweiterung des klassischen π-Kalküls um quantitative Aspekte wird in Kapitel 4 mit dem *stochastischen π-Kalkül* angegeben. Neben der Syntax und Semantik zweier Varianten dieses Kalküls enthält das Kapitel außerdem eine stochastische Erweiterung des bereits im 3. Kapitel vorgestellten Genexpressionsbeispiels. In Kapitel 5 wird mit dem Kalkül der *β-Binder* eine Erweiterung des klassischen π-Kalküls um Kompartimente angegeben, die eine explizite Abgrenzung von biologischen Entitäten untereinander erlaubt. In einer stochastischen Variante des Kalküls werden die Vorzüge der Erweiterungen um quantitative Aspekte sowie um Kompartimente vereint. Anschließend wird erneut das Genexpressionsbeispiel betrachtet. Das Kapitel schließt mit einigen Mustermodellierungen relevanter biologischer Phänomene. Neben diesen Modellierungen besteht der Beitrag dieser Arbeit zur aktuellen Forschung in einem Vergleich der verschiedenen formalen Konzepte zur systembiologischen Modellierung in Kapitel 6, der sowohl informell anhand einer Menge von Modelleigenschaften erfolgt, wie auch formal in einer Analyse der Expressivität der vorgestellten Prozesskalküle. Kapitel 7 vervollständigt die Arbeit mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf zukünftige Forschung in dem Gebiet.

Kapitel 2

Systembiologie und Theoretische Informatik

Obwohl der Begriff Systembiologie erst seit wenigen Jahren in Verwendung ist, sind die ersten Arbeiten in dieser Disziplin bereits ein halbes Jahrhundert alt. 1952 entwickelten die britischen Nobelpreisträger Alan Lloyd Hodgkin und Andrew Fielding Huxley ein mathematisches Modell einer Nervenzelle und legten damit den Grundstein für die Simulation biologischer Prozesse. Acht Jahre später gab Denis Noble ein mathematisches Modell eines schlagenden Herzes an, dessen Detailliertheit ausreichend war, um damit Medikamente und Defibrillationsgeräte am Computer zu testen.

Seit diesen frühen Schritten ist die Bedeutung der Computersimulation in der Biologie stark gewachsen. Durch Laborversuche werden die Daten gewonnen, mit denen formale Modelle spezifiziert werden können, die schließlich eine Simulation des untersuchten Systems am Computer erlauben. Die Simulation wiederum beschleunigt die Datenbeschaffung, da sie Vorhersagen liefert, die gezielt experimentell validiert werden können. Vor der Entwicklung konkreter Modelle für biologische Phänomene ließen sich lediglich breit gefächerte Experimente durchführen, die darauf abzielten, relevante Eigenschaften von Molekülen zu identifizieren. Die unvorstellbare Anzahl an verschiedenen Genen, Proteinen und anderen Molekülen, die an den Lebensprozessen beteiligt sind, machen eine vollständige experimentelle Erschöpfung insbesondere der Interaktionen zwischen verschiedenen Molekülen jedoch unmöglich. Erst die computergenerierten Vorhersagen über mögliche Interaktionspartner innerhalb einer Zelle geben den Folgeexperimenten eine Richtung.

Dieses Kapitel soll dem Leser einen Überblick über die aktuellen Zielstellungen in der Systembiologie und die damit verbundenen Anforderungen an die theoretische Informatik geben. Stellenweise orientiert sich das Kapitel an dem Überblicksartikel [Car1_05] von Luca Cardelli. Des Weiteren sei dem interessierten Leser der Überblicksartikel [ReSh_02] empfohlen.

Das Hauptaugenmerk der Forschung in der Systembiologie liegt derzeit auf der einzelnen Zelle. Die umfassende Formalisierung des Zusammenspiels der Komponenten einer Zelle sind der erste Schritt, um einem Verständnis der Interaktionen zwischen verschiedenen Zellen näherzurücken. Schließlich könnte man sich an eine Formalisierung von Gewebe und ganzen Organen heranwagen. Der nächste Schritt wäre die Erstellung eines Gesamtmodells eines lebenden Organismus und als Endziel

ließe sich die Modellierung von Populationen verschiedenster Lebewesen proklamieren. Von diesem Ziel ist die Forschung weit entfernt. Selbst die exakte Beschreibung eines einzelnen Gens oder Proteins erweist sich nach wie vor als schwierig.

Zum besseren Verständnis des Subjektes der in dieser Arbeit vorgestellten Modellierungen soll dem Leser an dieser Stelle ein knapper Überblick über den allgemeinen Aufbau von Zellen vermittelt werden. Für weiterführende Informationen sei beispielsweise auf [PlHe_06] verwiesen. Mehrzellige Lebewesen setzen sich aus *Eukaryotenzellen* zusammen, die in verschiedene separate Komponenten - sogenannte Kompartimente - untergliedert sind. Daneben existieren auch noch *Prokaryotenzellen*, die stets einzellige Lebewesen wie Bakterien darstellen und deren Zellraum nicht weiter unterteilt ist. Beide Zelltypen stellen von Membranen umgebene Flüssigkeitsbläschen dar, in denen verschiedene Moleküle, wie Proteine oder die DNA, aus der sich das *Genom* zusammensetzt, vorliegen. Während bei Prokaryoten das Genom direkt in der als *Cytosol* bezeichneten Zellflüssigkeit liegt, enthalten Eukaryoten einen durch eine separate Membran umschlossenen *Zellkern*, der das Genom von anderen Zellkomponenten abkapselt. Weitere Komponenten der Eukaryotenzelle sind die *Mitochondrien*, die Nährstoffe wie Glukose abbauen und damit für den Energiehaushalt der Zelle verantwortlich sind, das glatte und das rauhe *Endoplasmatische Retikulum*, an denen die Synthese von Lipiden bzw. Proteinen erfolgt und der *Golgi-Apparat*, der verschiedene Stoffe mit Membranen umhüllt, um sie innerhalb der Zelle oder sogar zwischen benachbarten Zellen zu transportieren. Unter diesen membranumhüllten Stoffen sind zwei Arten besonders hervorzuheben: *Sekretvesikel* transportieren Abfallstoffe zur Zellmembran, um sie nach außen abzugeben. *Lysosomen* hingegen zerlegen ihren Inhalt in kleinere, für die Zelle nützliche Bausteine und realisieren dadurch einen intrazellulären Verdauungsprozess. Neben diesen funktionellen Kompartimenten der Eukaryotenzelle existieren auch noch strukturgebende Elemente, für die sich der Oberbegriff *Cytoskelett* etabliert hat. Diese stabförmigen, festen Makromoleküle unterteilen sich in *Mikrofilamente*, *Mikrotubuli* und *Intermediär-Filamente*. Die Gesamtheit der Kompartimente einer Zelle wird unter dem Begriff der *Organellen* zusammengefasst. Zusammen mit dem Cytosol bilden sie das *Cytoplasma*. Pflanzenzellen weisen im Vergleich mit Tierzellen zwei weitere Elemente auf. Zum einen die den Mitochondrien ähnlichen *Chloroplasten*, die Glukose mittels Photosynthese herstellen und zum anderen die Zellwand, die als feste Struktur um die Zellmembran gehüllt ist und dem Organismus eine starre Gestalt verleiht. Die Eukaryotenzelle unterscheidet sich neben der Existenz eines Zellkerns noch in einem weiteren Punkt von der Prokaryotenzelle. Sie tritt in einem Lebewesen in stark unterschiedlicher Gestaltung auf. Diese Differenzierungsmöglichkeit ist der Schlüssel zur Entwicklung von Muskelgewebe, Nervenzellen oder Blutkörperchen, die jeweils Eukaryotenzellen mit unterschiedlichen Zusammensetzungen von Organellen und verschiedenartiger äußerer Form darstellen. Abbildung 2.1 zeigt eine typische Eukaryotenzelle als Schema.

2.1 Zelluläre Maschinen

Die molekularen Bausteine einer lebenden Zelle lassen sich grob in drei Klassen von Makromolekülen unterteilen, die jeweils spezifische Aufgaben besitzen, allerdings bei deren Bewältigung in hohem Maße von den Einflüssen der anderen Molekülklassen abhängig sind. Die zuvor beschriebenen Organellen bestehen fast ausschließlich

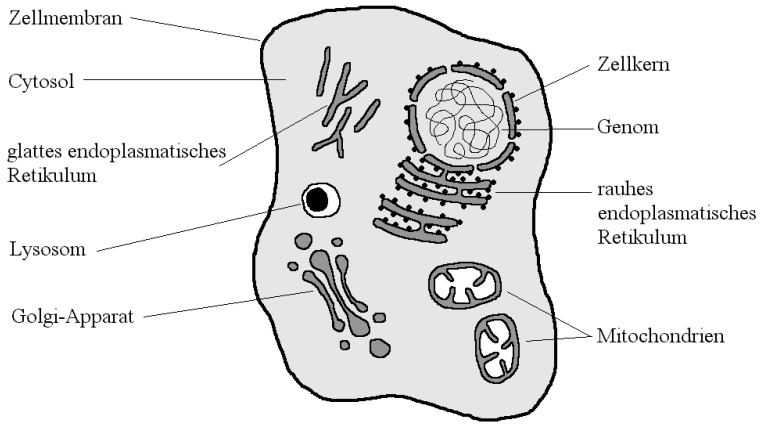


Abbildung 2.1: schematische Darstellung einer Eukaryotenzelle

aus diesen Molekülen sowie aus *Polysacchariden* (Zuckerstoffen), die überwiegend als Energiespeicher Verwendung finden und in den folgenden Betrachtungen eine untergeordnete Rolle spielen.

- Die *DNA* und *RNA* dienen als Informationsspeicher. Sie bestehen aus *Nucleinsäuren* und bilden das Genom, welches einen digitalen Bauplan des Organismus darstellt. Die einzelnen Gene des Genoms instruieren die Produktion von Proteinen und steuern deren Einbettung in die Oberfläche einer Membran.

- *Proteine*, die sich aus Ketten von *Aminosäuren* zusammensetzen, agieren als Nachrichtenüberbringer zwischen Genen und steuern, falls sie in eine Zellmembran eingebettet sind, den Transport von Stoffen in das Zellinnere oder aus der Zelle heraus. Sie können darüber hinaus auch Signale an benachbarte Zellen oder Moleküle übermitteln. Eine weitere wichtige Funktion von Proteinen ist das Katalysieren von chemischen Reaktionen. Proteine, die letztgenannte Aufgabe übernehmen, werden auch *Enzyme* genannt. In Form von größeren Komplexen (d.h. als Bestandteile des Cytoskeletts) können Proteine auch strukturgebende Funktionen übernehmen.

- *Membranen* setzen sich aus Schichten von *Phospholipiden* zusammen und grenzen sowohl ganze Zellen als auch Komponenten von Zellen gegeneinander ab. Sie bilden außerdem sogenannte *Transport-Vesikel*, um Stoffe innerhalb der Zelle oder zwischen verschiedenen Zellen zu transportieren.

Alle drei Klassen können als informationsverarbeitende Agenten oder Maschinen im Sinne der Informatik aufgefasst werden, die sich ähnlich wie Automaten in der klassischen Automatentheorie durch eine Menge von Zuständen und Regeln beschreiben lassen. Die Anwendung einer Regel verursacht eine Transition zwischen Zuständen, das heißt den Übergang vom aktuellen Zustand in einen Folgezustand. Die Beschreibung der Aktivitäten von intrazellulären Makromolekülen durch automatenähnliche Maschinen stellt eine notwendige Abstraktion dar. Die Einbeziehung aller Details würde die Betrachtung auf quantenmechanischer Ebene erzwingen und damit die Komplexität des Modells auf ein Niveau bringen, bei dem keine sinnvolle Analyse

mehr möglich wäre. Die vollzogene Abstraktion ist allerdings konsistent mit sämtlichen experimentellen Beobachtungen, da tatsächlich viele molekularbiologische Prozesse diskret ablaufen. D.h. die Schritte, die den Übergang eines Zustandes in einen anderen ausmachen, müssen nicht notwendigerweise in die Betrachtung mit einbezogen werden. Beispielsweise kann eine Phosphatgruppe an einer speziellen Bindungsstelle eines Proteins andocken und damit die Bindungsstelle belegen. Es existieren also exakt zwei Zustände für jede solche Bindungsstelle des Proteins (belegt oder frei). Die Wahrscheinlichkeit, dass zwei Phosphatgruppen um eine Bindungsstelle konkurrieren und dabei über längere Zeit einen Zwischenzustand erzeugen, der nicht als einer der beiden genannten Zustände interpretiert werden kann, ist vernachlässigbar.

Wir wollen nun die drei Makromolekülklassen etwas genauer betrachten und dabei bereits die wesentlichen Eigenschaften, die für eine Modellierung durch Konzepte der theoretischen Informatik relevant sind, herausarbeiten. Wir bezeichnen dabei die Gesamtheit der Moleküle einer bestimmten Klasse mit ihren jeweiligen Zuständen und Transitionen als eine *abstrakte Maschine*.

2.1.1 Gen-Maschine

Das Genom eines Lebewesens ist mit einem großen Softwareprogramm vergleichbar, das die Vorgänge in einer Zelle koordiniert und reguliert. Es besteht aus einem langen DNA-Doppelstrang, der wiederum aus Paaren von Nucleotiden besteht. Dabei werden die Informationen (das Programm) durch vier verschiedene Basen kodiert, von denen jeweils eine in jedem Nucleotid enthalten ist. Der lange zu einer Helix gewundene DNA-Doppelstrang des Genoms ist in Abschnitte unterteilt, die Gene genannt werden und jeweils den Bauplan für ein Protein darstellen. Genauer gesagt ist ein Gen nochmals in zwei Regionen unterteilt, von denen die erste als *Input-* oder *Regulierungsregion* bezeichnet wird und eine Reihe von Bindungsstellen für spezielle Proteine, die *Transkriptionsfaktoren*, bereitstellt. Die zweite Region wird *Output-* oder *Kodierungsregion* genannt und enthält die Abfolge der Aminosäuren des Proteins, das von dem Gen produziert wird. Dabei werden die zwanzig verschiedenen Aminosäuren stets durch ein eindeutig festgelegtes Tripel von Nucleotiden charakterisiert. Durch das Andocken von Transkriptionsfaktoren an die Inputregion des Gens kann die Proteinsynthese entweder hemmend oder verstärkend beeinflusst werden. Die Synthese des kodierten Proteins wird nicht direkt am Gen vollzogen. Vielmehr findet zunächst eine *Transkription* (ein Abschreiben oder Kopieren) der DNA in einen ebenfalls aus Nucleotiden bestehenden RNA-Strang statt, der den Zellkern verlässt und erst anschließend in ein Protein übersetzt wird. Bei diesem letzten als *Translation* bezeichneten Schritt bleibt die RNA zunächst erhalten, so dass durchaus mehrere Proteine aus einem einzigen RNA-Strang generiert werden können. Der Translationsprozess wird erst durch den Zerfall des im Vergleich zur DNA wesentlich instabileren RNA-Moleküls beendet. Der gesamte Vorgang der Proteinsynthese wird auch *Genexpression* genannt.

Da die DNA des Gens selbst stets unverändert bleibt, wird der Zustand der *Gen-Maschine* einer Zelle (in der Literatur oft auch *Gen-Regulierungsnetzwerk* genannt) durch die Konzentration der Transkriptionsfaktoren und der RNA-Stränge in der Zelle bestimmt. Die genaue Wirkung, die die Protein-Gen-Interaktionen in der Inputregion eines Gens bei der Transkription verursachen, ist zur Zeit noch nicht völlig

geklärt. Aktuelle Experimente deuten darauf hin, dass Gene in der Lage sind, relativ komplexe Berechnungen auf ihrer Eingabe, also der An- bzw. Abwesenheit von Transkriptionsfaktoren, durchzuführen. Die Beschreibung von Zustandsübergängen erfolgt daher nicht durch exakte Formeln, sondern durch eine (informelle) grafische Notation, in der ein Gen durch einen horizontalen Balken gekennzeichnet wird, aus dem ein (evtl. sich verzweigender) Pfeil hervorgeht, der auf andere Gene verweisen kann. Ein spitzes Ende des Pfeils ist als verstärkende Wirkung des von dem Gen produzierten Proteins auf das Zielgen zu interpretieren, während ein stumpfes Pfeilende eine hemmende Wirkung signalisiert. Abbildung 2.2 zeigt ein Beispiel mit drei Genen. Jedes Gen hat in der Regel eine Ausgabe, kann aber mehrere Eingaben haben. Auf diese Weise lassen sich die Interaktionen zwischen den Genen wie ein Hardware-Schaltplan darstellen. Allerdings gibt es hierbei einen markanten Unterschied: Während bei elektronischer Hardware gleichartige „Gates“ nicht auf die gleiche Weise verbunden werden müssen, ist bei einem Gen durch die Sequenz der Nucleotide genau festgelegt, mit welchen anderen Genen es in Verbindung steht.

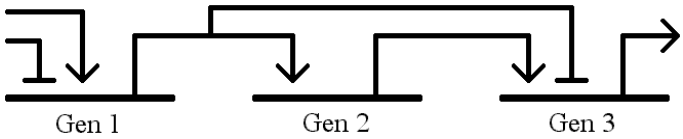


Abbildung 2.2: Schaltplan-Notation für drei Gene. Gen 1 produziert ein Protein, das die Produktion von Gen 2 verstärkt und die von Gen 3 hemmt. Gen 2 verstärkt mit seinem Protein die Produktion von Gen 3.

2.1.2 Protein-Maschine

Proteine setzen sich aus einer oder mehreren ineinander verschlungenen Ketten von Aminosäuren zusammen. Im Gegensatz zu DNA-Strängen ist bei einem Protein jedoch nicht die genaue Sequenz der Bausteine entscheidend für dessen Verhalten, sondern die dreidimensionale Struktur und zwar insbesondere die für andere Moleküle zugängliche Oberfläche. Die spezielle Faltung der Aminosäureketten ermöglicht anderen Molekülen (z.B. auch anderen Proteinen) das Andocken an bestimmten Bindungsstellen. Nur die Bindungsstellen an der Oberfläche stehen für Interaktionen mit anderen Molekülen zur Verfügung. Durch das Andocken eines Moleküls kann sich die Faltung der Aminosäureketten verändern, wodurch bisher verfügbare Bindungsstellen im Inneren des Proteins verschwinden, oder vorher unzugängliche Bindungsstellen an die Oberfläche kommen.

Manche Bindungsstellen erlauben ausschließlich das Andocken von Phosphatgruppen und funktionieren daher wie binäre Schalter mit den Zuständen frei oder belegt. In der Regel wird dieser als *Phosphorilierung* bzw. *Dephosphorilierung* bezeichnete Prozess von einem Enzym katalysiert. Dabei dockt zunächst das Enzym (welches ebenfalls ein Protein ist) an das zu phosphorilierende Protein an, dieses bindet nun eine Phosphatgruppe aus der Umgebung und anschließend trennt sich das Enzym wieder ab. Enzyme können auf diese Weise auch andere Moleküle zur Reaktion bringen, bleiben dabei jedoch selbst stets unverändert. Der Vorgang des Bindens zweier Proteine, wie bei der beschriebenen Katalysation, wird als *Komplexbildung*

bezeichnet. Eine Bindungsstelle, die derartige Interaktionen ermöglicht, heißt im Folgenden *Interaktionsstelle*. Wie bereits im vorangehenden Abschnitt erläutert, bilden einige als Transkriptionsfaktoren bezeichnete Proteine Komplexe mit Genen, um die Produktion anderer Proteine zu manipulieren. Auch Proteine unterliegen, ähnlich wie RNA-Stränge, stochastischen Zerfallsprozessen. Damit die Konzentration eines Proteins in einer Zelle stabil bleiben kann, müssen daher ständig neue Proteine produziert werden.

Vereinfacht kann man die *Protein-Maschine* einer Zelle (in der Literatur auch häufig als *Biochemisches Netzwerk* bezeichnet) wie folgt formalisieren: Jedes Protein besteht aus einer Menge von Schaltern und Interaktionsstellen, die jeweils verfügbar (an der Oberfläche) oder unzugänglich (im Inneren) sind. Jeder Schalter ist darüber hinaus entweder frei (ohne Phosphatgruppe) oder besetzt und jede Interaktionsstelle ist entweder frei oder in eine Komplexbildung involviert. Damit ist der Zustand der Protein-Maschine charakterisiert. Fünf Operationen modellieren das dynamische Verhalten des Proteins und damit die Zustandsübergänge: Durch (De-)Phosphorilierung wird ein verfügbarer Schalter auf den Zustand besetzt bzw. frei gesetzt und durch Komplexbildung (-trennung) werden zwei Proteine zu einem Komplex verbunden oder ein Komplex in Teilkomplexe zerlegt. Diese vier Operationen haben eine genau definierte Veränderung der Verfügbarkeit der Schalter und Interaktionsstellen der involvierten Proteine zur Folge. Die fünfte Operation ist der Zerfall eines Proteins in seine Bestandteile. Für die Interaktionen zwischen Proteinen hat sich ebenfalls eine grafische Notation etabliert, die in Abbildung 2.3 an einem Beispiel gezeigt wird. Ein Protein wird als ein Rechteck mit abgerundeten Ecken dargestellt, auf dessen Oberfläche sich Kreise für die Schalter und Quadrate für die Interaktionsstellen befinden. Gestrichelt gezeichnete Bindungsstellen sind unzugänglich und grau ausgefüllte Bindungsstellen sind besetzt (im Fall von Schaltern) oder in eine Komplexbildung involviert (bei Interaktionsstellen).

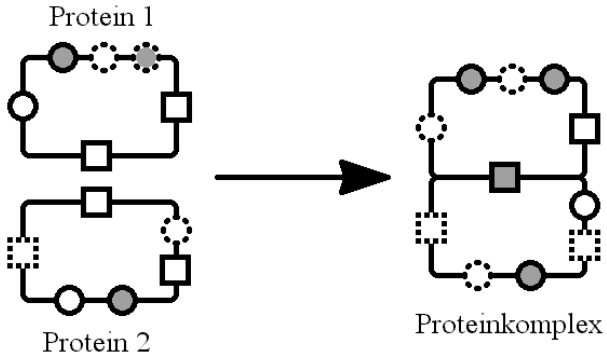


Abbildung 2.3: Zwei Proteine bilden einen Komplex. Die Verfügbarkeit der Bindungsstellen beider Proteine wird dabei verändert.

2.1.3 Membran-Maschine

Zellen werden von Membranen, die sich aus zwei Schichten von Phospholipiden zusammensetzen, gegen ihre Umgebung abgegrenzt. Darüber hinaus werden einzelne Bereiche innerhalb einer Zelle, wie beispielsweise der Zellkern, ebenfalls durch Mem-

branen vom übrigen Cytoplasma separiert. Zellmembranen sind zwar undurchlässig für die meisten Substanzen, verfügen aber über komplizierte Mechanismen, um Nährstoffe oder andere für die Zelle relevante Materialien wie Proteine kontrolliert in das Zellinnere bzw. aus der Zelle heraus zu transportieren. Damit übernehmen auch Membranen eine bedeutende Rolle bei der Realisierung der informationsverarbeitenden Prozesse einer Zelle. Um verschiedene Substanzen erkennen zu können, sind in eine Membran eine Vielzahl von Proteinen mit spezifischen Bindungsstellen eingebettet, die sowohl nach innen wie auch nach außen gerichtet sein können. Kommt eine von der Zelle benötigte Substanz in die Nähe der Zellmembran, so kann sie an die dafür vorgesehene Bindungsstelle eines Oberflächenproteins andocken und je nach Bedarf in das Innere transportiert werden. Der Transport erfolgt jedoch nicht durch das Öffnen der Zellmembran, was schließlich den unkontrollierten Austritt oder Eintritt von anderen Substanzen zur Folge haben könnte, sondern über ein „Verschlingen“ des zu transportierenden Materials. Die Membran umschließt das Transportgut vollständig, so dass dieses von einer Membranblase eingehüllt ist. Nun wird die Membranblase im Inneren der Zelle von der restlichen Zellmembran abgelöst. Als letztes wird die Membranblase aufgelöst, wodurch das Transportgut zum Cytoplasma der Zelle hinzugefügt wird.

Neben dieser Operation des „Verschlingens“, die als *Endocytose* bezeichnet wird, existiert auch eine inverse Operation die *Exocytose* genannt wird und das „Ausspeien“ einer in eine Membranblase gehüllten Substanz in die Umgebung der Zelle realisiert. Dabei verschmilzt die Membranblase beim Austritt des Transportgutes aus der Zelle mit der Zellmembran. Zellen können darüber hinaus mit anderen Zellen verschmelzen (*Mate*) oder sich unter Aufteilung ihrer Komponenten in mehrere Zellen zerteilen (*Mito*). Abbildung 2.4 zeigt die vier Operationen in einer grafischen Darstellung. Der Zustand der *Membran-Maschine* einer Zelle läßt sich nicht so trivial beschreiben, wie dies bei der Gen- und Protein-Maschine der Fall war. Sowohl die verschachtelte Struktur der Zellmembran und der in ihr eingeschlossenen membranumhüllten Komponenten, wie auch die Menge der Oberflächenproteine und deren Bindungsstellen müssen in einer Zustandsbeschreibung berücksichtigt werden. Verkomplizierend kommt noch hinzu, dass Oberflächenproteine manche Bindungsstellen nur dem Membraninneren bzw. nur dem Äußeren zur Verfügung stellen, was zur Folge hat, dass jeder Bindungsstelle eine Orientierung zugeordnet werden muss.

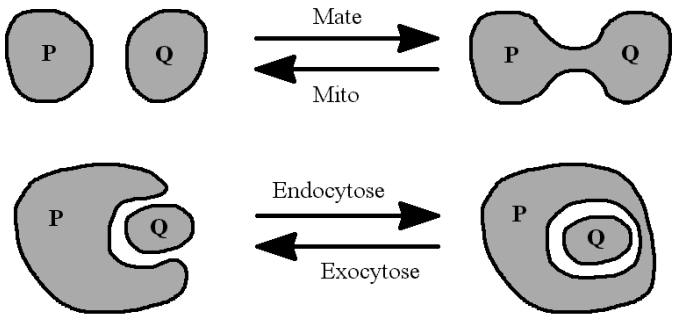


Abbildung 2.4: Die Operationen der Membran-Maschine

2.2 Theoretische Ansätze in der Systembiologie

Für die drei im letzten Abschnitt vorgestellten zellulären Maschinen existieren jeweils Modelle, die im Wesentlichen auf einer Beschreibung der Molekülanzahl und -dichte mittels Differentialgleichungen basieren. Damit lassen sich bereits stochastische Simulationen durchführen und so Vorhersagen über die Entwicklung eines Systems mit zuvor definiertem Anfangszustand gewinnen. Diese Ansätze beschreiben jedoch nicht alle Komponenten eines Systems als einzelne Teilsysteme mit unabhängigem Verhalten, sondern approximieren das Gesamtverhalten des Systems innerhalb eines gegebenen Zeitrahmens. Stellt man die Anzahl eines bestimmten Molekültyps als Funktion über die Zeit dar, so ergeben sich in solchen Simulationen meist recht glatte Kurven. In realen Systemen entstehen aufgrund der diskreten Übergänge zwischen Systemzuständen jedoch oft starke Fluktuationen der Molekülanzahl, die eine nicht zu vernachlässigende Bedeutung für einzelne Komponenten des Systems haben können. Die Detailliertheit von Modellen auf der Basis von Differentialgleichungen reicht also lediglich für einen groben Überblick über die Entwicklung des gesamten Systems. Außerdem ist ein solcher Ansatz nicht für die gleichzeitige Modellierung aller drei Makromolekülklassen geeignet, da keine ausreichende Skalierbarkeit sowohl der räumlichen wie auch der zeitlichen Dimension der Operationen der abstrakten Maschinen vorliegt. Beispielsweise laufen Proteininteraktionen in Sekundenbruchteilen ab, während für eine Interaktion zwischen verschiedenen Genen einige Minuten vergehen können. Ein weiteres Beispiel ist der enorme Größenunterschied zwischen einem Protein und einer Zellmembran. Einer Verwendung von Differentialgleichungs-Modellen für die Simulation von sehr komplexen Systemen stehen auch die Schwierigkeiten bei der Erweiterung der Modelle, die oft eine komplette Überarbeitung aller Formeln bedingen, entgegen.

Um eine Zelle einheitlich in einem Modell beschreiben zu können, ist ein Ansatz vonnöten, der die Erweiterbarkeit und Skalierbarkeit in den Vordergrund rückt. Er sollte die Interaktionen zwischen den Entitäten möglichst unabhängig von der tatsächlichen Systemgröße beschreiben und sich dennoch für die Umsetzung in Software-Simulatoren eignen. Wir stellen im Folgenden einige in der Systembiologie verwendete theoretische Ansätze vor, die genau dieses Ziel der ganzheitlichen Erfassung des Verhaltens kompletter Zellen verfolgen. Zuvor sollen jedoch einige Eigenschaften diskutiert werden, die eine gewisse Vergleichbarkeit von Modellen ermöglichen und außerdem die Anforderungen der gegebenen Aufgabe an die theoretische Informatik spezifizieren.

2.2.1 Eigenschaften von Modellen

Die Anzahl an wissenschaftlichen Gruppierungen, die sich mit der Disziplin Systembiologie auseinandersetzen, nimmt stetig zu. Es ist nicht verwunderlich, dass dabei eine Vielzahl an Theorien und Formalismen mit teilweise sehr ähnlichen, teilweise allerdings auch komplementären Konzepten entstehen. Viele der Ansätze sind für das in dieser Arbeit proklamierte Ziel des Zusammenfassens der drei abstrakten zellulären Maschinen völlig ungeeignet und sind auch nicht in diesem Bestreben konstruiert worden. Einige Ansätze dienen lediglich einer möglichst unmissverständlichen Kommunikation zwischen Biologen. Dazu zählen hauptsächlich grafische Notationen ohne formalen Hintergrund. Andere Ansätze, die speziell für die Simulation

eines bestimmten Phänomens entwickelt wurden, sind zwar sehr formal und detailliert ausgearbeitet, besitzen dafür aber keine Anpassungsfähigkeit. Wir wollen in diesem Abschnitt wesentliche Eigenschaften charakterisieren, die für einen formalen Ansatz gemäß der Intention der ganzheitlichen Beschreibung von Zellen relevant sind. Das soll jedoch nicht heißen, dass ein Ansatz ohne diese Eigenschaften keine Daseinsberechtigung hätte. Lediglich wird damit die Eignung für die formale und einheitliche Beschreibung von Gen-, Protein- und Membran-Maschine einer Zelle in Frage gestellt.

- **formal:** Auch wenn das Verständnis der zellulären Vorgänge und deren qualitative Beschreibung der wichtigste Aspekt der Modellbildung sein mögen, so ist der Blick in der Systembiologie von Anfang an stets auf eine Simulation gerichtet, die Vorhersagen ermöglicht. Dazu ist es unerlässlich, eine exakte, formale Notation zu verwenden. Dies ist auch dann sinnvoll, wenn bestimmte biologische Phänomene noch nicht vollständig erklärbar sind. Die von einer Simulation generierten Vorhersagen motivieren eine experimentelle Validierung, die notfalls die Korrektur oder eine größere Detailtiefe des formalen Modells ermöglichen. Die begleitende (formale) Modellkonstruktion während der Laborforschung kann also durchaus förderlich für beide Seiten sein.

- **verständlich:** Das Modell dient nicht bloß der Konstruktion von Simulationsprogrammen, sondern soll auch die Vermittlung von Forschungsergebnissen zwischen verschiedenen Wissenschaftlergruppen erleichtern. Dabei ist eine leicht verständliche Notation von Vorteil. Eine Vielzahl benutzter Symbole ist der Verständlichkeit ebenso abträglich wie eine große Anzahl verschiedener Operationen.

- **intuitiv benutzbar:** Das Kodieren von Forschungsergebnissen in der formalen Notation eines abstrakten Modells geschieht von Menschenhand. Daher ist es sinnvoll, wenn die Notation die Inhalte nicht bloß verständlich wiedergibt, sondern auch in intuitiver Weise aufschreiben lässt. Sicherlich könnten die meisten biologischen Prozesse in abstrakter Form direkt in einer imperativen Programmiersprache implementiert werden. Der nötige Aufwand zur Anpassung von gewöhnlichen Datenstrukturen in solchen Sprachen an die speziellen Gegebenheiten in der Molekularbiologie würde dieses Vorhaben allerdings schnell zu einem sehr großen und unübersichtlichen Projekt werden lassen.

- **skalierbar:** Wie bereits erwähnt, klaffen die räumlichen Dimensionen verschiedener Moleküle ebenso wie die zeitlichen Dimensionen ihrer Interaktionen stark auseinander. Ein gutes Modell sollte diesen Aspekt berücksichtigen, ohne dadurch unnötig an Komplexität zu gewinnen.

- **erweiterbar:** Die Konstruktion eines Modells, insbesondere bei komplizierten und noch nicht völlig verstandenen Themengebieten wie der Zell- und Molekularbiologie, stellt einen langen Prozess dar, bei dem unweigerlich zu späteren Zeitpunkten Anpassungen und Erweiterungen des Modells vorgenommen werden müssen. Dies sollte möglichst keine komplette Überarbeitung des Modells erfordern. Es bieten sich daher Modellierungsansätze mit modularem Aufbau an, in denen einzelne Komponenten problemlos ausgetauscht werden können.

- **anpassungsfähig:** Die Verschiedenartigkeit der drei zellulären Maschinen stellt einen besonders hohen Anspruch an die Anpassungsfähigkeit der formalen Beschreibungsmittel. Ein Ansatz mit sehr speziellen Operationen mag für die Modellierung einer der Makromolekülklassen besonders geeignet, dafür aber für die beiden anderen gänzlich ungeeignet sein. Es sind daher Ansätze mit sehr allgemeinen und elementaren Operationen zu bevorzugen.

- **stochastisch:** Die meisten Prozesse in lebenden Organismen scheinen nicht völlig deterministisch abzulaufen, sondern mit gewissen Wahrscheinlichkeiten behaftet zu sein. Ein Modell sollte auch diesem Aspekt Rechnung tragen.

- **quantitativ:** Einige der bisher entwickelten Modelle konzentrieren sich auf den qualitativen Aspekt der Thematik. Das heißt sie versuchen die Struktur und die Reihenfolge von Abläufe in einem System exakt zu erfassen, gehen allerdings nicht näher darauf ein, wie oft bestimmte Prozesse auftreten, mit welcher Dauer und in welchen Abständen dies geschieht. Das Verhalten der drei zellulären Maschinen hängt stark von quantitativen Aspekten ab, weshalb auch ein passendes Modell nicht auf deren Beschreibung verzichten sollte.

- **validierbar:** In der theoretischen Informatik, spielt stets auch die Validierbarkeit eines Modells eine große Rolle. In gleicher Weise profitiert die Systembiologie davon, wenn die genaue Einhaltung der Spezifikation für eine Formel nachgewiesen werden kann. Wie bei großen Softwareprojekten wächst die Fehlerwahrscheinlichkeit sowie die Schwierigkeit, Fehler aufzuspüren, mit der Größe der Kodierung. In gleichem Maße wächst damit auch die Notwendigkeit von formaler Verifikation.

- **dynamisch:** Im Gegensatz zu vielen grafischen Notationen, die lediglich den Ablauf eines Prozesses als statischen Plan darstellen, soll ein Modell im Sinne unserer Intention auch die Transitionen zwischen Zuständen nachvollziehbar machen und damit die dynamische Struktur des Systems modellieren.

- **nebenläufig:** Die Makromoleküle einer Zelle wie auch die Komponenten von biologischen Systemen im Allgemeinen verhalten sich wie unabhängige Agenten, die gleichzeitig verschiedene Berechnungen ausführen und diese gelegentlich in einer Kommunikation synchronisieren. Die Nebenläufigkeit und Autonomie der zu beschreibenden Komponenten sollte idealerweise auch im verwendeten Modell Berücksichtigung finden.

Einige der genannten Eigenschaften lassen sich problemlos in einem Modell vereinen, während andere sich gegenseitig behindern. Z.B. lassen sich auf natürliche Weise quantitative Aspekt in stochastischen Modellen unterbringen. Schwieriger wird es, wenn ein Modell trotz hoher Skalierbarkeit und Anpassungsfähigkeit noch intuitiv und verständlich sein soll, da die beiden erstgenannten Eigenschaften stets zu einem sehr abstrakten Ansatz führen, der sich evtl. weit von den intuitiven Vorstellungen des zu beschreibenden Systems entfernt. Wir wollen nun einige konkrete Modellierungsansätze vorstellen und später in Kapitel 6 bezüglich der hier genannten Eigenschaften vergleichen.

2.2.2 Einige Modellierungsansätze

Neben den Prozesskalkülen, in denen das Verhalten von Systemkomponenten durch nebenläufige, miteinander kommunizierende Prozesse beschrieben wird, existieren noch weitere Ansätze, die in dieser Arbeit nicht behandelt werden, aber dennoch nicht unerwähnt bleiben sollen. Ein äußerst wohluntersuchter sprachtheoretischer Ansatz ist mit den *P-Systemen* gegeben, in denen Berechnungen über Termersetzungsregeln ausgeführt werden. Der Ansatz ist direkt von biologischen Gegebenheiten motiviert und unterstützt insbesondere Membran-Operationen auf natürliche Weise. Als Einstieg sei die Arbeit [PeRo_06] empfohlen. Weiterhin existiert mit der *Systems Biology Markup Language* (SBML) eine allgemeine und strukturierte Beschreibungssprache für biologische Systeme ([EcLe_06]). Ein Programmiersprachenähnlicher Ansatz, der *Bio-calculus* ([NOMK_99]) ist besonders anpassungsfähig, da er eine einheitliche Syntax mit einer ganzen Auswahl verschiedener Semantiken kombinieren kann. Auch *Petri-Netze*, die ursprünglich nicht direkt im Zusammenhang mit der Systembiologie standen, werden mittlerweile in einigen Arbeiten (beispielsweise [StBW_06]) für Modellierungen biologischer Phänomene eingesetzt. Die grafischen Notationsmöglichkeiten dieses Ansatzes gewährleisten eine intuitive Erfassung der beschriebenen Vorgänge. Zuletzt sei noch erwähnt, dass es einige Arbeiten zum *Model Checking* biologischer Systeme gibt ([PeCo_03], [ChFa_03], [MaVa_05], [HKNPT_06]). Hierbei werden Spezifikationen in formalen Logiken (wie CTL) erstellt, die zur Analyse von Eigenschaften eines Systems ausgenutzt werden können.

2.3 Prozesskalküle

Ein Kalkül ist ein formaler Ansatz zur Beschreibung von Rechenvorgängen, wobei mit Rechenvorgängen im Sinne der Informatik jede Transformation von Information (*Input*) in neue Information (*Output*) gemeint ist. Viele Kalküle und kalkülähnliche Berechnungsmodelle wie Programmiersprachen fokussieren hierbei den Daten-Begriff, d.h. Informationen werden auf virtuellen Speicherplätzen abgelegt und mittels geeigneter Operationen manipuliert. Der Systemzustand wird durch die Menge der aktuell gespeicherten Daten charakterisiert. Im Gegensatz dazu sind Prozesskalküle solche Kalküle, die sich auf die Beschreibung des Verhaltens von Systemen spezialisieren. Dabei spielt die Einteilung des Systems in Prozesse, die jeweils eine Komponente des Systemverhaltens modellieren, eine entscheidende Rolle. Der Zustand eines Systems wird im Wesentlichen durch die verschiedenen Verhaltensmöglichkeiten, welche die Prozesse zu einem gegebenen Zeitpunkt besitzen, bestimmt.

Wir wollen in diesem Abschnitt einige Prozesskalküle vorstellen und deren Tauglichkeit für eine Anwendung in der Systembiologie herausarbeiten. Die Entstehung von Prozesskalkülen ist eng mit der wachsenden Bedeutung von verteilten und kommunizierenden technischen Systemen wie Telekommunikationsnetzen und modernen Computersystemen verbunden. Die Eigenschaften solcher Systeme und besonders deren Komplexität lassen die Analyse und Verifikation mit Hilfe von klassischen Modellen zunehmend schwieriger werden und schaffen die Notwendigkeit für neue Modelle, die dem kommunizierenden Aspekt der Systeme die erforderliche Bedeutung einräumen. Erst in jüngster Zeit haben auch die Systembiologen die Vorteile dieser Kalküle für ihre Zwecke entdeckt.

Während in den Kapiteln 3, 4 und 5 Prozesskalkül-Ansätze auf Basis des π-Kalküls vorgestellt und im Detail erläutert werden, soll der verbleibende Teil dieses Kapitels einige andere Konzepte, die aber ebenfalls in der Gruppe der Prozesskalküle anzusiedeln sind, in knapperer Form präsentieren, um später eine genaue Einordnung der π-Kalkül-Ansätze, die den Kern der Arbeit bilden, zu ermöglichen.

2.3.1 κ-Kalkül

Mit dem κ-Kalkül liegt ein Ansatz vor, der speziell für Proteininteraktionen entwickelt wurde und die in Abschnitt 2.1.2 vorgestellte grafische Notation mit einer formalen Semantik ausstattet. Die Autoren Vincent Danos und Cosimo Laneve abstrahieren von den Operationen der Protein-Maschine mit einer speziellen Form von Graphersetzungsregeln. Ein Protein wird hierbei als ein Knoten eines ungerichteten Graphen aufgefasst, so dass ein System, das aus einer Menge von Proteinen besteht, einen vollständigen Graph ergibt. Jede Bindungsstelle eines Proteins ist ein potenzieller Start- bzw. Endpunkt für eine Kante, d.h. im Fall einer Komplexbildung zwischen zwei Proteinen entsteht eine Kante und beim Auftrennen eines Komplexes wird eine Kante aus dem Graphen gelöscht. Des Weiteren können Bindungsstellen eines Proteins über bestimmte Regeln versteckt (unzugänglich) oder wieder sichtbar (verfügbar) gemacht werden. Schalter, das heißt Bindungsstellen, die nur einer Phosphorilierung dienen, werden in dem Ansatz vernachlässigt, da die Zielstellung lediglich darin besteht, eine möglichst einfache formale Beschreibungsmöglichkeit für die komplexen Vorgänge in Protein-Netzwerken anzugeben, ohne auf die Bedeutung von Proteinen für andere Moleküle, wie DNA-Stränge oder Zellmembranen, einzugehen.

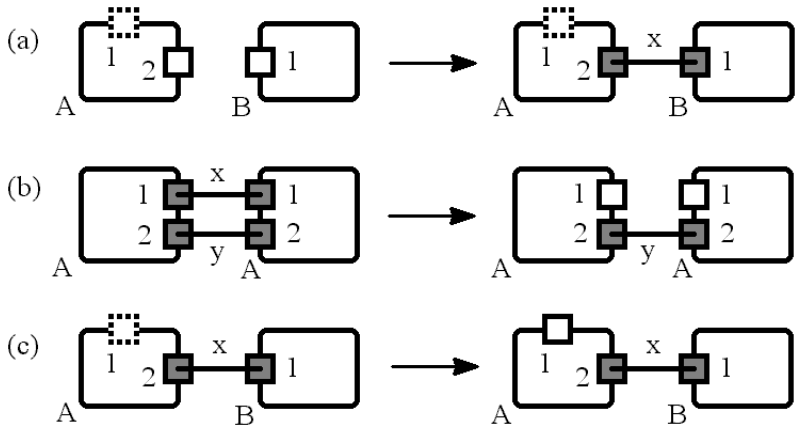


Abbildung 2.5: Beispiele für Graphersetzungsregeln im κ-Kalkül

Abbildung 2.5 zeigt beispielhaft einige Graphersetzungsregeln. Wir passen dabei die Darstellung aus Abbildung 2.3 an die Graph-Vorstellung an, indem wir in Proteinkomplexen nun eine Kante zwischen den involvierten Bindungsstellen zweier Proteine zeichnen, anstatt die Hülle der Proteine in direkten Kontakt zu setzen. Darüber hinaus sind die Bindungstellen eines Proteins nun durchnummeriert, um eine eindeutige Unterscheidung der versteckten, sichtbaren und in Komplexbildungen involvierten Bindungsstellen zu gewährleisten. Auch die Kanten werden zur exakten

Differenzierung beschriftet, wobei jeder Kante eindeutig ein Kleinbuchstabe zugeordnet wird. Der Name eines Proteins wird in Großbuchstaben an seine Außenhülle geschrieben.

Parallel zu der Graph-Notation existiert im κ-Kalkül auch eine textuelle Repräsentation von Proteinen und ihren Bindungsstellen, die sich über eine Menge von Termersetzungsregeln transformieren lässt. Diese Form lässt zwar die Übersichtlichkeit und Lesbarkeit der grafischen Notation vermissen, ist aber einer analytischen Betrachtung insbesondere in formalen Beweisen von Modelleigenschaften leichter zugänglich. Weiterhin lässt sich für die textuelle Darstellung auf einfachere Weise eine Übersetzung in andere Kalküle angeben. Die Regeln aus Abbildung 2.5 lassen sich wie folgt textuell darstellen.

(a) $A(\overline{1}, 2),\ B(1) \longrightarrow A(\overline{1}, 2^x),\ B(1^x)$

(b) $A(1^x, 2^y),\ A(1^x, 2^y) \longrightarrow A(1, 2^y),\ A(1, 2^y)$

(c) $A(\overline{1}, 2^x),\ B(1^x) \longrightarrow A(1, 2^x),\ B(1^x)$

Auf beiden Regelseiten stehen durch Kommata voneinander getrennte Proteine, die sich jeweils aus einem Proteinnamen und einem Tupel von Bindungsstellen zusammensetzen. Überstrichene Bindungsstellen sind versteckt, während Bindungsstellen mit einem Kleinbuchstaben im Exponenten in eine Komplexbildung über die entsprechende Kante involviert sind. Für die Regeln eines Modells gilt eine ganze Menge von Einschränkungen, um die Ableitungsschritte möglichst elementar zu halten. Beispielsweise darf nicht mehr als eine Kante durch eine Regel gelöscht oder erzeugt werden. Es ist möglich, die Bindungsstellen eines Proteins nicht vollständig anzugeben, um eine Art *pattern-matching* zu realisieren. In der Regel (a) könnte es beispielsweise irrelevant sein, ob Bindungsstelle 1 in Protein A versteckt ist oder nicht. Die passende Regel lautet in diesem Fall

(a′) $A(2),\ B(1) \longrightarrow A(2^x),\ B(1^x)$.

Beispiel 2.3.1: Wir betrachten als Ausgangssituation die Proteinmenge

$$A(1^x, 2^y),\ A(1^x, 2^y),\ A(\overline{1}, 2),\ B(1)\ .$$

Sie lässt sich gemäß Regel (a), (c) und (b) (in dieser Reihenfolge) wie folgt transformieren:

$$\begin{aligned}
&\longrightarrow A(1^x, 2^y),\ A(1^x, 2^y),\ A(\overline{1}, 2^x),\ B(1^x)\\
&\longrightarrow A(1^x, 2^y),\ A(1^x, 2^y),\ A(1, 2^x),\ B(1^x)\\
&\longrightarrow A(1, 2^y),\ A(1, 2^y),\ A(1, 2^x),\ B(1^x)
\end{aligned}$$

□

Abschließend sei anzumerken, dass diese Einführung in den κ-Kalkül seine Syntax und Semantik nur sehr skizzenhaft beschreibt. Eine ausführliche Arbeit über den Kalkül liegt mit [DaLa_04] vor. Die Arbeit behandelt neben einer exakten Beschreibung der Theorie auch eine Übersetzung in den π-Kalkül, den wir in Kapitel 3 vorstellen.

2.3.2 Brane-Kalkül

Der von Luca Cardelli entwickelte Brane-Kalkül ([Car2_05]) ist ähnlich wie der κ-Kalkül ein von grafischen Notationen der Biologen motivierter Formalismus, der sich einer ganz bestimmten Klasse biologischer Prozesse widmet. Während der κ-Kalkül Proteininteraktionen betrachtet, widmet sich der Brane-Kalkül den Membranoperationen. Die Grundidee des Kalküls besteht darin, sämtliche Berechnungen eines Systems an Membranen zu knüpfen und damit ein hohes Maß an Kontrolle über die räumliche Struktur des Systems zu erhalten. Ein System des Kalküls besteht aus parallel existierenden Membranen, die mit einer Reihe von Aktionen verknüpft sind und wiederum ein System aus beliebig vielen inneren Membranen enthalten können. Die Aktionen modellieren die in Abschnitt 2.1.3 beschriebenen Membranoperationen, wobei die Operationen Endocytose und Mito in je zwei speziellere Operationen aufgeteilt werden, da sonst ein unkontrollierbares „Verschlingen“ bzw. „Abspalten“ von beliebig vielen Komponenten in einem Schritt möglich wäre. Im Fall der Endocytose heißen die Spezialisierungen *Phagocytose* für das „Verschlingen“ genau einer externen Membran und *Pinocytose* für das Verschlingen von null externen Membranen (es wird eine leere Membran innerhalb der aktiven Membran gebildet). Die Mito-Operation teilt sich in die Spezialfälle *Bud* und *Drip* auf, wobei die erste das Abspalten genau einer internen Membran realisiert und die zweite das Abspalten von null internen Membranen (d.h. eine leere Membran trennt sich aus der aktiven Membran heraus).

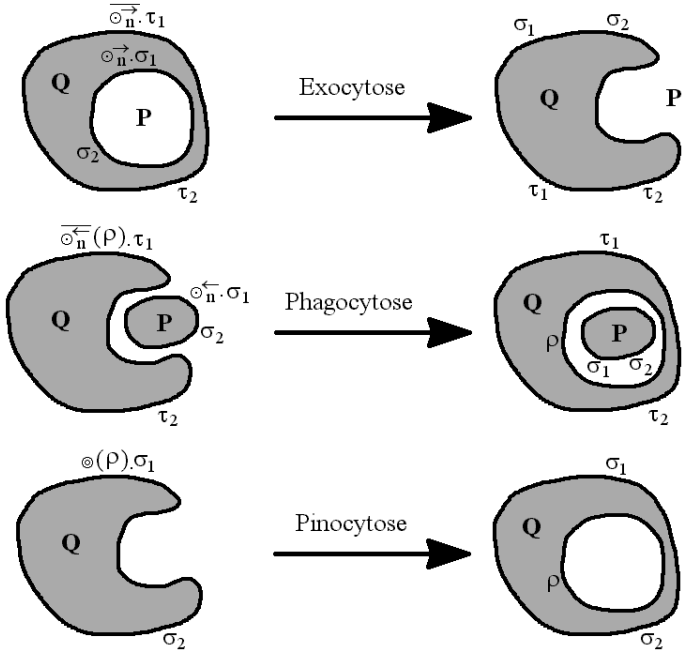


Abbildung 2.6: grafische Notation der Membranoperationen

Abbildung 2.6 zeigt die Operationen Exocytose, Phagocytose und Pinocytose in einer grafischen Repräsentation. Nachfolgend sind die gleichen Operationen in einer textuellen Repräsentation angegeben.

Exocytose: $\overline{\odot_n^{\rightarrow}}.\tau_1|\tau_2[\odot_n^{\rightarrow}.\sigma_1|\sigma_2[P] \circ Q] \longrightarrow P \circ \sigma_1|\sigma_2|\tau_1|\tau_2[Q]$

Phagocytose: $\odot_n^{\leftarrow}.\sigma_1|\sigma_2[P] \circ \overline{\odot_n^{\leftarrow}}(\rho).\tau_1|\tau_2[Q] \longrightarrow \tau_1|\tau_2[\rho[\sigma_1|\sigma_2[P]] \circ Q]$

Pinocytose: $\circledcirc(\rho).\sigma_1|\sigma_2[Q] \longrightarrow \sigma_1|\sigma_2[\rho[\diamond] \circ Q]$

Diese drei Operationen geben dem Kalkül bereits seine volle Expressivität, denn die übrigen drei Operationen lassen sich mit ihrer Hilfe ausdrücken, wie in [Car2_05] gezeigt wurde. In [BuGo_06] wird eine Variante des Kalküls vorgestellt, die stattdessen die anderen drei Operationen verwendet.

Wir kommen nun zu einigen Erläuterungen bezüglich der syntaktischen Elemente des Kalküls, ohne jedoch zu sehr ins Detail zu gehen. Ein Ausdruck der Form $\sigma[P]$ bezeichnet eine Membran, welche die durch σ spezifizierten Aktionen durchführen kann und in deren Inneren das System P liegt. Systeme bzw. Teilsysteme werden stets mit Großbuchstaben bezeichnet, während griechische Buchstaben die Fähigkeiten von Membranen repräsentieren. $P \circ Q$ und $\sigma|\tau$ stellen parallele Kompositionen von Systemen bzw. Membranfähigkeiten dar und $!P$, $!\sigma$ stehen für eine beliebige Anzahl identischer paralleler Systeme P bzw. Fähigkeiten σ. Membranfähigkeiten sind im Wesentlichen eine Sequenz von Aktionen $a_1.a_2.\ldots$, wobei in der hier vorgestellten Variante des Brane-Kalküls $\odot_n^{\rightarrow}$ für eine Exocytose, $\odot_n^{\leftarrow}$ für eine Phagocytose und $\circledcirc(\rho)$ für eine Pinocytose steht. Bei den Operationen der Exocytose und Phagocytose wird die Aktion von der zu verschlingenden bzw. der auszuspeienden Membran durchgeführt, während die verschlingende bzw. ausspeiende Membran jeweils synchron eine Gegenaktion $\overline{\odot_n^{\rightarrow}}$ bzw. $\overline{\odot_n^{\leftarrow}}(\rho)$ durchführt. In der Gegenaktion der Phagocytose und in der Pinocytose-Aktion werden zusätzlich die Fähigkeiten der entstehenden Membran als Parameter angegeben. Ausdrücke des Brane-Kalküls lassen sich mit Hilfe einer strukturellen Kongruenzrelation in äquivalente Ausdrücke umformen, um anschließend eine Operation anwenden zu können. Wir verzichten hier auf genaue Definitionen, merken allerdings an, dass Kongruenzrelation und Semantik des Kalküls eine starke Analogie zur Theorie des π-Kalküls aufweisen.

In der Literatur wird in der Regel nicht von *dem* Brane-Kalkül geschrieben, vielmehr wird unter dem Begriff eine Familie von Kalkülen zusammengefasst, deren Vertreter alle als Ausgangspunkt das Konzept der membrangebundenen Berechnung besitzen, sich aber in Syntax und Expressivität unterscheiden. In [Car2_05] wird beispielsweise eine Erweiterung um Moleküle wie Proteine und Nährstoffe angegeben, die sich durch chemische Reaktionen manipulieren lassen. Eine Verfeinerung dieser Erweiterung erlaubt das Modellieren von Proteinkomplexen. Auch eine Aufnahme von Kommunikationsaktionen im π-Kalkül-Stil bzw. im Ambienten-Kalkül-Stil wird diskutiert. Andere Kalkül-Varianten bereichern die Aktions-Syntax durch Auswahl-Operatoren zur Modellierung von Fallunterscheidungen. Die Anzahl der Arbeiten zu Brane-Kalkülen ist beträchtlich, wir empfehlen beispielsweise [DaPr_05], [Bus_06] und [MiBa_06].

2.3.3 Bio-Ambients

Der Bio-Ambients-Kalkül stellt eine Erweiterung des Ambienten-Kalküls dar, der wiederum auf dem π-Kalkül basiert. Da das Konzept der Ambients, also der Einbettung von Prozessen in abstrakte Hüllen mit einem eigenen Pool an Operationen, die Idee des klassischen π-Kalküls, sämtliche Berechnungen als synchrone Interaktionen zu verstehen, sehr stark erweitert, betrachten wir die Bio-Ambients als eigenständi-

gen Kalkül und nicht als Variante des π-Kalküls. Da wir den π-Kalkül sowie seine stochastischen Erweiterungen in den Kapiteln 3 und 4 noch genauer kennenlernen werden, beschränken wir uns in dieser knappen Einführung der Bio-Ambients auf einige Erläuterungen zu dem Konzept der Ambients.

Ähnlich wie beim Brane-Kalkül die Membranen sollen Ambients einerseits die visuelle Abgrenzung von (möglicherweise verschachtelten) Kompartimenten erlauben und andererseits die Operationen der Membran-Maschine in möglichst elementarer Form zur Verfügung stellen. Ein wesentlicher Unterschied zum Brane-Kalkül besteht darin, dass bei Bio-Ambients nicht die Hülle selbst zu Berechnungen in der Lage ist, sondern sämtliche Vorgänge durch die eingebetteten Prozesse realisiert werden. Dazu existieren neben den klassischen Kommunikationspräfixen, wie wir sie im Zusammenhang mit dem π-Kalkül kennenlernen werden, weitere Präfixe für die Aktionen von Ambients.

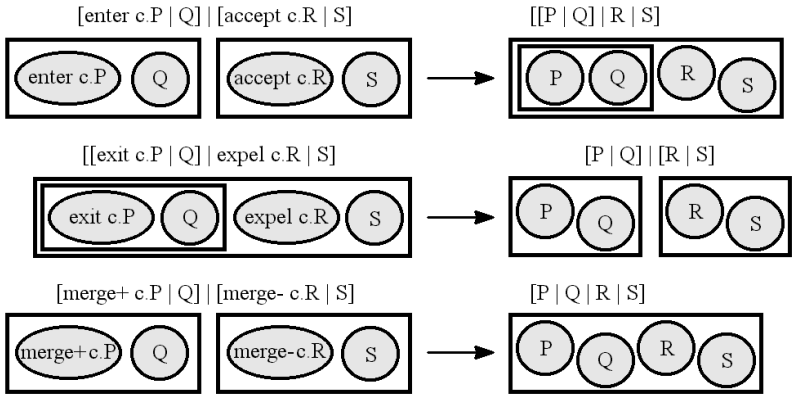


Abbildung 2.7: Ambient-Aktionen in grafischer und textueller Notation

Abbildung 2.7 zeigt die Aktionen, die jeweils durch Teilprozesse beider involvierter Ambients synchron eingeleitet werden müssen. Das Aktionspaar *enter*, *accept* ist mit einer Endocytose vergleichbar, während das Paar *exit*, *expel* das Analogon zu einer Exocytose darstellt. Mit *merge+* und *merge-* können zwei Ambients gemäß einer Mate-Operation zu einem Ambient verschmelzen.

Im Gegensatz zum π-Kalkül wird bei den Kommunikationspräfixen zwischen vier verschiedenen Typen unterschieden. Der Typ *local* entspricht der klassischen Version der Kommunikation im π-Kalkül. Zwei Teilprozesse können hierbei über komplementäre Präfixe (eins zum Senden und eins zum Empfangen von Information) innerhalb eines Ambients kommunizieren. Der Typ *s2s* (sibling to sibling) erlaubt auf gleiche Weise eine Interaktion zwischen zwei Ambients, die sich im selben übergeordneten Ambient befinden. Die Typen *c2p* (child to parent) und *p2c* (parent to child) ermöglichen das Senden von Information von einem Ambient zu seinem umgebenden Ambient bzw. in die umgekehrte Richtung.

Die Semantik des Kalküls der Bio-Ambients wird wie beim π-Kalkül über Ableitungsregeln im operationalen Stil angegeben, wobei eine Kongruenzrelation strukturelle Umformungen von Prozesstermen erlaubt, um die Vorbedingungen einer Regel zu erfüllen. Für eine detaillierte Einführung in die Theorie des Kalküls sei dem Leser die Arbeit [RPSCS_04] empfohlen.

Kapitel 3

π-Kalkül

Der π-Kalkül ist heute der bekannteste und bedeutendste interaktionsbasierte Prozesskalkül. Er wurde Ende der 1980er Jahre von Robin Milner, Joachim Parrow und David Walker als eine Erweiterung des von Robin Milner bereits in den 1970er Jahren vorgestellten *Calculus of Communicating Systems (CCS)* entwickelt und ist mittlerweile aufgrund einer Vielzahl an Publikationen ein überaus wohluntersuchter Ansatz zur Systemanalyse und -validierung. Für eine ausführliche Einführung in die Theorie seien dem interessierten Leser die Werke [Mil_99], [SaWa_01], [Par_01] und [Hof_03] empfohlen.

Wir stellen zunächst in Abschnitt 3.1 die Syntax des Kalküls sowie einige damit in Verbindung stehende Grundlagen vor. In den Abschnitten 3.2 und 3.3 werden zwei unterschiedliche Semantiken des π-Kalküls diskutiert. Des Weiteren wird auf Varianten des klassischen Kalküls eingegangen. Im Abschnitt 3.4 wenden wir uns schließlich einem Anwendungsbeispiel aus der Zellbiologie zu.

3.1 Syntax

Der π-Kalkül zeichnet sich durch eine einfache, auf wenige, aber sehr mächtige Elemente beschränkte Syntax aus. Die zwei zentralen Begriffe der Theorie sind die des *Prozesses* und des *Namens*. Ein Prozess ist ein Ausdruck, der auf abstrakte Weise das Verhalten eines beliebigen Objektes beschreibt. Ein Name kann ein beliebiges Objekt des zu beschreibenden Systems repräsentieren, wie beispielsweise einen Ort, einen Zustand oder eine physikalische Entität (wie Maschinen, Menschen, Zellen, Proteine). Am wichtigsten ist allerdings die Funktion eines Namens als *Kommunikationskanal* oder *Link*. Prozesse können über einen gemeinsamen Kanal synchronisiert Informationen austauschen. Dabei werden sowohl der Kanal wie auch die gesendeten Daten durch Namen repräsentiert. Dieses Konzept wird dadurch so mächtig, dass auch Kommunikationskanäle selbst zwischen Prozessen kommuniziert werden können und damit ein Modellierungsansatz für Mobilität entsteht. Durch die einem Prozess zur Verfügung stehenden Kommunikationskanäle wird sein physikalischer wie auch sein virtueller Bezug zu anderen Prozessen hergestellt. Durch das Löschen eines Kanals x und den Empfang eines anderen Kanals y kann beispielsweise die Bewegung eines Objektes von einem Ort x zu einem neuen Standpunkt y beschrieben werden, wobei alle Objekte am Ort x über Kanal x kommunizieren und alle am Ort y befindlichen Objekte über Kanal y.

Im Folgenden sei $\mathcal{N} = \{u, v, x, y, z, u', v', x', y', z', u_1, v_1, x_1, y_1, z_1, \ldots\}$ eine abzählbar unendliche Menge von Namen.

Definition 3.1.1 (π-Kalkül-Syntax): Die Menge der *π-Kalkül-Prozesse* (oder kurz *π-Prozesse*) ist durch folgende Grammatik gegeben:

$$P ::= \mathbf{0} \quad | \quad \pi.P \quad | \quad (\nu x)P \quad | \quad [x = y]P \quad | \quad !P \quad | \quad P|P \quad | \quad P + P$$

wobei π ein Aktionspräfix gemäß folgender Grammatik darstellt:

$$\pi ::= \overline{x}y \quad | \quad x(y) \quad | \quad \tau \qquad \square$$

Es ist sinnvoll, Präzedenzregeln für den Umgang mit komplexen, aus vielen Komponenten zusammengesetzten Prozessen anzugeben. Wir wollen hier festlegen, dass die oben definierten Operatoren von links nach rechts abnehmend stark binden. Zum Beispiel ist der Prozess

$$\overline{x}y.\mathbf{0} + !x(z).\mathbf{0} \mid \overline{u}v.\mathbf{0}$$

identisch zu

$$(\overline{x}y.\mathbf{0}) \; + \; (!\,(x(z).\mathbf{0}) \mid (\overline{u}v.\mathbf{0})\,) \; .$$

Im Folgenden wird das Verhalten von π-Prozessen, die in dieser Arbeit stets durch die Buchstaben P, Q und R bezeichnet werden, informell erläutert.

- $\mathbf{0}$ ist der untätige Prozess, der weder zur Kommunikation, noch zu irgendeinem anderen Verhalten fähig ist.

- Die Aktionsmöglichkeiten eines Prozesses $\pi.P$ werden durch das Präfix π bestimmt. Das Präfix in dem Prozess $\overline{x}y.P$ (Output) erlaubt das synchronisierte Senden des Namens y über den Kanal x. Danach verhält sich der Prozess gemäß der Definition von P. Die Synchronisation besteht hierbei darin, dass das Senden nur möglich ist, wenn ein anderer parallel ausgeführter Prozess vorhanden ist, der mit einem Präfix $x(z)$ (Input) beginnt, d.h. der einen Namen über den Kanal x empfangen will. Der empfangene Name wird für den Platzhalter z eingesetzt und steht damit für weitere Aktionen innerhalb des Prozesses zur Verfügung. Das Präfix τ wird verwendet, um eine nicht beobachtbare Aktion darzustellen. Wir nennen ein Präfix *geschützt*, wenn es innerhalb eines durch ein anderes Präfix begonnenen Prozesses liegt. Es kann in diesem Fall nicht in Aktionen verwickelt werden. Beispielsweise ist das Präfix $x(z)$ in $x(y).x(z).\mathbf{0}$ geschützt.

- Mit der Restriktion $(\nu x.P)$ wird der Name x lokal auf den Prozess P begrenzt. In diesem Fall heißt x *privat* in P. Innerhalb der Komponenten von P darf x also als Kommunikationskanal genutzt werden, ein Senden von Namen an andere Prozesse über x ist allerdings nicht möglich. Der Gültigkeitsbereich eines privaten Namens kann erweitert werden, indem der Name einem Prozess außerhalb des Gültigkeitsbereiches über einen nicht privaten Kanal gesendet wird. Dieser Vorgang wird *Scope Extrusion* genannt.

- Das Verhalten eines Prozesses kann von einem Vergleich zweier Namen abhängig gemacht werden. Der Prozess $[x = y]P$ zeigt das Verhalten von P, falls die Namen x und y identisch sind, andernfalls verhält er sich wie der untätige Prozess $\mathbf{0}$.

- In der parallelen Komposition $P|Q$ agieren die beiden Teilprozesse P und Q gleichzeitig und unabhängig voneinander und können über gemeinsame Kanäle kommunizieren.
- Der als Summe bezeichnete Prozess $P + Q$ verhält sich nichtdeterministisch entweder gemäß P oder gemäß Q und verwirft den jeweils anderen Teilprozess.
- Der Replikationsoperator ! erlaubt es, ein unendliches Verhalten zu formalisieren. Der Prozess $!P$ steht für eine unbegrenzte Anzahl parallel ausgeführter Prozesse P.

Beispiel 3.1.2: In dem Prozess

$$\overline{x}y.\mathbf{0} \mid x(z).\overline{x'}z.\mathbf{0}$$

können die beiden parallelen Komponenten über den Kanal x miteinander kommunizieren. Die erste Komponente sendet den Namen y, der in der zweiten für sämtliche Vorkommen des Namens z eingesetzt wird. Das Ergebnis der Interaktion ist

$$\mathbf{0} \mid \overline{x'}y.\mathbf{0} \ .$$

Die zweite Komponente wäre nun in der Lage, den empfangenen Namen über den Kanal x' weiterzusenden, wenn es einen Interaktionspartner für diesen Kanal gäbe.□

Um das Einsetzen von empfangenen Namen für alle Vorkommen eines Eingabeparamters in einem Prozess der Form $x(y).P$ zu formalisieren, verwenden wir das Prinzip der syntaktischen Substitution. Doch zuvor benötigen wir die Begriffe der gebundenen und freien Vorkommen von Namen.

Definition 3.1.3: In Prozessen der Form $x(z).P$ und $(\nu z)P$ heißen alle Vorkommen des Namens z *gebunden*. P heißt hierbei *Gültigkeitsbereich* der Restriktion bzw. des Input-Präfixes. Ein Vorkommen eines Namens heißt *frei*, falls es nicht gebunden ist. Die Menge der Namen, die in P wenigstens ein freies Vorkommen haben, wird mit $\mathrm{fn}(P)$ bezeichnet und die Menge der Namen, die in P gebunden vorkommen mit $\mathrm{bn}(P)$. □

Beispiel 3.1.4: In folgendem Prozess ist das erste und das letzte Vorkommen von x frei, während die beiden mittleren Vorkommen gebunden sind. Alle Vorkommen von z sind gebunden und das Vorkommen von y ist frei.

$$x(z).(\nu x)\overline{z}x.\mathbf{0} \mid \overline{x}y.\mathbf{0}$$

Tatsächlich läßt sich die Restriktion wie die Deklaration einer lokalen Variable in einer Prozedur einer Programmiersprache auffassen, während das Input-Präfix wie eine Parameterübergabe an eine Prozedur zu verstehen ist (mit dem Unterschied, dass die Übergabe des Parameters hier nicht zwingend beim Aufruf, also Prozessbeginn, stattfinden muss). Die beiden Komponenten der obigen Komposition verhalten sich demnach wie parallel ablaufende Prozeduren, von denen die erste den Parameter z empfängt und x lokal deklariert, wodurch sie den Zugriff auf das freie (globale) x verliert. □

Definition 3.1.5 (Substitution): Eine *Substitution* ist eine Funktion $\sigma : \mathcal{N} \to \mathcal{N}$ mit

$$\sigma(x) = \begin{cases} y_i & \text{für } x = x_i \in \{x_1, \ldots, x_n\},\ n \in \mathbb{N} \\ x & \text{sonst} \end{cases}$$

falls die Mengen $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_n\}$ den Definitionsbereich bzw. Wertebereich von σ angeben. Statt $\sigma(x)$ schreiben wir auch $x\sigma$ oder $x\{y_1, \ldots, y_n/x_1, \ldots, x_n\}$. Der *support* einer Substitution σ ist definiert als $\text{supp}(\sigma) = \{x \mid x\sigma \neq x\}$ und bezeichnet die Menge der Namen, die durch die Substitution ersetzt werden. □

Die Anwendung einer Substitution σ auf einen Prozess bedeutet, dass alle Vorkommen von Namen $x \in \text{supp}(\sigma)$ durch $x\sigma$ ersetzt werden. Dabei ist jedoch sicherzustellen, dass keine ehemals freien Namen nach dem Ersetzen gebunden sind. Der Prozess $P = y(z).\overline{z}x.\mathbf{0}$ würde durch die Anwendung der Substitution $\sigma = \{z/x\}$ z.B. fälschlicherweise durch den Prozess $P' = y(z).\overline{z}z.\mathbf{0}$ ersetzt. Das Vorkommen des Namens x ist in P frei, das entsprechende Vorkommen von z in P' allerdings gebunden. Um dies zu vermeiden, führen wir die α-Konversion ein, die ein Umbenennen von gebundenen Namen gestattet. In unserem Beispiel würde der problematische Name z in einen noch nicht verwendeten Namen konvertiert. Nach anschließender Anwendung der Substitution σ ergibt sich beispielsweise der Prozess $P'' = y(z').\overline{z'}z.\mathbf{0}$.

Definition 3.1.6 (α-Konversion): Die Substitution der gebundenen Vorkommen eines Namens z in einem Prozess oder Teilprozess $x(z).P$ bzw. $(\nu z)P$ durch einen vorher nicht verwendeten Namen (mit Resultat $x(z').P\{z'/z\}$ bzw. $(\nu z')P\{z'/z\}$) wird als *α-Konversion* bezeichnet. Zwei Prozesse P und Q, die durch eine endliche Anzahl von α-Konversionen ineinander überführt werden können, heißen *α-konvertibel* (Notation: $P =_\alpha Q$) und werden im Folgenden als identisch erachtet. □

Beispiel 3.1.7: Der folgende Prozess veranschaulicht die Anwendung von Substitution und α-Konversion bei einer Kommunikation von Teilprozessen.

$$(\nu z)(x(y).\overline{z}y.\mathbf{0} \mid z(u).\mathbf{0}) \mid \overline{x}z.\overline{z}y.\mathbf{0}$$

Die Interaktion der ersten und letzten Komponente des Prozesses über den gemeinsamen Kanal x führt zu dem Prozessterm

$$(\nu z)(\overline{z}y.\mathbf{0}\{z/y\} \mid z(u).\mathbf{0}) \mid \overline{z}y.\mathbf{0}\,.$$

Die Anwendung der Substitution $\{z/y\}$ ist allerdings erst nach α-Konvertierung der gebundenen Vorkommen des Names z möglich. Wir wählen den neuen Namen z' und erhalten

$$(\nu z')(\overline{z'}z.\mathbf{0} \mid z'(u).\mathbf{0}) \mid \overline{z}y.\mathbf{0}\,.$$

Das Vorkommen von z in der letzten Komponente des Prozesses wird durch die α-Konversion nicht berührt, da es nach wie vor frei vorkommt. □

3.2 Reduktionssemantik

In diesem Abschnitt wollen wir die bereits informell erläuterte Bedeutung der syntaktischen Elemente von π-Prozessen durch Angabe einer operationellen Semantik

formal definieren. Die Idee der Semantik ist, dass ein Prozess P mittels Anwendung von Reduktionsregeln zu einem Prozess P' evolvieren kann. Wir schreiben hierfür $P \longrightarrow^* P'$, wobei $\longrightarrow^*$ die reflexive und transitive Hülle der *Reduktionsrelation* $\longrightarrow$ ist, die in Definition 3.2.1 eingeführt wird.

Definition 3.2.1 (Reduktion): Ein Prozess P kann genau dann zu einem Prozess P' evolvieren, wenn sich $P \longrightarrow^* P'$ mittels einer endlichen Anzahl von Anwendungen der sechs im Folgenden aufgeführten Regeln ableiten läßt.

$$\textsc{Inter:}\quad \frac{}{(\overline{x}y.P_1 + Q_1) \mid (x(z).P_2 + Q_2) \longrightarrow P_1 \mid P_2\{y/z\}}$$

$$\textsc{Tau:}\quad \frac{}{\tau.P + Q \longrightarrow P}$$

$$\textsc{Sum:}\quad \frac{P_1 \longrightarrow P_1'}{P_1 + P_2 \longrightarrow P_1'} \qquad \textsc{Par:}\quad \frac{P_1 \longrightarrow P_1'}{P_1 \mid P_2 \longrightarrow P_1' \mid P_2}$$

$$\textsc{Res:}\quad \frac{P \longrightarrow P'}{(\nu z)P \longrightarrow (\nu z)P'} \qquad \textsc{Struct:}\quad \frac{Q \longrightarrow Q',\ Q \equiv P,\ Q' \equiv P'}{P \longrightarrow P'}$$

Eine Folge von Prozessen, in der jeder Prozess jeweils über eine Menge von Regelanwendungen aus seinem Vorgänger hervorgeht (Notation: $P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_n$), wird *Reduktionsableitung* genannt. Dabei findet in jedem *Ableitungsschritt* $P_i \rightarrow P_{i+1}$ maximal eine Anwendung der Regeln Inter oder Tau statt. □

Im Zähler jeder Regel sind die Bedingungen angegeben, die für den im Nenner spezifizierten Reduktionsschritt erfüllt sein müssen. Da die Bedingungen jeweils weitere Regelanwendungen erforderlich machen, müssen alle Reduktionsableitungen mindestens eine der Regeln Inter oder Tau enthalten. Wir wollen die Bedeutung der einzelnen Regeln nun etwas genauer betrachten.

Inter: Mit dieser Regel wird die Kommunikation zweier paralleler Teilprozesse über einen gemeinsamen Kanal x realisiert. Der Name y wird im empfangenden Prozess per Substitution für den Platzhalter z eingesetzt. In beiden Prozessen werden evtl. vorhandene zusätzliche Summanden eliminiert.

Tau: Auch bei der nicht beobachtbaren Aktion werden zusätzliche Summanden eliminiert und das Präfix, in diesem Fall τ wird „verbraucht“.

Sum, Par: Diese Regeln gewährleisten, dass Interaktionen auch innerhalb von alternativen bzw. parallelen Komponenten eines Prozesses stattfinden können. Bei Alternativen werden alle nicht benötigten Komponenten verworfen, während bei parallelen Teilprozessen die übrigen Komponenten unberührt bleiben.

Res: Analog zu den Regeln Sum und Par ermöglicht die Regel Res Interaktionen innerhalb einer Restriktion. Die Kommunikation über ein Präfix innerhalb der Restriktion mit einem außerhalb befindlichen Präfix ist nicht direkt möglich. Allerdings lässt sich der Restriktionsoperator gegebenenfalls durch Umstrukturierung des Prozessterms verschieben (siehe strukturelle Kongruenz).

Struct: Die Regel STRUCT nimmt eine besondere Rolle ein. Sie gestattet das Restrukturieren von Prozessen in eine semantisch äquivalente Form. Dabei kommen weitere Regeln zum Einsatz, die in nachfolgender Definition aufgeführt sind.

Definition 3.2.2 (Strukurelle Kongruenz): Zwei Prozesse P und Q, die sich durch Anwendung der folgenden Regeln ineinander überführen lassen, heißen *strukturell kongruent* (Notation $P \equiv Q$).

(1) $P \equiv Q$, falls $P =_\alpha Q$

(2) $P + Q \equiv Q + P, \quad P + (Q + R) \equiv (P + Q) + R, \quad P + \mathbf{0} \equiv P$

(3) $P \mid Q \equiv Q \mid P, \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R, \quad P \mid \mathbf{0} \equiv P$

(4) $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P, \quad (\nu x)(P|Q) \equiv P|(\nu x)Q$ falls $x \notin \mathrm{fn}(P)$,
$(\nu x)(P + Q) \equiv P + (\nu x)Q$ falls $x \notin \mathrm{fn}(P), \quad (\nu x)\mathbf{0} \equiv \mathbf{0}$

(5) $!P \equiv P \,|\, !P$

(6) $[x = x]P \equiv P$ □

Die Strukturelle Kongruenzrelation $\equiv$ modelliert sowohl die Möglichkeit von α-Konversionen, als auch die Kommutativität und Assoziativität der binären Operatoren Summe und parallele Komposition. Darüber hinaus ermöglicht sie, den Gültigkeitsbereich einer Restriktion über Teilprozesse auszudehnen, in denen der gebundene Name nicht frei vorkommt. Außerdem wird das „Entfalten“ einer Replikation und das Anwenden des Vergleichsoperators als strukurelle Kongruenzumformung verstanden und nicht etwa mit einer zusätzlichen Reduktionsregel umgesetzt. Der Vorteil dieser Kapselung von Umformungen in einer Krongruenzrelation liegt in der geringen Anzahl und eleganten Einfachheit der Reduktionsregeln.

Beispiel 3.2.3: Die folgenden Prozesse sind paarweise strukturell kongruent.

$$\begin{aligned}
& y(z).z(u).\mathbf{0} \mid !(\nu z)(z(x).\mathbf{0} \mid \overline{z}u.\mathbf{0}) \\
\equiv\ & y(z).z(u).\mathbf{0} \mid (\nu z)(z(x).\mathbf{0} \mid \overline{z}u.\mathbf{0}) \mid !(\nu z)(z(x).\mathbf{0} \mid \overline{z}u.\mathbf{0}) \\
\equiv\ & y(z).z(u).\mathbf{0} \mid (\nu z')(z'(x).\mathbf{0} \mid \overline{z'}u.\mathbf{0}) \mid !(\nu z)(z(x).\mathbf{0} \mid \overline{z}u.\mathbf{0}) \\
\equiv\ & (\nu z')(y(z).z(u).\mathbf{0} \mid z'(x).\mathbf{0} \mid \overline{z'}u.\mathbf{0}) \mid !(\nu z)(z(x).\mathbf{0} \mid \overline{z}u.\mathbf{0})
\end{aligned}$$

Bei jeder Replikation der letzten Komponente wird ein neuer gebundener Name z erzeugt, der verschieden zu allen anderen Vorkommen von z in den anderen Komponenten des Prozesses ist. Die α-Konversion von z zu z' im replizierten Prozess (dritte Zeile) veranschaulicht dies. □

Jegliches nicht endliches Verhalten eines Systems muss nach der vorgestellten Syntax und Semantik des π-Kalküls mittels Replikationsoperatoren realisiert werden. Dieser Operator ist in Beweisen von π-Kalkül-Eigenschaften und der Verfikation von Modellen leicht zu handhaben. Ein Nachteil besteht jedoch in der schlechten Lesbarkeit von größeren Modellierungen. Ein Prozessterm, der mehrere verschachtelte Vorkommen des Operators enthält, erschließt sich unserer Vorstellung erst nach eingehendem Studium. Eine Alternative zum Replikationsoperator, die eine Anlehnung

an programmiersprachliche Konzepte darstellt, ist die Verwendung von rekursiven Definitionsgleichungen der Form

$$A(x_1, \ldots, x_n) = P\,.$$

Ein System nach dieser alternativen Syntax besteht aus einer endlichen Menge solcher Gleichungen sowie einem initialen Prozess, der den Ausgangspunkt einer Ableitung nach der Reduktionssemantik darstellt. Statt des Replikationsoperators sind in der Syntax der π-Prozesse nun *Bezeichner* der Form $A\langle y_1, \ldots, y_n\rangle$ erlaubt, die wie ein Funktionsaufruf mit n Parametern zu verstehen sind. Auch auf der rechten Seite einer Definitionsgleichung dürfen solche Bezeichner auftauchen, insbesondere der Bezeichner, der in der Gleichung definiert wird, wodurch rekursives Verhalten zum Ausdruck gebracht werden kann. Um das Verhalten dieser Terme in der Semantik zu verankern, wird die Regel (5) der strukturellen Kongruenz durch die folgende Regel ersetzt:

$$(5) \quad A\langle y_1, \ldots, y_n\rangle \equiv P\{y_1, \ldots, y_n/x_1, \ldots, x_n\}, \text{ falls } A(x_1, \ldots, x_n) = P$$

Beispiel 3.2.4: Der nach der ursprünglichen Syntax des π-Kalküls definierte Prozess

$$P \;=\; x(y).([y = v_1]!(\overline{y}z.\mathbf{0} \mid y(u).\overline{y}u.\mathbf{0}) + [y = v_2]!(\overline{x}z.\mathbf{0} \mid x(u).\overline{x}u.\mathbf{0}))$$

lässt sich in folgendes System nach der Syntax mit Definitionsgleichungen überführen.

$$\begin{array}{rcl} Q & = & x(y).([y = v_1]R\langle y\rangle + [y = v_2]R\langle x\rangle) \\ R(x_1) & = & \overline{x_1}z.R\langle x_1\rangle \mid x_1(u).\overline{x_1}u.R\langle x_1\rangle \end{array}$$

Das System mit initialem Prozess Q weist das gleiche Verhalten auf wie der Prozess P. □

Die Frage, ob eine der beiden Varianten des Kalküls Verhalten modellieren kann, das die jeweils andere nicht realisieren kann, ist naheliegend. In [SaWa_01] wird gezeigt, dass dies nicht möglich ist, beide Varianten also die gleiche Berechnungsstärke besitzen.

Eine weitere nützliche Variante ist der *polyadische π-Kalkül*, in dem es gestattet ist, Tupel von Namen mit einem einzigen Präfix zu senden bzw. zu empfangen. Systeme, in denen viele Informationen über den gleichen Kanal kommuniziert werden, lassen sich damit deutlich übersichtlicher und eleganter modellieren. Die Aktionspräfixe werden hierfür wie folgt erweitert.

$$\pi ::= \; \overline{x}(y_1, \ldots, y_n) \quad | \quad x(y_1, \ldots, y_n) \quad | \quad \tau$$

Die Regel Inter hat dementsprechend die Gestalt

$$\frac{n \geq 0}{(\overline{x}(y_1, \ldots, y_n).P_1 + Q_1) \mid (x(z_1, \ldots, z_n).P_2 + Q_2) \longrightarrow P_1 \mid P_2\{y_1, \ldots, y_n/z_1, \ldots, z_n\}}$$

Man beachte, dass beide Präfixe die gleiche Anzahl an Namen in ihrem Argument besitzen müssen und dass insbesondere auch das Senden eines leeren Tupels ($n = 0$) möglich ist. Letzteres ist als eine Art Handschlag zwischen den Kommunikationspartnern zu verstehen, bei dem lediglich eine Synchronisierung, jedoch kein Informationsaustausch stattfindet. Auch diese Variante des π-Kalküls liefert keine größere

Berechnungsstärke, denn polyadische Eingabe und Ausgabe lassen sich auch mit einer Sequenz von einstelligen Präfixen ausdrücken. Dazu reicht es allerdings nicht aus, die Namen nacheinander über den gleichen Kanal zu senden und auf der anderen Seite entsprechend zu empfangen, da hierbei Informationen irrtümlich beim falschen Empfänger ankommen können, wie das folgende Beispiel zeigt.

Beispiel 3.2.5: Der Prozess P des polyadischen π-Kalküls, der das Senden der drei Namen y_1, y_2 und y_3 an einen von zwei Empfängern (zweite und dritte parallele Komponente) realisiert, soll in einen äquivalenten Prozess der klassischen (*monadischen*) Variante des π-Kalküls übersetzt werden.

$$P = \overline{x}(y_1, y_2, y_3).P_1 \mid x(z_1, z_2, z_3).P_2 \mid x(u_1, u_2, u_3).P_3$$

Der folgende Prozess ist keine korrekte Übersetzung, da die erste Komponente für jeden zu sendenden Namen erneut einen Empfänger wählt, der im Fehlerfall nicht identisch mit dem Empfänger des ersten Namens ist.

$$\begin{array}{rcl} P_{Fehler} & = & \overline{x}y_1.\overline{x}y_2.\overline{x}y_3.P_1 \mid x(z_1).x(z_2).x(z_3).P_2 \mid x(u_1).x(u_2).x(u_3).P_3 \\ & \longrightarrow & \overline{x}y_2.\overline{x}y_3.P_1 \mid x(z_2).x(z_3).P_2\{y_1/z_1\} \mid x(u_1).x(u_2).x(u_3).P_3 \\ & \longrightarrow & \overline{x}y_3.P_1 \mid x(z_2).x(z_3).P_2\{y_1/z_1\} \mid x(u_2).x(u_3).P_3\{y_2/u_1\} \\ & \longrightarrow & P_1 \mid x(z_3).P_2\{y_1, y_3/z_1, z_2\} \mid x(u_2).x(u_3).P_3\{y_2/u_1\} \end{array}$$

In der Beispielableitung wird y_2 von der dritten Komponente empfangen, während y_1 und y_3 von der zweiten Komponente empfangen werden. Eine korrekte Übersetzung stellt der Prozess

$$P' = (\nu x)\overline{v}x.\overline{x}y_1.\overline{x}y_2.\overline{x}y_3.P_1 \mid v(x).x(z_1).x(z_2).x(z_3).P_2 \mid v(x).x(u_1).x(u_2).x(u_3).P_3$$

dar, in dem zunächst ein privater Kanal x, der für die nachfolgenden Interaktionen verwendet wird, über den globalen Kanal v an einen der beiden Empfänger gesandt wird. Anschließend kann nur dieser Empfänger die drei Namen y_1, y_2 und y_3 erhalten, da nur er und der sendende Teilprozess den privaten Namen x kennen. □

Wir kommen nun zu einer Kritik an der bisher definierten Semantik. Die Reduktionssemantik erlaubt die Kommunikation zwischen Prozessen nur, falls diese als Teilprozesse eines übergeordneten Gesamtprozesses vorliegen. Daraus ergibt sich ein gravierender Nachteil für eine Modellierung großer Systeme: Alle Komponenten des Systems müssen, falls Sie miteinander kommunizieren, in einem einzigen Prozess zusammengefasst werden. Die Aktionen innerhalb dieser Semantik werden daher auch *Intraaktionen* genannt. Für eine Modellierung in der Systembiologie ist eine Semantik wünschenswert, die es gestattet, kommunizierende Systemkomponenten einzeln zu implementieren und damit echte *Interaktionen* zwischen verschiedenen Prozessen bzw. zwischen einem Prozess und seiner nicht spezifizierten Umgebung zulässt. Eine solche Semantik wird im nächsten Abschnitt vorgestellt und diskutiert.

3.3 Transitionssemantik

Die *Transitionssemantik* des π-Kalküls stellt eine Verallgemeinerung der Reduktionssemantik dar, in der die Regeln der strukturellen Kongruenzrelation sowie die Reduktionsregeln in einer Menge von Transitionsregeln vereinigt sind. Statt einer

Reduktionsrelation gibt es nun vier Transitionsrelationen, die jeweils mit einer von vier Aktionen verknüpft sind, die von Prozessen in der Umgebung beobachtet werden können und eine Kommunikation über Prozessgrenzen hinweg ermöglichen.

- Die *Eingabeaktion* $x(y)$ ermöglicht einem Prozess P den Namen y über den Kanal x zu empfangen und dabei zu P' zu evolvieren. Die Transitionsrelation $\xrightarrow{x(y)}$ drückt dieses Verhalten aus: $P \xrightarrow{x(y)} P'$.
- Die *freie Ausgabeaktion* $\overline{x}y$ ermöglicht entsprechend das Senden des Namens y über Kanal x an einen fremden Prozess. Wir schreiben hierfür $P \xrightarrow{\overline{x}y} P'$. Bei der freien Ausgabe ist der gesendete Name als global zu verstehen, gleiche Namen in verschiedenen Prozessen sind also identisch.
- Zusätzlich zur freien Ausgabeaktion gibt es auch noch die *gebundene Ausgabeaktion* $\overline{x}\nu y$, die ebenfalls das Senden des Namens y über Kanal x ermöglicht, allerdings mit dem Unterschied, dass der Name y zuvor nur dem sendenden Prozess bekannt ist. Ein evtl. bereits vorhandener Name y im empfangenden Prozess ist nicht mit dem gesendeten y identisch. Namenskonflikte müssen daher per α-Konversion vermieden werden. Für eine Transition des Prozesses P, bei der eine gebundene Ausgabe stattfindet, schreiben wir $P \xrightarrow{\overline{x}\nu y} P'$.
- Zuletzt gibt es noch die *interne Aktion* τ, die eine Intraaktion eines Prozesses realisiert, die von fremden Prozessen nicht beobachtet werden kann. Ein Evolvieren des Prozesses P zu P' über eine interne Aktion wird mit $P \xrightarrow{\tau} P'$ notiert. Wir werden später sehen, dass die interne Aktion jegliches in der Reduktionssemantik beschreibbare Verhalten modelliert.

Für die Definition der Transitionsrelationen spielt der Begriff der gebundenen Namen auch in den Aktionen eine Rolle. Wir legen an dieser Stelle fest, dass alle verwendeten Namen in den Aktionen $x(y)$ und $\overline{x}y$ frei vorkommen, während in $\overline{x}\nu y$ der Name x frei und der Name y gebunden vorkommt.

Definition 3.3.1 (Transition): Ein Prozess P kann genau dann zu einem Prozess P' evolvieren, wenn sich eine *Transitionsableitung* der Form $P \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} P'$ mit den *Transitionsbeschriftungen* $\alpha_i \in \{x(y), \overline{x}y, \overline{x}\nu y, \tau\}$ mittels einer endlichen Anzahl von Anwendungen der nachfolgend aufgeführten Regeln ableiten läßt.

$$\textsc{Out}: \frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \qquad \textsc{Inp}: \frac{}{x(z).P \xrightarrow{x(y)} P\{y/z\}}$$

$$\textsc{Tau}: \frac{}{\tau.P \xrightarrow{\tau} P} \qquad \textsc{Match}: \frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}$$

$$\textsc{Sum-L}: \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \textsc{Sum-R}: \frac{P \xrightarrow{\alpha} P'}{Q + P \xrightarrow{\alpha} P'}$$

$$\textsc{Par-L}: \frac{P \xrightarrow{\alpha} P',\ \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \textsc{Par-R}: \frac{P \xrightarrow{\alpha} P',\ \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{Q \mid P \xrightarrow{\alpha} Q \mid P'}$$

$$\textsc{Comm-L:}\quad \frac{P \xrightarrow{\overline{x}y} P', \quad Q \xrightarrow{x(y)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \textsc{Comm-R:}\quad \frac{P \xrightarrow{\overline{x}y} P', \quad Q \xrightarrow{x(y)} Q'}{Q \mid P \xrightarrow{\tau} Q' \mid P'}$$

$$\textsc{Close-L:}\quad \frac{P \xrightarrow{\overline{x}\nu y} P', \quad Q \xrightarrow{x(y)} Q', \quad y \notin \mathrm{fn}(Q)}{P \mid Q \xrightarrow{\tau} (\nu y)(P' \mid Q')}$$

$$\textsc{Close-R:}\quad \frac{P \xrightarrow{\overline{x}\nu y} P', \quad Q \xrightarrow{x(y)} Q', \quad y \notin \mathrm{fn}(Q)}{Q \mid P \xrightarrow{\tau} (\nu y)(Q' \mid P')}$$

$$\textsc{Res:}\quad \frac{P \xrightarrow{\alpha} P', \quad z \notin \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'} \qquad \textsc{Open:}\quad \frac{P \xrightarrow{\overline{x}y} P', \quad y \neq x}{(\nu y)P \xrightarrow{\overline{x}\nu y} P'}$$

$$\textsc{Rep-Act:}\quad \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P} \qquad \textsc{Rep-Comm:}\quad \frac{P \xrightarrow{\overline{x}y} P', \quad P \xrightarrow{x(y)} P''}{!P \xrightarrow{\tau} (P' \mid P'') \mid !P}$$

$$\textsc{Rep-Close:}\quad \frac{P \xrightarrow{\overline{x}\nu y} P', \quad P \xrightarrow{x(y)} P'', \quad y \notin \mathrm{fn}(P)}{!P \xrightarrow{\tau} (\nu y)(P' \mid P'') \mid !P}$$

□

Man beachte, dass an beliebiger Stelle stets eine Umbenennung von gebundenen Namen mittels einer α-Konversion möglich und teilweise, wie in den nachfolgenden Erläuterungen gezeigt, auch notwendig ist, um Verwechslungen von ursprünglich verschiedenen Namen zu vermeiden.

Out, Inp, Tau: Diese drei Regeln realisieren das Senden und Empfangen von Namen sowie die nicht beobachtbare Aktion. Im Gegensatz zur Reduktionssemantik sind hier keine Kommunikationspartner erforderlich. Dies spiegelt die Sichtweise eines einzelnen Prozesses wieder, für den es keine Rolle spielt, woher er eine Nachricht bekommt, oder wohin er sie versendet. Die Reduktionssemantik nimmt hingegen eher die Sichtweise eines globalen Beobachters ein, der alle Interaktionspartner kennt. Als Beispiel für die Regeln betrachte man die einfache Ableitung

$$x(z).\overline{z}y.\mathbf{0} \xrightarrow{x(u)} \overline{u}y.\mathbf{0} \xrightarrow{\overline{u}y} \mathbf{0},$$

die sich erst der Regel Inp und anschließend der Regel Out bedient.

Match: Ein Prozess mit einem erfolgreichen Namensvergleich wird durch diese Regel mit einem Prozess ohne diesen Vergleich identifiziert. Man beachte die Ähnlichkeit mit der entsprechenden Regel der strukturellen Kongruenz in Abschnitt 3.2.

Sum-L, Sum-R: Durch diese beiden Regeln werden Transitionen innerhalb von Summen unter Verwerfung der nicht benötigten Alternativen realisiert. Da wir in der Transitionssemantik die Regeln der strukturellen Kongruenz in die Transitionsregeln integriert haben, müssen wir für beide Operanden des Auswahloperators eine eigene Regel definieren. In gleicher Weise gilt dies auch für

die Regeln PAR-L, PAR-R, COMM-L, COMM-R, CLOSE-L und CLOSE-R, in denen parallele Komposition von Prozessen eine Rolle spielt.

Par-L, Par-R: Ähnlich der PAR Regel in der Reduktionssemantik wird durch diese Regeln eine Interaktion innerhalb einer Komponente einer parallelen Komposition ermöglicht. Allerdings muss hier als zusätzliche Nebenbedingung gewährleistet sein, dass ein in der Aktion vorkommender gebundener Name in keiner der nicht benötigten parallelen Komponenten frei vorkommen darf. Der Grund hierfür wird in der Ableitung

$$(\nu z)\overline{x}z.\overline{z}y.\mathbf{0} \mid z(u).\mathbf{0} \xrightarrow{\overline{x}\nu z} \overline{v}y.\mathbf{0} \mid z(u).\mathbf{0}\,,$$

verdeutlicht, in der zunächst die Regel PAR-L und in deren Bedingung die Regel OPEN und schließlich in deren Bedingung die Regel OUT verwendet wird. Die zusätzliche Voraussetzung $\mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset$ erzwingt eine Umbenennung des Namens z vor Anwendung von PAR-L, da z in der zweiten parallelen Komponente frei vorkommt. Ohne die Voraussetzung wäre unter Verzicht auf die Umbenennung auch die Ableitung

$$(\nu z)\overline{x}z.\overline{z}y.\mathbf{0} \mid z(u).\mathbf{0} \xrightarrow{\overline{x}\nu z} \overline{z}y.\mathbf{0} \mid z(u).\mathbf{0}$$

gültig, in der nun fälschlicherweise eine Kommunikation zwischen den beiden Komponenten über den Kanal z möglich wäre.

Comm-L, Comm-R: Wie in der Reduktionssemantik ist auch in der Transitionssemantik eine Interaktion zwischen parallelen Komponenten eines Prozesses möglich. Diese wird mit den Regeln COMM-L und COMM-R umgesetzt, indem in der einen Komponente eine Ausgabe und in der anderen eine Eingabe mit gleichem Kanal stattfindet. Die Gesamtaktion ist für fremde Prozesse nicht beobachtbar (Intraaktion) und wird daher mit der Relation $\xrightarrow{\tau}$ ausgedrückt.

Close-L, Close-R, Open: Die CLOSE-L und CLOSE-R Regeln realisieren, ähnlich wie die COMM-L und COMM-R Regeln, eine Interaktion zwischen parallelen Komponenten innerhalb eines Prozesses. Allerdings wird hier ein in der sendenden Komponente privater Name y an einen Kommunikationspartner gesandt und dabei der Gültigkeitsbereich der Restriktion erweitert (*Scope Extrusion*). Um die erste Bedingung der Regeln zu erfüllen, ist eine Anwendung der Regel OPEN erforderlich, die das Senden eines privaten Namen auf das Senden eines globalen Namen zurückführt und dabei den Restriktionsoperator zunächst verschwinden lässt (*Öffnen der Restriktion*). Die CLOSE-L bzw. CLOSE-R Regel führt den Restriktionsoperator allerdings sofort wieder ein (*Schließen der Restriktion*) und zwar nun mit beiden parallelen Komponenten als Gültigkeitsbereich. Die Ableitung

$$(\nu z)\overline{x}z.\overline{z}y.\mathbf{0} \mid x(y).y(u)\mathbf{0} \xrightarrow{\tau} (\nu z)(\overline{z}y.\mathbf{0} \mid z(u).\mathbf{0})$$

veranschaulicht dies. Für die zusätzliche Nebenbedingung $y \notin \mathrm{fn}(Q)$ in den CLOSE-L und CLOSE-R Regeln gilt die gleiche Begründung wie für die Bedingung in den PAR-L und PAR-R Regeln.

Res: Analog zur Regel RES der Reduktionssemantik erlaubt diese Regel Interaktionen innerhalb des Gültigkeitsbereichs einer Restriktion, sofern der private Name z nicht in die Aktion involviert ist. Das Kommunizieren mit einem Prozess außerhalb des Gültigkeitsbereichs über einen von z verschiedenen Namen ist ebenfalls möglich. Beides wird in der Beispielableitung

$$(\nu z)x(y).(\overline{z}y.P \mid z(v).Q) \xrightarrow{x(u)} (\nu z)(\overline{z}u.P \mid z(v).Q) \xrightarrow{\tau} (\nu z)(P \mid Q\{u/v\})$$

veranschaulicht. Die erste Transition benutzt RES und INP, während die zweite RES, COMM-L, OUT und INP verwendet.

Rep-Act, Rep-Comm, Rep-Close: Zuletzt werden noch einige Regeln benötigt, um mit Replikationen umzugehen. REP-ACT ermöglicht die Entfaltung einer Replikation, falls der replizierte Teilprozess P für eine Interaktion gemäß einer anderen Regel benötigt wird. Dies verhindert das Replizieren von unnötig vielen parallelen Kopien von P. Eine Interaktion zwischen zwei replizierten Prozessen P wird mit REP-COMM oder REP-CLOSE realisiert, wobei letztgenannte Regel analog zur CLOSE-L und CLOSE-R Regel gleichzeitig eine Scope Extrusion bewirkt.

Die größere Anzahl an Regeln macht die Transitionssemantik schwerer zu handhaben als die Reduktionssemantik. Dies gilt insbesondere für formale Beweise von Eigenschaften des Kalküls sowie der damit entwickelten Modelle. Man beachte auch, dass die binären Operatoren | und + in der Transitionssemantik nicht mehr kommutativ oder assoziativ sind. In einer Ableitung kann zwar über die Regeln PAR-L und PAR-R bzw. SUM-L und SUM-R eine beliebige Komponente ausgewählt werden, dies geschieht jedoch nicht durch syntaktische Manipulation des Prozessterms wie das Vertauschen von Komponenten oder das Verschieben von Klammern.

Der Vorteil der Transitionssemantik liegt in ihrer besseren Eignung zur Modellierung großer Systeme. Durch Aktionspräfixe, die mit externen Prozessen kommunizieren können, lässt sich eine Art Schnittstelle eines Prozesses mit seiner Umgebung implementieren.

Beispiel 3.3.2: Der Prozess

$$(\nu x)(\nu y)(z(u).([u = v_1]\overline{z}v_1.P_1 + [u = v_2]\overline{x}y.P_2) \mid x(v).Q)$$

besitzt mit dem Präfix $z(u)$ in seiner ersten parallelen Komponente eine Schnittstelle zu seiner Umgebung, über die er einen Namen, der nicht identisch mit den gebundenen Namen des Prozesses x und y ist, empfangen kann. Anschließend werden zwei Fälle unterschieden: Falls der Name v_1 empfangen wurde, so wird dieser Name über den Schnittstellenkanal zurückgesandt. (Man beachte, dass der Empfänger dieser Botschaft nicht notwendigerweise der Sender der ersten Nachricht sein muss.) Der zweite Fall tritt ein, falls der empfangene Name identisch mit v_2 ist und führt zu einer Intraaktion mit der zweiten Komponente des Prozesses. □

Das Beispiel lässt sich unter Verwendung der Reduktionssemantik nur modellieren, indem sämtliche Prozesse, die über den Kanal z mit dem Prozess aus dem Beispiel kommunizieren können, als parallele Teilprozesse hinzugefügt werden. Durch

die Möglichkeit, Interaktionspartner unspezifiziert zu lassen, und mit diesen asynchron zu kommunizieren, ist ein Prozess in der Transitionssemantik in der Lage, Verhalten aufzuzeigen, das sich in der Reduktionssemantik nicht formulieren lässt. Umgekehrt lässt sich jedoch sämtliches Verhalten in der Reduktionssemantik ausschließlich durch τ-Transitionen der Transitionssemantik ausdrücken. Das folgende Theorem, dessen Beweis in [SaWa_01] nachzulesen ist, fasst dies zusammen.

Theorem 3.3.3: $P \longrightarrow P'$ gdw. $\exists P''.P \xrightarrow{\tau} P'' \wedge P'' \equiv P'$,
wobei $\longrightarrow$ die Reduktionsrelation und $\xrightarrow{\tau}$ die τ-Transitionsrelation ist. □

Auch in der Transitionssemantik lässt sich der Replikationsoperator durch Definitionsgleichungen ersetzen. Hierzu werden die Transitionsregeln Rep-Act, Rep-Comm und Rep-Close durch folgende Regel ersetzt:

$$\textsc{Def}: \quad \frac{P\{y_1,\ldots,y_n/x_1,\ldots,x_n\} \xrightarrow{\alpha} P', \quad A(x_1,\ldots,x_n) = P}{A\langle y_1,\ldots,y_n\rangle \xrightarrow{\alpha} P'}$$

Man beachte, dass hier ähnlich wie bei den drei Regeln für die Replikation nur dann ein Ersetzen des Bezeichners möglich ist, wenn der eingesetzte Prozessterm unmittelbar für eine Aktion gebraucht wird.

Um polyadische Eingaben und Ausgaben zu ermöglichen, sind entsprechende Erweiterungen der Regeln Inp und Out sowie der Transitionsbeschriftungen nötig. Wie auch schon bei der Reduktionssemantik, stellt dies lediglich eine syntaktische Variation dar. Sämtliche Erläuterungen zu den polyadischen Präfixen aus Abschnitt 3.2 gelten analog.

In der Literatur existieren einige Arbeiten, die sich mit einer Anwendung des klassischen π-Kalküls in der Systembiologie auseinandersetzen. Zwei Arbeiten von Aviv Regev, William Silverman und Ehud Shapiro, in denen beispielhaft eine Modellierung der RTK-MAPK Signal Transduktion als π-Prozess beschrieben wird, seien hier hervorzuheben ([ReSS_01], [ReSS_00]). Der Ansatz über klassische π-Prozesse stößt jedoch schnell an eine Grenze, wie das im folgenden Abschnitt betrachtete Anwendungsbeispiel, das der Arbeit [PRSS_01] entnommen ist, zeigt.

3.4 Anwendungsbeispiel: Genexpression mit positivem Feedback

In diesem Abschnitt wollen wir versuchen, einen einfachen Vorgang aus der Zellbiologie im π-Kalkül zu modellieren. Wir werden sehen, dass die Möglichkeiten des klassischen π-Kalküls noch nicht ausreichen, um das biologische Verhalten exakt nachzubilden und damit eine Erweiterung und Anpassung des Kalküls an molekularbiologische Bedürfnisse motivieren. Der Vorgang, den wir beschreiben wollen, ist ein einfaches Beispiel von Genexpression, bei dem zwei Gene sich wechselseitig durch die von ihnen produzierten Proteine beeinflussen. Abbildung 3.1 zeigt die Interaktion der beiden Gene (Gen A und Gen TF) in der Notation, die in Abschnitt 2.1.1 eingeführt wurde. Das Gen TF produziert einen Transkriptionsfaktor (Protein TF), der die Produktion des Proteins A durch das Gen A, sowie die Produktion am Gen TF selbst, verstärkt. Der Transkriptionsfaktor muss allerdings durch das Binden an Protein A aktiviert werden. (In der Abbildung wird dies durch den Ausgabepfeil des

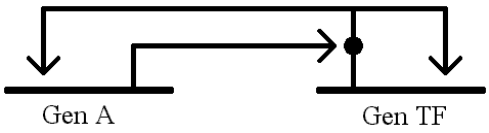


Abbildung 3.1: Genexpression mit positivem Feedback

Gens A auf den Kreis in der Ausgabe von Gen TF dargestellt.) Genauer gesagt ist die Bindungsstelle des Transkriptionsfaktors, die für das Andocken an die beiden Gene erforderlich ist, im Inneren der Proteinstruktur verborgen und wird erst durch die Wirkung von Protein A an die Oberfläche gebracht. Proteine, die derartige Aktivierungsfunktionen besitzen, werden auch als *Kinasen* bezeichnet. In Abbildung 3.2 ist die Aktivierung des Proteins TF und die anschließende Regulierung der Proteinproduktion in der Notation aus Abschnitt 2.1.2 dargestellt. Die Evolution des

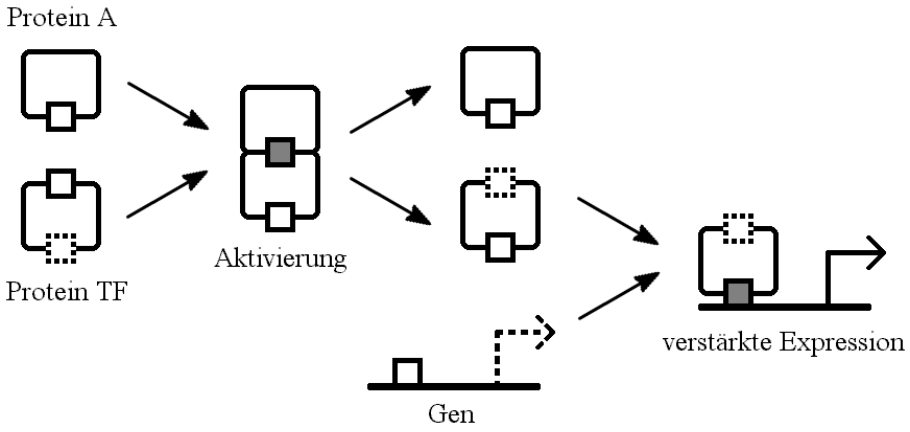


Abbildung 3.2: Aktivierung des Proteins TF durch das Protein A und anschließende positive Regulierung der Genexpression

Systems lässt sich folgendermaßen beschreiben: Zunächst produzieren beide Gene mit geringer Ausgaberate Proteine. Nach einer gewissen Zeit kommt es zur Aktivierung der Transkriptionsfaktoren (Protein TF) durch die Kinasen (Protein A). Die aktivierten Transkriptionsfaktoren docken an beide Gene an und realisieren damit eine Rückkopplung der Ausgabe (positives Feedback).

Nachfolgend ist ein System im polyadischen π-Kalkül unter Verwendung der Transitionssemantik angegeben, das als Modellskizze des beschriebenen Vorgangs dienen soll. Wir verwenden dabei der Übersichtlichkeit halber Definitionsgleichungen statt des Replikationsoperators. Des Weiteren weichen wir von unserer Namenskonvention ab und interpretieren nun jedes Wort in Kleinbuchstaben als Namen und jedes Wort in Großbuchstaben als Prozessbezeichner. Die Modellierung soll so detailliert wie möglich erfolgen, daher wird die Genexpression in mehreren Schritten vollzogen. Die Gene werden in regelmäßigen Abständen in RNA-Moleküle übersetzt (Transkription), die anschließend durch die Zellmaschinerie in Proteine übersetzt werden können (Translation). Sowohl die RNA-Molküle wie auch die produzierten Proteine sind Zerfallsprozessen unterlegen, die ebenfalls im Modell Berücksichtigung finden.

$$
\begin{aligned}
\textsc{Sys} &= (\nu\, tf^-)(\nu\, tf^+)(\nu\, trans)(\nu\, zerf_r)(\nu\, zerf_p)(\nu\, aktiv)(\nu\, binden)\\
&\quad (\textsc{GenA} \mid \textsc{GenTF} \mid \textsc{Transkript} \mid \textsc{Translat} \mid \textsc{RnaZerf} \mid\\
&\quad \textsc{ProtZerf})\\[1ex]
\textsc{GenA} &= tf^-().(\textsc{GenA} \mid \textsc{RnaA}) + tf^+().(\textsc{GenA} \mid \textsc{RnaA})\\
\textsc{RnaA} &= trans().(\textsc{RnaA} \mid \textsc{ProteinA}) + zerf_r().\mathbf{0}\\
\textsc{ProteinA} &= (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\textsc{BindungA} \mid \textsc{Kinase})\\
\textsc{BindungA} &= \overline{binden}(bb_1, bb_2, bb_3).\textsc{GebundenA} + zerf_p().\overline{bb_3}().\mathbf{0}\\
\textsc{GebundenA} &= \overline{bb_1}().\textsc{BindungA} + zerf_p().\overline{bb_3}().\overline{bb_3}().\mathbf{0}\\
\textsc{Kinase} &= \overline{bb_2}(aktiv).\textsc{Kinase} + bb_3().\mathbf{0}\\[1ex]
\textsc{GenTF} &= tf^-().(\textsc{GenTF} \mid \textsc{RnaTF}) + tf^+().(\textsc{GenTF} \mid \textsc{RnaTF})\\
\textsc{RnaTF} &= trans().(\textsc{RnaTF} \mid \textsc{ProteinTF}) + zerf_r().\mathbf{0}\\
\textsc{ProteinTF} &= binden(bb'_1, bb'_2, bb'_3).\textsc{GebundenTF} + zerf_p().\mathbf{0}\\
\textsc{GebundenTF} &= bb'_1().\textsc{ProteinTF} + bb'_2(aktiv').bb'_1().\textsc{AktivTF}(aktiv') +\\
&\quad bb'_3().\mathbf{0}\\
\textsc{AktivTF}(a) &= \overline{a}().\textsc{AktivTF}(a) + zerf_p().\mathbf{0}\\[1ex]
\textsc{Transkript} &= \overline{tf^-}().\textsc{Transkript} + aktiv().\overline{tf^+}().\textsc{Transkript}\\
\textsc{Translat} &= \overline{trans}().\textsc{Translat}\\
\textsc{RnaZerf} &= \overline{zerf_r}().\textsc{RnaZerf}\\
\textsc{ProtZerf} &= \overline{zerf_p}().\textsc{ProtZerf}
\end{aligned}
$$

In [PRSS_01] wird das gleiche System in einer stochastischen Variante auf Basis einer Reduktionssemantik angegeben. Der initiale Prozess Sys besteht aus sechs parallelen Komponenten, von denen die ersten beiden die Gene realisieren und damit materielle Entitäten des Systems darstellen. Die anderen vier Komponenten modellieren die molekularbiologische Maschinerie, die Transkriptions-, Translations- und Zerfallsvorgänge einleitet, und sind daher nicht direkt mit materiellen Objekten der Anschauung verknüpft. Prozesse dieser Gestalt wollen wir im Folgenden als *Umweltprozesse* bezeichnen. Der Prozess Transkript sendet beispielsweise in regelmäßigen Abständen eine leere Botschaft (nullstellige Ausgabe) über den Kanal tf^-, die von einem der beiden Genprozesse empfangen werden kann und zur Spaltung des jeweilgen Prozesses in zwei parallele Komponenten führt, von denen die erste das unveränderte Gen und die zweite einen generierten RNA-Strang (Prozess RnaA bzw. RnaTF) darstellt. Alternativ kann der Prozess Transkript die leere Botschaft über Kanal *aktiv* empfangen und anschließend die Transkription eines der Gene über den Kanal tf^+ anregen. Damit wird die Wirkung des aktivierten Transkriptionsfaktors AktivTF realisiert, der über den Kanal *aktiv* sendet. Analog zu Transkript regt der Prozess Translat die Translation von der RNA zu den Proteinen an, die durch die Prozesse ProteinA und ProteinTF beschrieben werden. Die beiden Prozesse RnaZerf und ProtZerf senden regelmäßig Botschaften an die verschiedenen RNA- und Protein-Prozesse und bewirken damit deren Termination, wodurch der Zerfall der jeweiligen Makromoleküle realisiert wird.

Jeder Prozess ProteinA definiert drei private Namen bb_1, bb_2 und bb_3, die sein Kommunikationsrückgrat (*backbone*) darstellen und jeweils eine spezifische Bedeu-

tung besitzen. Darüber hinaus besteht ein ProteinA-Prozess aus zwei parallelen Komponenten BindungA und Kinase, welche die Bindungsmöglichkeit des Proteins mit Transkriptionsfaktoren bzw. die Funktion als Kinase beschreiben. Der Prozess BindungA kann die drei backbone-Kanäle über den Kanal *binden* an einen ProteinTF-Prozess senden und damit verschiedene Interaktionsmöglichkeiten eröffnen. Die molekularbiologische Interpretation dieses Vorgangs ist die Komplexbildung zwischen den beiden Proteinen. In den jeweiligen Prozessen findet dabei ein Übergang zu GebundenA bzw. GebundenTF statt. Nun können die backbone-Kanäle in Aktion treten. Über bb_1 können die beiden gebundenen Proteine synchron eine Trennung und damit die Rückkehr zu den Prozessen BindungA und ProteinTF einleiten. Der Kanal bb_2 dient der Aktivierung des Transkriptionsfaktors durch die Kinase. Nach einer entsprechenden Interaktion, bei der die Kinase den Namen *aktiv* an das gebundene Protein TF sendet, findet eine synchrone Trennung des Komplexes über bb_1 statt und der Prozess GebundenTF geht in AktivTF über. AktivTF kann nun den zuvor erhaltenen Namen *aktiv* für eine Kommunikation mit dem Umweltprozess Transkript nutzen und damit die Feedback-Schleife schließen. Der Kanal bb_3 dient der Synchronisation der Zerfallsvorgänge der Prozesse, die an dem Komplex beteiligt sind. Der Zerfall wird stets zwischen dem ProtZerf- und dem GebundenA-Prozess über Kanal $zerf_p$ vereinbart und dann über bb_3 an Kinase und GebundenTF weitergereicht, so dass alle Komponenten gleichzeitig terminieren. Auch im ungebundenen Protein A synchronisieren die beiden Komponenten BindungA und Kinase ihren Zerfall über bb_3.

Sämtliche Namen des Systems, die nicht bereits in einem Teilprozess durch Restriktionen oder Eingabe-Präfixe eingeführt werden, sind in Sys explizit durch Restriktionen von der Einwirkung äußerer Faktoren geschützt. Diese Einschränkung dient hier lediglich der Vereinfachung. Um beispielsweise die Möglichkeit der Bindung des Proteins A an beliebige andere Makromoleküle zu erlauben, würde es ausreichen, die Restriktion $(\nu\, binden)$ aus der Prozessdefinition zu entfernen. Der Prozess BindungA könnte dann seine backbone-Kanäle auch an externe Prozesse senden.

Problematik des Modells: Wie am Anfang dieses Abschnittes bereits erwähnt, ist diese Modellierung keine exakte Nachbildung des gewünschten Systemverhaltens. Tatsächlich wird der wichtigste Aspekt des Anwendungsbeispiels, nämlich die verstärkte Genexpression bei Anwesenheit des aktivierten Transkriptionsfaktors, nicht umgesetzt. Es wird zwar qualitativ ausgedrückt, dass zwei verschiedene Möglichkeiten der Transkription existieren (Kanäle tf^- und tf^+ in den Prozessen Transkript und GenA bzw. GenTF), allerdings führt dies in einer Ableitung nicht zwangsläufig zu einer größeren Anzahl generierter RNA-Prozesse. Im klassischen π-Kalkül wird zwischen Aktionen, die um die gleiche Ressource konkurrieren, nichtdeterministisch ausgewählt, d.h. es spielt für die Ableitung keine Rolle, wie viele gleichartige (oder ähnliche) Aktionsmöglichkeiten ein Prozess aufweist. Ein externer Beobachter stellt zwar die Kommunikation über den Kanal tf^+ nach der Aktivierung eines Transkriptionsfaktors fest, kann aber keine Veränderung der Auftrittsrate von Transkriptionen damit in Verbindung bringen.

Auch an anderen Stellen des Modells fehlt eine quantitative Beschreibung der Gegebenheiten. Beispielsweise soll der Zerfall eines Proteinkomplexes, der nach seiner Einleitung durch den Prozess ProtZerf von BindungA mittels Kanal bb_3

den übrigen Prozessen des Komplexes mitgeteilt wird, nicht durch andere eventuell zeitverbrauchende Aktionen unterbrochen werden können. Wünschenswert wäre also eine ausschließliche Verarbeitung von Kommunikation mittels des Kanals bb_3, sobald ein Zerfall in Gang gesetzt wurde.

Darüber hinaus wird die Dauer von Aktionen in der aktuellen Modellierung nicht berücksichtigt. Beispielsweise dauert ein Translationsvorgang in der Regel länger als die Aktivierung eines Transkriptionsfaktors durch eine Kinase. Um statistische Aussagen über die Evolution eines Systems machen zu können, ist eine Umsetzung dieser Information unerlässlich.

Die angeführten Probleme motivieren eine Erweiterung des π-Kalküls, die auch die quantitativen Aspekte des Systemverhaltens mit einbezieht. Das folgende Kapitel stellt mit dem stochastischen π-Kalkül eine mögliche Erweiterung vor, die zu diesem Zweck Konzepte aus der Wahrscheinlichkeitstheorie ausnutzt und daher eine vielversprechende Annäherung an die Methodik von Simulatoren darstellt.

Kapitel 4

Stochastischer π-Kalkül

Der von Corrado Priami vorgestellte stochastische π-Kalkül ([Pri_95]) stellt eine Erweiterung des im vorangehenden Abschnitt eingeführten klassischen π-Kalküls dar, die sich speziell der Modellierung der quantitativen Aspekte eines Systems widmet. Durch das Hinzufügen von Parametern mathematischer Verteilungen zu den Aktionspräfixen wird eine Aktion mit einem zeitlichen Verhalten verknüpft. Neben diesem *stochastischen Ansatz* gibt es auch noch zwei weitere Konzepte für quantitative Modellierungen, die in Erweiterungen anderer Prozesskalküle zum Einsatz kommen. *Probabilistische Prozesskalküle* weisen jeder möglichen Transition in einer Ableitung explizit eine Wahrscheinlichkeit zu. Die mit diesem Ansatz entworfenen Kalküle sind jedoch stets auf synchrone Aktionen beschränkt, während der π-Kalkül mit der Transitionssemantik auch asynchrones Verhalten realisieren kann. Eine weitere Idee ist durch die *temporalen Prozesskalküle* gegeben, die einer Aktion in Abhängigkeit der verwendeten Namen explizit eine Dauer zuweisen. Auch dieser Ansatz kommt für unser Vorhaben nicht in Frage, da die Dauer von biologischen Prozessen von einer Vielzahl an Faktoren abhängen kann wie beispielsweise der Verfügbarkeit von benötigten Ressourcen.

Der stochastische π-Kalkül (kurz: Sπ) wurde ursprünglich nicht zur Modellierung biologischer Systeme entwickelt, sondern sollte eine Performanceanalyse technischer Systeme zur Entwurfszeit ermöglichen. In unserer Anwendung wird der Kalkül allerdings nicht zum Entwurf neuer, sondern zur Beschreibung und Analyse bestehender Systeme eingesetzt. Wir werden sehen, dass der stochastische π-Kalkül nach Priami unseren Anforderungen noch nicht vollkommen genügt. Aus diesem Grund wird im Anschluss an den folgenden Abschnitt 4.1, der den Kalkül in Syntax und Semantik vorstellt, in Abschnitt 4.2 eine Variante des Kalküls diskutiert, die wesentlich besser auf die Anwendung in der Systembiologie zugeschnitten ist. Abschließend wird in Abschnitt 4.3 das Genexpressions-Anwendungsbeispiel des vorangehenden Abschnitts fortgesetzt.

4.1 Syntax und Semantik

Die für die stochastische Erweiterung erforderlichen syntaktischen Anpassungen des π-Kalküls sind gering. Die Aktionspräfixe werden lediglich um einen Parameter ergänzt, der eine Exponentialverteilung charakterisiert. Die Vorstellung hierbei ist, dass Aktionen jeweils eine Dauer besitzen, die gemäß einer mathematischen Vertei-

lung zufällig ermittelt wird. Die Wahl der Exponentialverteilung lässt sich mit der Eigenschaft der *Gedächnislosigkeit* dieser Verteilung begründen. Für das Auftreten einer Aktion ist die Kenntnis des Zeitpunktes des letzten Auftretens der gleichen Aktion nicht erforderlich. Dies ermöglicht eine einfache Implementierung in Simulatoren und gewährleistet eine gute Performance. Ein weiterer Grund ist das häufige Vorkommen von Prozessen mit exponentiell verteilter Dauer in natürlichen wie auch technischen Systemen. Wir bezeichnen eine Aktion in Verbindung mit einem Parameter zur Charakterisierung ihres zeitlichen Verhaltens als *Aktivität*. Die zuvor eingeführten Konventionen zur Benennung von Namen und Prozessen behalten wir bei und definieren die Syntax des stochastischen π-Kalküls wie folgt.

Definition 4.1.1 (Sπ-Syntax): Die Menge der *Sπ-Prozesse* ist durch folgende Grammatik gegeben:

$$P ::= \mathbf{0} \quad | \quad \pi.P \quad | \quad (\nu x)P \quad | \quad [x = y]P \quad | \quad !P \quad | \quad P|P \quad | \quad P + P$$

wobei π eine Aktivität gemäß folgender Definition darstellt:

$$\pi ::= \overline{x}y[r] \quad | \quad x(y)[r] \quad | \quad \tau[r]$$

Die *Aktionsrate* $r \in \mathbb{R}^+$ ist der Parameter einer Exponentialverteilung. □

Die Verteilungsfunktion der Exponentialverteilung

$$F_{Exp}(t) = 1 - e^{-rt}$$

ordnet einer Dauer (für die Ausführung einer Aktivität) abhängig vom jeweiligen Parameter r eine Wahrscheinlichkeit zu. Somit lässt sich die Dauer einer Aktivität über die Inverse der Verteilungsfunktion zufällig ermitteln

$$t = \frac{\ln(1 - F_{Exp})}{-r} ,$$

wobei für F_{Exp} eine gleichverteilte Zufallsvariable im Interval $[0, 1)$ eingesetzt wird. Da gleichverteilte Zufallsvariablen sehr einfach mittels linearer Kongruenzgeneratoren zu ermitteln sind, lässt sich dieser Ansatz gut in Simulationssoftware umsetzen.

Während im klassischen π-Kalkül nichtdeterministisch zwischen mehreren möglichen Aktionen eines Prozesses ausgewählt wird, gibt es im stochastischen π-Kalkül eine sogenannte *Wettlaufbedingung*, die ein genaues Verfahren zur Auswahl des nächsten auszuführenden Schrittes angibt. Für jede mögliche Transition, das heißt für jede mögliche Interaktion oder Intraaktion eines Prozesses, wird eine Rate berechnet, über die mit der oben vorgestellten Inversionsmethode eine Dauer ermittelt werden kann. Die schnellste Transition, also diejenige mit der kürzesten Dauer, gewinnt das Rennen und der zugehörige Ableitungsschritt wird vollzogen. Die übrigen ermittelten Transitionsdauern sind damit belanglos. Für den resultierenden Prozessterm wird das Verfahren erneut gestartet. Das Verwerfen der Dauer einer nicht erfolgreichen Transition spiegelt die Gedächtnislosigkeit der Exponentialverteilung wieder.

Beispiel 4.1.2: Man betrachte den Prozess

$$P_1 = (x(y)[5].x(z)[7].\mathbf{0} + x(y)[3].\mathbf{0}) \mid \overline{x}u[6].\mathbf{0} ,$$

in dem fünf verschiedene Transitionen möglich sind. Die erste parallele Komponente kann einen Namen, der für den Platzhalter y eingesetzt wird, über eine Interaktion mit einem externen Prozess empfangen und zwar entweder mit Rate 5 (erste Alternative) oder mit Rate 3 (zweite Alternative). Analog kann die zweite parallele Komponente den Namen u an einen externen Prozess versenden. Zwei weitere Möglichkeiten bestehen in Intraaktionen der beiden parallelen Komponenten über den gemeinsamen Namen x. Hierbei ist wieder eine der beiden Alternativen der ersten Komponente als Empfänger auszuwählen. Während für die ersten drei Alternativen eine Aktionsrate explizit mit der Aktivität gegeben ist, muss die Rate für die beiden Intraaktionen aus den Raten der beteiligten Aktivitäten ermittelt werden.□

Um die Raten von Intraaktionen berechnen zu können, führen wir zunächst einige Begriffe ein. Die *Ausgangsrate* (*exit rate*) eines Prozesses $er(P)$ ist definiert als die Summe der Raten aller möglichen Transitionen eines Prozesses:

$$er(P) = \sum_{P \xrightarrow{\mu_i[r_i]} P_i \in T(P)} r_i$$

wobei μ_i das Label einer Transition und $T(P)$ die Menge der möglichen Transitionen von P ist. Wir werden die Menge möglicher Transitionen sowie ihre Label später im Zusammenhang mit den Transitionsregeln genau definieren, im Moment interessiert uns aber nur die Berechnung von Raten für Synchronisationen zwischen Eingabe- und Ausgabeaktivitäten. Die *scheinbare Rate* (*apparent rate*) einer Aktion a in einem Prozess P ist definiert als

$$r_a(P) = \sum_{P \xrightarrow{a[r_i]} P_i \in T(P)} r_i \ .$$

$r_a(P)$ ist die Summe der Raten aller ungeschützten Vorkommen der Aktion $a \in \{\overline{x}y, \overline{x}\nu y, x(y), \tau\}$ in einem Prozess. Dies ist die Aktionsrate, die ein externer Beobachter wahrnimmt, der zwar beispielsweise das Senden eines Namens y über einen Kanal x feststellt, aber nicht den Prozessterm kennt und daher auch nicht in der Lage ist, den sendenden Teilprozess zu identifizieren.

Beispiel 4.1.3: In unserem Beispielprozess P_1 ist die scheinbare Rate der Aktion $\overline{x}u$ 6, denn die Aktion kommt nur einmal mit Aktionsrate 6 vor. Die scheinbare Rate der zweifach vorkommenden Aktion $x(y)$ ist $5 + 3 = 8$ während die scheinbare Rate der Aktion $x(z)$ nicht definiert (bzw. 0) ist, da diese Aktion aufgrund des vorangehenden Präfixes zu keiner unmittelbaren Transition führen kann. □

Die scheinbare Rate einer Intraaktion ist definiert als das Minimum der scheinbaren Raten der beteiligten Aktionen in ihren jeweiligen Teilprozessen:

$$r_{a,\overline{a}}(P|Q) = \min(r_a(P), r_{\overline{a}}(Q))$$

mit $a = x(z)$ und $\overline{a} = \overline{x}y$. (Für den Fall, dass der sendende Teilprozess die erste parallele Komponente und der empfangende Prozess die zweite Komponente darstellt, gilt eine symmetrische Variante der Formel.) Die Minimumbildung spiegelt

die Bedeutung der langsameren Komponente für die Intraaktion wieder. Der schnellere Teilprozess muss gegebenenfalls auf den langsameren warten, um die Transition abzuschließen.

Mit Hilfe dieser Definitionen lassen sich Aussagen über die Wahrscheinlichkeiten von bestimmten Aktionen gewinnen. Sei $P \xrightarrow{\mu_i[r_i]} P_i \in T(P)$. Die *Auftrittswahrscheinlichkeit* dieser Transition ist wie folgt definiert:

$$p = \frac{r_i}{er(P)} .$$

Ist bereits bekannt, dass eine Aktion a aufgetreten ist, so lässt sich die *bedingte Auftrittswahrscheinlichkeit* einer Transition $P \xrightarrow{a[r_i]} P_i \in T(P)$ folgendermaßen berechnen:

$$p_a = \frac{r_i}{r_a(P)} .$$

Geht man davon aus, dass bei einer Intraaktion die beiden involvierten Teilprozesse unabhängig voneinander entscheiden, welche Aktionen ausgewählt werden, so erhält man für die *Auftrittswahrscheinlichkeit einer Intraaktion* folgende Formel:

$$p_{a,\overline{a}} = \frac{r_1}{r_a(P)} * \frac{r_2}{r_{\overline{a}}(Q)} ,$$

wobei r_1 die Aktionsrate der empfangenden Aktion a, r_2 die Aktionsrate der sendenden Aktion $\overline{a}$ und P bzw. Q die jeweiligen an der Intraaktion beteiligten Teilprozesse sind. Um die *Rate einer Intraaktion* zu ermitteln, stellen wir die Formel für die bedingte Auftrittswahrscheinlichkeit (die auch bei Intraaktionen anwendbar ist) nach der Aktionsrate um und setzen die Intraaktionswahrscheinlichkeit $p_{a,\overline{a}}$ und die scheinbare Rate $r_{a,\overline{a}}(P|Q)$ der Intraaktion ein:

$$R_{a,\overline{a}}(P, Q, r_1, r_2) = \frac{r_1}{r_a(P)} * \frac{r_2}{r_{\overline{a}}(Q)} * \min(r_a(P),\, r_{\overline{a}}(Q)) .$$

Beispiel 4.1.4: Wir betrachten erneut den Prozess P_1 aus den vorangegangenen zwei Beispielen. Mit

$$P = x(y)[5].x(z)[7].\mathbf{0} + x(y)[3].\mathbf{0} \text{ und } Q = \overline{x}u[6].\mathbf{0}$$

ergibt sich für die Intraaktion, bei der die erste Alternative in P gewählt wird

$$R_{x(y),\overline{x}u}(P, Q, 5, 6) = \frac{5}{8} * \frac{6}{6} * \min(8, 6) = 3{,}75$$

und für die Auswahl der zweiten Alternative in P

$$R_{x(y),\overline{x}u}(P, Q, 3, 6) = \frac{3}{8} * \frac{6}{6} * \min(8, 6) = 2{,}25 .$$

Die Ausgangsrate des Prozesses ist $er(P_1) = 5 + 3 + 6 + 3,75 + 2,25 = 20$. Damit ergeben sich Auftrittswahrscheinlichkeiten für die Interaktionen mit externen Prozessen über die Aktivitäten $x(y)[5]$, $x(y)[3]$ und $\overline{x}u[6]$ von 25%, 15% bzw. 30% und für die Intraaktionen $(x(y)[5], \overline{x}u[6])$ und $(x(y)[3], \overline{x}u[6])$ von 18,75% bzw. 11,25%. □

An dieser Stelle ist eine wichtige Anmerkung zu den Eingabeaktionen angebracht. Bei einer Transition der Form $x(y).P \xrightarrow{x(z)} P\{z/y\}$, die eine freie Eingabe durch einen externen Prozess darstellt, kann die Wahl des zu empfangenden Namens z eine wesentliche Bedeutung für die weitere Evolution des Prozesses P spielen. Ein einzelnes Präfix $x(y)$ kann daher mehrere Transitionen ermöglichen und damit die Ausgangsrate des Prozesses stärker beeinflussen als eine Ausgabeaktion. Im Prozess P_1 der vorangehenden Beispiele wird der empfangene Name im weiteren Verlauf nicht mehr verwendet und somit die Rate der Eingabeaktionen nicht mehrfach in der Berechnung der Ausgangsrate berücksichtigt. In folgendem Beispiel spielt der empfangene Name eine größere Rolle.

Beispiel 4.1.5: In dem Prozess

$$P_2 = (\nu v)(x(y)[3].y(u)[5].\mathbf{0} \mid \overline{z}v[6].\mathbf{0})$$

kann der Empfang eines Namens y zwei verschiedene Folgeverhalten des Prozesses hervorrufen. Wird der Name z für y eingesetzt, so ist im Anschluss eine Intraaktion zwischen den beiden parallelen Komponenten über z möglich. Für den Empfang eines beliebigen anderen Namens sind lediglich weitere asynchrone Interaktionen mit externen Prozessen möglich. Weiterhin macht es im zweiten Fall einen Unterschied, ob ein neuer, in P_2 nicht vorkommender Name, oder der einzige frei vorkommende Name x empfangen wird. Die Evolution des Prozesses selbst hängt zwar nicht davon ab, aber ein externen Beobachter nimmt zwei verschiedene Transitionsbeschriftungen für die Aktion war, die durch das zweite Präfix in der ersten Komponente des Prozesses eingeleitet wird. Hier wird entweder über den Kanal x oder einen beliebigen neuen Kanal empfangen. In der Regel wird mit einem Namen eine bestimmte Bedeutung verknüpft, so dass ein Beobachter die beiden Aktionen unterschiedlich interpretieren würde. Insgesamt ermöglicht das Präfix $x(y)$ also drei Transitionen und die Ausgangsrate des Prozesses ergibt sich zu $er(P) = 3 + 3 + 3 + 6 = 15$. □

Die Anzahl der möglichen Transitionen, die sich durch eine freie Eingabeaktion $x(y)$ in einem Prozess P ergeben, wird durch die Kardinalität der Menge der *aktiven Namen*

$$\mathrm{AN}_{x(y)}(P) = \{z \mid z \in \mathrm{fn}(P) \wedge \exists P \xrightarrow{s}{}^{*}\ P' \in \mathrm{TS}_z(P).P \xrightarrow{s}{}^{*}\ P' \notin \mathrm{TS}_{\mathrm{new}}(P)\}$$

bestimmt, wobei $\longrightarrow^*$ die reflexive und transitive Hülle der Transitionsrelation und s eine beliebige Sequenz von Labeln mit dazugehörigen Raten $\mu_i[r_i]$ $(i \geq 0)$ ist. $\mathrm{TS}_{\mathrm{new}}(P)$ ist die Menge der Sequenzen von Transitionen $P \xrightarrow{\mu_1[r_1]} P_1 \xrightarrow{\mu_2[r_2]} \ldots \xrightarrow{\mu_n[r_n]} P_n$, die von P ausgehend unter der Bedingung, dass in der freien Eingabe $x(y)$ für y ein neuer, in P nicht vorkommender Name eingesetzt wird, möglich sind. In $\mathrm{TS}_z(P)$ ist für freie Eingaben außerdem der Empfang des Namens z, der in P frei vorkommt, gestattet. Die tatsächliche Anzahl möglicher Eingabenamen einer Aktion $x(y)$ ist durch $|\mathrm{AN}_{x(y)}(P)| + 1$ gegeben. Um das unnötige Vergrößern der Ausgaberate eines Prozesses zu vermeiden, sollte in einer Modellierung wenn möglich jeder vorkommende Name durch eine Restriktion geschützt werden. Verzicht auf Restriktionen in Verbindung mit Eingabepräfixen spiegelt einen außerordentlich großen Einfluß von externen Quellen auf das interne Verhalten eines Prozesses wider, der in realen Systemen üblicherweise nicht vorhanden ist.

Eine weitere Anmerkung betrifft Prozesse, in denen ein Aktionspräfix mehrfach vorkommt. Die Prozesse $\overline{x}y[r].\mathbf{0} + \overline{x}y[r].\mathbf{0}$ und $\overline{x}y[r].\mathbf{0}$, die im klassischen π-Kalkül äquivalentes Verhalten aufweisen, sollen in der stochastischen Erweiterung durchaus unterschieden werden. Tatsächlich hat die Aktion $\overline{x}y$ im ersten Prozess eine doppelt so große scheinbare Rate wie im zweiten Prozess. Um das häufigere Auftreten der Aktion umzusetzen, müssen die beiden möglichen Transitionen auch in der Ausgaberate Berücksichtigung finden. Die Unterscheidung gelingt mit der gewohnten Beschriftung von Transitionen nicht, denn welche der beiden Alternativen im ersten Prozess auch immer gewählt wird, die Transition trägt stets das Label $\overline{x}y$. Wir ergänzen daher die Label (oder Beschriftungen) von Transitionen um eine *Route*, die Informationen über die Auswahl von Komponenten in parallelen Kompositionen wie auch in Alternativen enthält. Bei Verzweigung in die erste Komponente wird die Route (ein Wort über dem Alphabet $\{1, 2\}$) um eine 1, bei Verzweigung in die zweite Komponente um eine 2 ergänzt. Außerdem ersetzen wir das Label τ bei einer nicht beobachtbaren Intraaktion durch eine genaue Beschreibung der kommunizierenden Komponenten $\langle \theta_1 a, \theta_2 \overline{a} \rangle$, wobei θ_1 und θ_2 Routen innerhalb der kommunizierenden Teilprozesse und a, $\overline{a}$ (die natürlich auch in umgekehrter Reihenfolge auftreten können) Eingabe- bzw. Ausgabepräfixe mit gemeinsamem Kommunikationskanal sind.

Beispiel 4.1.6: Wir betrachten den Prozess

$$P_3 = \overline{x}y[r_1].\mathbf{0} \mid (\nu z)((z(u)[r_2].\overline{z}u[r_3].\mathbf{0} + \overline{x}z[r_4].\mathbf{0}) \mid \overline{z}y[r_5].\mathbf{0}),$$

der folgende Transitionen besitzt

$$\begin{array}{lll} P_3 & \xrightarrow{1\overline{x}y[r_1]} & \mathbf{0} \mid (\nu z)((z(u)[r_2].\overline{z}u[r_3].\mathbf{0} + \overline{x}z[r_4]\mathbf{0}) \mid \overline{z}y[r_5].\mathbf{0}) \\ P_3 & \xrightarrow{2\langle 1z(u), \overline{z}y\rangle[r']} & \overline{x}y[r_1].\mathbf{0} \mid (\nu z)(\overline{z}y[r_3].\mathbf{0} \mid \mathbf{0}) \\ P_3 & \xrightarrow{212\overline{x}\nu z[r_4]} & \overline{x}y[r_1].\mathbf{0} \mid (\nu z)(\mathbf{0} \mid \overline{z}y[r_5].\mathbf{0}) \end{array}$$

In der ersten Transition wurde eine Interaktion über das Ausgabepräfix der ersten parallelen Komponente ausgeführt. Die einzige mögliche Intraaktion wurde in der zweiten Transition vollzogen (es gilt: $r' = \min(r_2, r_5)$), während die dritte Transition die gebundene Ausgabe des Namens z in der zweiten Alternative der mittleren parallelen Komponente ausführt. □

Wir führen nun die Semantik des stochastischen π-Kalküls anhand der um Raten und Routen erweiterten Transitionsregeln formal ein. Hierbei gilt in gleicher Weise wie beim klassischen π-Kalkül, dass Namenskonflikte zwischen gebundenen und freien Vorkommen von Namen insbesondere bei Anwendung der Regel OPEN mittels α-Konversion zu vermeiden sind.

Definition 4.1.7 (Transition): Ein Sπ-Prozess P kann genau dann zu einem Sπ-Prozess P' evolvieren, wenn sich eine *Transitionsableitung* der Form $P \xrightarrow{\mu_1[r_1]} \ldots \xrightarrow{\mu_n[r_n]} P'$ mit einer endlichen Anzahl von Anwendungen der nachfolgend aufgeführten Regeln ableiten läßt.

$$\text{OUT:} \quad \frac{}{\overline{x}y[r].P \xrightarrow{\overline{x}y[r]} P} \qquad\qquad \text{INP:} \quad \frac{}{x(z)[r].P \xrightarrow{x(y)[r]} P\{y/z\}}$$

$$\textsc{Tau:}\ \frac{}{\tau[r].P \xrightarrow{\tau[r]} P} \qquad \textsc{Match:}\ \frac{P \xrightarrow{\mu[r]} P'}{[x=x]P \xrightarrow{\mu[r]} P'}$$

$$\textsc{Sum-L:}\ \frac{P \xrightarrow{\mu[r]} P'}{P+Q \xrightarrow{1\mu[r]} P'} \qquad \textsc{Sum-R:}\ \frac{P \xrightarrow{\mu[r]} P'}{Q+P \xrightarrow{2\mu[r]} P'}$$

$$\textsc{Par-L:}\ \frac{P \xrightarrow{\mu[r]} P',\ \mathrm{bn}(\mu)\cap\mathrm{fn}(Q)=\emptyset}{P\,|\,Q \xrightarrow{1\mu[r]} P'\,|\,Q} \qquad \textsc{Par-R:}\ \frac{P \xrightarrow{\mu[r]} P',\ \mathrm{bn}(\mu)\cap\mathrm{fn}(Q)=\emptyset}{Q\,|\,P \xrightarrow{2\mu[r]} Q\,|\,P'}$$

$$\textsc{Comm-L:}\ \frac{P \xrightarrow{\theta_1\overline{x}y[r_1]} P',\quad Q \xrightarrow{\theta_2 x(y)[r_2]} Q'}{P\,|\,Q \xrightarrow{\langle\theta_1\overline{x}y,\,\theta_2 x(y)\rangle[R_{x(y),\,\overline{x}y}(Q,P,r_2,r_1)]} P'\,|\,Q'}$$

$$\textsc{Comm-R:}\ \frac{P \xrightarrow{\theta_1\overline{x}y[r_1]} P',\quad Q \xrightarrow{\theta_2 x(y)[r_2]} Q'}{Q\,|\,P \xrightarrow{\langle\theta_2 x(y),\,\theta_1\overline{x}y\rangle[R_{x(y),\,\overline{x}y}(Q,P,r_2,r_1)]} Q'\,|\,P'}$$

$$\textsc{Close-L:}\ \frac{P \xrightarrow{\theta_1\overline{x}\nu y[r_1]} P',\quad Q \xrightarrow{\theta_2 x(y)[r_2]} Q',\quad y\notin\mathrm{fn}(Q)}{P\,|\,Q \xrightarrow{\langle\theta_1\overline{x}y,\,\theta_2 x(y)\rangle[R_{x(y),\,\overline{x}\nu y}(Q,P,r_2,r_1)]} (\nu y)(P'\,|\,Q')}$$

$$\textsc{Close-R:}\ \frac{P \xrightarrow{\theta_1\overline{x}\nu y[r_1]} P',\quad Q \xrightarrow{\theta_2 x(y)[r_2]} Q',\quad y\notin\mathrm{fn}(Q)}{Q\,|\,P \xrightarrow{\langle\theta_2 x(y),\,\theta_1\overline{x}y\rangle[R_{x(y),\,\overline{x}\nu y}(Q,P,r_2,r_1)]} (\nu y)(Q'\,|\,P')}$$

$$\textsc{Res:}\ \frac{P \xrightarrow{\mu[r]} P',\quad z\notin\mathrm{fn}(\mu)\cup\mathrm{bn}(\mu)}{(\nu z)P \xrightarrow{\mu[r]} (\nu z)P'} \qquad \textsc{Open:}\ \frac{P \xrightarrow{\theta\overline{x}y[r]} P',\quad y\neq x}{(\nu y)P \xrightarrow{\theta\overline{x}\nu y[r]} P'}$$

$$\textsc{Rep-Act:}\ \frac{P \xrightarrow{\mu[r]} P'}{!P \xrightarrow{\mu[r]} P'\,|\,!P}$$

$$\textsc{Rep-Comm:}\ \frac{P \xrightarrow{\theta_1\overline{x}y[r_1]} P',\quad P \xrightarrow{\theta_2 x(y)[r_2]} P''}{!P \xrightarrow{\langle\theta_1\overline{x}y,\,\theta_2 x(y)\rangle[R_{x(y),\,\overline{x}y}(P,P,r_2,r_1)]} (P'\,|\,P'')\,|\,!P}$$

$$\textsc{Rep-Close:}\ \frac{P \xrightarrow{\theta_1\overline{x}\nu y[r_1]} P',\quad P \xrightarrow{\theta_2 x(y)[r_2]} P'',\quad y\notin\mathrm{fn}(Q)}{!P \xrightarrow{\langle\theta_1\overline{x}y,\,\theta_2 x(y)\rangle[R_{x(y),\,\overline{x}\nu y}(P,P,r_2,r_1)]} (\nu y)(P'\,|\,P'')\,|\,!P}$$

In allen Regeln stehen θ, θ_1, θ_2 für beliebige Routen und μ für ein beliebiges Transitionslabel. $\mathrm{fn}(\mu)$ ist definiert als $\mathrm{fn}(a)$, wobei a die einzige im Label μ vorkommende Aktion ist. Analog wird auch die Funktion bn von Aktionen auf Label erweitert. □

Auf die gleiche Weise wie bei der Transitionssemantik des klassischen π-Kalküls lässt sich eine Sπ-Variante mit Definitionsgleichungen formulieren. Tatsächlich verwendet die ursprünglich von Corrado Priami vorgeschlagene Syntax Definitionen statt des Replikationsoperators.

Eine Verwendung von polyadischen Eingabe- und Ausgabe-Präfixen ist in Sπ nicht ohne weiteres möglich. Das Senden von mehreren Namen über den gleichen Kommunikationskanal kann mit jeweils verschiedenen Raten geschehen, so dass sich das Problem ergibt, eine gemeinsame Rate für die Gesamtaktion finden zu müssen. Dies verändert allerdings auch die Auftrittswahrscheinlichkeit der Aktion, was einer semantisch äquivalenten Darstellung entgegensteht. Wie das Anwendungsbeispiel aus Abschnitt 3.4 zeigt, erleichtert eine polyadische Kalkülvariante die Modellierung biologischer Systeme erheblich, daher ist das Fehlen einer solchen Variante tatsächlich als Nachteil von Sπ zu sehen.

Der stochastische π-Kalkül weist allerdings noch wesentlich schwerwiegendere Probleme auf. Die eigentlich als Vorteil gedachte Transitionssemantik, die Interaktionen mit externen Prozessen und damit die Möglichkeit einer Implementierung von Schnittstellen eines Prozesses mit sich bringt, führt im Fall von freien Eingabeaktionen zu einer Vielzahl an ähnlichen Transitionen. Die Auswirkung auf die Ausgaberate und damit auf die Auftrittswahrscheinlichkeiten aller möglichen Transitionen eines Prozesses lässt sich während der Modellierung kaum abschätzen. Doch auch bei Verzicht auf freie Eingaben durch die Verwendung geeigneter Restriktionen der Kommunikationskanäle ergibt sich eine wenig intuitive Abhängigkeit der Auftrittswahrscheinlichkeiten von der Anzahl möglicher Transitionen. Wir ziehen zur Veranschaulichung folgendes Beispiel heran.

Beispiel 4.1.8: Zwei Proteine, die in der Lage sind, sich über eine Bindungsstelle zu einem Komplex zu formieren, können in einer Zelle jeweils in unterschiedlicher Konzentration vorliegen. Wir beschreiben die Proteine vereinfachend durch Teilprozesse der Form $\overline{x}y[3].\mathbf{0}$ und $x(y)[3].\mathbf{0}$. Daneben sei in der Zelle eine weitere Aktion, etwa eine Genexpression möglich, die wir als einfache Ausgabe der Form $!\overline{z}(u)[2].\mathbf{0}$ modellieren. In folgendem Zellprozess mit jeweils einem Protein von jeder Sorte ergibt sich eine Ausgangsrate von 5 (Eine Intraaktion mit Rate $\frac{3}{3} * \frac{3}{3} * 3$ und eine Interaktion mit Rate 2) und damit eine Auftrittswahrscheinlichkeit der Genexpression von 40%.

$$P = (\nu x)(\overline{x}y[3].\mathbf{0} \mid x(y)[3].\mathbf{0} \mid !\overline{z}(u)[2].\mathbf{0})$$

Im Prozess P' hingegen, der drei Proteine von der ersten Sorte und zwei von der zweiten Sorte enthält, ergibt sich eine Ausgangsrate von 8 (6 Intraaktionen mit Rate $\frac{3}{9} * \frac{3}{6} * 6$ und eine Interaktion mit Rate 2). Die Auftrittswahrscheinlichkeit der Genexpression ist daher auf 25% geschrumpft.

$$P' = (\nu x)(\overline{x}y[3].\mathbf{0} \mid \overline{x}y[3].\mathbf{0} \mid \overline{x}y[3].\mathbf{0} \mid x(y)[3].\mathbf{0} \mid x(y)[3].\mathbf{0} \mid !\overline{z}(u)[2].\mathbf{0})$$

Die Abhängigkeit der Genexpression von den damit nicht im Zusammenhang stehenden Proteinen ist sicherlich nicht erwünscht. Vielmehr erwartet man bei gleichbleibender Konzentration der Transkriptionsfaktoren des Gens (die in den Beispielprozessen nicht modelliert wurden) eine Expression mit konstanter Rate. □

Arbeiten, die den stochastischen π-Kalkül auf biologische Problemstellungen anwenden, verwenden aufgrund der genannten Probleme stets angepasste Varianten

des Kalküls, von denen die einflussreichste im nachfolgenden Abschnitt diskutiert wird.

4.2 Biochemischer stochastischer π-Kalkül

Speziell für die Modellierung von Genexpression und Proteininteraktionen wurde von der Gruppe um Aviv Regev, Corrado Priami, Ehud Shapiro und William Silverman eine Variante des stochastischen π-Kalküls entwickelt, die leichter zu implementieren ist und mit einer auf der Reduktionssemantik des klassischen π-Kalküls basierenden Semantik auskommt. Die Eignung dieses *biochemischen stochastischen π-Kalküls* wurde anhand zahlreicher kleinerer Modellierungsbeispiele in dem eigens implementierten Simulator BioSpi, der Prozesse des Kalküls interpretieren kann, gezeigt.

Die Syntax des biochemischen stochastischen π-Kalküls (kurz: bSπ) entspricht bis auf wenige Unterschiede der von Sπ: Alternative Prozessterme sind in bSπ nur erlaubt, wenn jede der Alternativen als äußersten Operator ein Präfix hat. Z.B. ist der Prozess $\overline{x}y.P_1 + x(z).P_2 + x(z).P_3$ erlaubt, der Prozess $\overline{x}y.P_1 \mid x(z).P_2 + x(z).P_3$ jedoch nicht. Auch für die Replikation gilt, dass der äußerste Operator des replizierten Prozesses ein Präfix sein muss. Die Regel (5) der strukurellen Kongruenz wird durch (5′) $!\pi.P \equiv \pi.(P \mid !\pi.P)$ ersetzt. Ein weiterer Unterschied liegt in der Assoziation von stochastischen Raten mit den Kanalnamen eines Präfixes statt mit den Präfixen selbst. Für zwei komplementäre Präfixe $\overline{x}y[r_1]$ und $x(z)[r_2]$ muss nun also stets $r_1 = r_2$ gelten.

Mit diesen Einschränkungen lässt sich eine deutlich leichter auszuwertende Wettlaufbedingung formulieren. Da in der Reduktionssemantik keine echten Interaktionen zwischen Prozessen möglich sind, sondern sämtliches Systemverhalten in einem Prozessterm zusammengefasst werden muss, reicht es nun aus, lediglich Raten für Intraaktionen zu berechnen. Die einem Namen zugeordnete Rate, die wir im Folgenden *Basisrate* nennen, wird dazu mit der Anzahl von ungeschützten Input- bzw. Output-Präfixen mit demselben Namen multipliziert. Neben der klassischen Kommunikation zwischen zwei parallelen Komponenten existiert im biochemischen stochastischen π-Kalkül noch eine speziellere Intraaktion, mit deren Hilfe *Homodimerisation*, d.h. Komplexbildung zweier identischer Proteine über komplementäre Interaktionsstellen modelliert werden kann. In unserem Kalkül bedeutet dies, dass zwei identische parallele Komponenten existieren, die jeweils ein Eingabe- und ein Ausgabepräfix mit einem Kanal $x \in \mathcal{H}$ als Alternativen besitzen, wobei $\mathcal{H} \subseteq \mathcal{N}$ die Menge der für Homodimerisationen reservierten Namen ist. Die Rate einer Homodimerisation ergibt sich durch $1/2 * r * |Q| * (|Q| - 1)$ wobei r die Basisrate des Kommunikationskanals und $|Q|$ die Anzahl der Teilprozesse mit alternativen Eingabe- und Ausgabepräfixen des Kanals ist. Wir definieren zwei Funktionen, die die Zählarbeit bei der Auswertung der semantischen Regeln übernehmen.

$$
\begin{aligned}
&\mathrm{In}_\tau(P) = 0 \\
&\mathrm{In}_x(\mathbf{0}) = 0 \\
&\mathrm{In}_x(\pi_1.P_1 + \ldots + \pi_n.P_n) = |\{\pi_i \mid i = 1, \ldots, n,\ \pi_i = x(y)\}| \\
&\mathrm{In}_x(P \mid Q) = \mathrm{In}_x(P) + \mathrm{In}_x(Q)
\end{aligned}
$$

$$\mathrm{In}_x((\nu z)P) = \begin{cases} \mathrm{In}_x(P) & \text{falls } x \neq z \\ 0 & \text{sonst} \end{cases}$$
$$\mathrm{In}_x(!P) = \mathrm{In}_x(P)$$
$$\mathrm{In}_x([x = x]P) = \mathrm{In}_x(P)$$

Für nicht beobachtbare Aktionen findet keine Zählung statt, daher stellt die im Präfix definierte Rate die endgültige Rate der Aktion dar. Für den Namen x eines Kanals zählt die Funktion $\mathrm{In}_x(P)$ die Vorkommen von Eingabe-Präfixen mit diesem Kanal in P, die unmittelbar in einen Reduktionsschritt involviert sein können, d.h. nicht durch ein voranstehendes Präfix geschützt sind. Analog berechnet die Funktion $\mathrm{Out}_x(P)$ die Vorkommen von Ausgabe-Präfixen. (In der Definition ist $x(y)$ durch $\overline{x}y$ zu ersetzen.) Wir führen nun die Semantik anhand der Reduktionsregeln ein.

Definition 4.2.1 (Reduktion in bSπ): Ein Prozess P kann genau dann zu einem Prozess P' evolvieren, wenn sich $P \longrightarrow^* P'$ mittels einer endlichen Anzahl von Anwendungen der sechs im Folgenden aufgeführten Regeln ableiten läßt.

$$\textsc{Inter:}\quad \frac{x \notin \mathcal{H}}{(\overline{x}y[r].P_1 + Q_1) \mid (x(z)[r].P_2 + Q_2) \overset{x,r,1,1}{\longrightarrow} P_1 \mid P_2\{y/z\}}$$

$$\textsc{Homo:}\quad \frac{x \in \mathcal{H}, \quad P = \overline{x}y[r].P_1 + x(z)[r].P_2}{(P + Q_1) \mid (P + Q_2) \overset{x,1/2*r,2,1}{\longrightarrow} P_1 \mid P_2\{y/z\}}$$

$$\textsc{Tau:}\quad \frac{}{\tau[r].P + Q \overset{\tau,r,1,1}{\longrightarrow} P}$$

$$\textsc{Par:}\quad \frac{P_1 \overset{a,r_b,r_i,r_o}{\longrightarrow} P_1'}{P_1 \mid P_2 \overset{a,r_b,r_i+\mathrm{In}_a(P_2),r_o+\mathrm{Out}_a(P_2)}{\longrightarrow} P_1' \mid P_2} \qquad \textsc{Res:}\quad \frac{P \overset{a,r_b,r_i,r_o}{\longrightarrow} P'}{(\nu z)P \overset{a,r_b,r_i,r_o}{\longrightarrow} (\nu z)P'}$$

$$\textsc{Struct:}\quad \frac{Q \overset{a,r_b,r_i,r_o}{\longrightarrow} Q', \quad Q \equiv P, \quad Q' \equiv P'}{P \overset{a,r_b,r_i,r_o}{\longrightarrow} P'}$$

Die Begriffe *Reduktionsableitung* und *Ableitungsschritt* sind analog zu den entsprechenden Begriffen in der Reduktionssemantik des klassischen π-Kalküls definiert. □

Die Reduktionsregeln sind nun ebenfalls mit Labeln versehen, die in Form eines Quadrupels (a, r_b, r_i, r_o) vorliegen. Der Parameter a steht für den Kommunikationskanal der Interaktion bzw. τ bei nicht beobachtbaren Operationen. Von den drei Ratenparametern steht r_b für die Basisrate des Namens, während r_i bzw. r_o die Anzahl der ungeschützten Eingabe- bzw. Ausgabe-Präfixe mit Namen a innerhalb des Prozesses angeben. Eine Regel Sum für Reduktionen innerhalb von Alternativen ist aufgrund der Einschränkung, dass sämtliche alternativen Teilprozesse mit einem Präfix beginnen müssen, nicht mehr notwendig. Stattdessen kommt die Regel Homo zur Modellierung von Homodimerisationen hinzu. Die Regel Struct verwendet die strukturelle Kongruenz aus Definition 3.2.2 (wobei Regel (5) durch (5′) zu ersetzen ist).

Über die Reduktionslabel lässt sich die Wettlaufbedingung für einen Prozess auswerten. Dies geschieht, indem für alle Namen, über die in einem Reduktionsschritt eine Kommunikation stattfinden kann, eine Rate $R_x(P)$ berechnet wird.

$$R_x(P) = r_b * r_i * r_o \qquad \text{für } P \overset{x,r_b,r_i,r_o}{\longrightarrow} P'$$

Die Summe dieser Raten ist die Ausgangsrate des Prozesses $er(P)$.

$$er(P) = \sum_{x \in \{y \mid P \overset{y,r_b,r_i,r_o}{\longrightarrow} P'\}} r_b * r_i * r_o$$

Man beachte, dass hier im Gegensatz zur Wettlaufbedingung in Sπ auch für mehrere mögliche Ableitungsschritte mit dem gleichen Kanalnamen nur ein Summand in die Ausgangsrate aufgenommen wird. Als nächstes wird ein Kanalname für die nächste auszuführende Aktion ausgewählt. Die Wahrscheinlichkeit der Wahl eines Namens x ist hierbei

$$p(x) = R_x(P)/er(P)\,.$$

Existieren mehrere Kommunikationsmöglichkeiten in P über den ausgewählten Namen, so wird unter diesen mit jeweils gleicher Wahrscheinlichkeit eine ausgewählt. (Aufgrund der stets identischen Raten der Intraaktionen mit gleichem Namen ist dieses Vorgehen tatsächlich korrekt.) Als letztes muss die Dauer der ausgewählten Intraaktion bestimmt werden, um das zeitliche Verhalten der Evolution des Prozesses zu erfassen. Die Dauer wird über die Inversionsmethode aus der Verteilungsfunktion der Exponentialverteilung ermittelt und ergibt sich zu

$$t = \frac{\ln(1 - F_{Exp})}{-er(P)}\,,$$

wobei für F_{Exp} eine gleichverteilte Zufallsvariable im Interval $[0, 1)$ eingesetzt wird. Das beschriebene Vorgehen entspricht einer leicht abgewandelten Anwendung des Gillespie-Algorithmus ([Gil_77]), dessen Eignung für die Modellierung biochemischer Netzwerke in zahlreichen experimentellen Studien gezeigt wurde.

In den meisten Arbeiten wird bSπ mit Definitionsgleichungen statt des Replikationsoperators eingeführt. Unser Kalkül lässt sich, wie bereits für die Redukionssemantik des klassischen π-Kalküls beschrieben, durch Ersetzen der Regel $(5')$ der strukturellen Kongruenz in eine entsprechende Variante überführen. Ebenso lässt bSπ die Verwendung von polyadischen Eingabe- und Ausgabe-Präfixen zu, da die Raten mit den Kanalnamen assoziiert werden. Um eine korrekte Übersetzung in einen monadischen Prozess zu gewährleisten, muss allerdings die Anzahl der Argumente in den polyadischen Präfixen für jeden Kanal eindeutig festgelegt werden, damit die Dauer und Auftrittswahrscheinlichkeit der Aktionen mit denen des monadischen Kalküls übereinstimmen.

Bisher haben wir für die Raten nur reelle Zahlen zugelassen. Dies entspricht der Auffassung, dass die Dauer aller modellierten Vorgänge vergleichbar ist. Wie bereits in Kapitel 2 erwähnt, kann die zeitliche Dimension von molekularen Vorgängen aber große Unterschiede aufweisen. Es kann daher sinnvoll sein, sogenannte *augenblickliche Aktionen* zuzulassen. Wir erweitern den Definitionsbereich für die Basisraten auf $r \in \mathbb{R}^+ \cup \{\infty\}$, wobei im Falle zweier komplementärer und ungeschützter Präfixe

mit unendlicher Rate in parallelen Teilprozessen die Wettlaufbedingung zu Gunsten einer Intraaktion zwischen diesen Teilprozessen außer acht gelassen wird. Sind mehrere Ableitungsschritte mit unendlichen Basisraten der beteiligten Präfixe möglich, so wird zwischen diesen nichtdeterministisch (d.h. mit gleicher Wahrscheinlichkeit) ausgewählt.

Beispiel 4.2.2: Dieses Beispiel soll sowohl die Verwendung unendlicher Basisraten, als auch den Intraaktionstyp der Homodimerisation veranschaulichen. Wir betrachten den Prozess

$$\overline{z}x[5].\overline{z}x[5].Q_1 \mid z(u)[5].((u(y)[\infty].\mathbf{0} + \overline{u}z[\infty].z(u)[5].Q_2) \mid z(u)[5].Q_3) \mid (x(y)[\infty].\mathbf{0} + \overline{x}z[\infty].z(u)[5].Q_2)$$

mit $x \in \mathcal{H}$. Zunächst ist keine Intraaktion unter Beteiligung der Präfixe mit unendlichen Raten möglich. Der einzige mögliche Ableitungsschritt besteht in einer Kommunikation der ersten beiden parallelen Komponenten über den Kanal z.

$$\stackrel{z,5,1,1}{\longrightarrow} \quad \overline{z}x[5].Q_1 \mid (x(y)[\infty].\mathbf{0} + \overline{x}z[\infty].z(u)[5].Q_2) \mid z(u)[5].Q_3 \mid (x(y)[\infty].\mathbf{0} + \overline{x}z[\infty].z(u)[5].Q_2)$$

Durch diesen Schritt sind zwei Intraaktionen ermöglicht worden: Entweder kann eine Kommunikation zwischen der ersten und dritten parallelen Komponente über den Kanal z stattfinden oder die zweite und vierte Komponente schließen sich in einer Homodimerisation zusammen. Die Wettlaufbedingung wird aufgrund der unendlichen Rate der Homodimerisation nicht berücksichtigt und es ergibt sich der Prozess

$$\stackrel{x,\infty,2,1}{\longrightarrow} \overline{z}x[5].Q_1 \mid (\mathbf{0} \mid z(u)[5].Q_2) \mid z(u)[5].Q_3 \ .$$

Zuletzt kommuniziert der erste Teilprozess über z mit einem der beiden übrigen. Wir entscheiden uns für den hinteren und eliminieren dabei den untätigen $\mathbf{0}$-Teilprozess in der Mitte über die STRUCT-Regel.

$$\stackrel{z,5,2,1}{\longrightarrow} Q_1 \mid z(u)[5].Q_2 \mid Q_3\{x/u\} \qquad \square$$

Wir schließen diesen Abschnitt mit einigen Literaturhinweisen ab. Der biochemische stochastische π-Kalkül wurde zunächst in der Arbeit [PRSS_01] vorgestellt und später für die Beschreibung von Glycogen Biosynthese ([ReSh_04]), biochemische Pathways (RTK-MAPK Pathway) und zirkadische Uhren verwendet ([Reg_01]). Die hohe Akzeptanz dieser und ähnlicher Varianten des stochastischen π-Kalküls in der wissenschaftlichen Gemeinschaft zeigt sich anhand der wachsenden Zahl an Veröffentlichungen aus verschiedensten Ländern. Als eine Auswahl seien dem geneigten Leser folgende aktuelle Arbeiten empfohlen: Céline Kuttler, Joachim Niehren, Cédric Lhoussaine und Denys Duchier schlagen in [KuLN_06], [KuDu_06], [KuNi_06] und [Kut_06] mehrere Erweiterungen des biochemischen stochastischen π-Kalküls um Konzepte aus objektorientierten Programmiersprachen vor, um eine intuitivere und strukturiertere Beschreibung großer Systeme zu gewährleisten. Die Ansätze werden stets an kleineren Modellierungen beispielhaft veranschaulicht. Ein weiterer Simulator auf Basis des biochemischen stochastischen π-Kalküls, SPiM, wird in [PhCa_04] zusammen mit einem Korrektheitsbeweis für eine abstrakte Maschine des Kalküls von Andrew Phillips und Luca Cardelli vorgestellt. Von den gleichen Autoren gibt es zwei Arbeiten über grafische Repräsentationen von Prozessen

in bSπ ([PhCa_05], [PhCC_06]). Von Claudio Eccher und Paola Lecca existiert eine Arbeit, die eine Übersetzung von SBML (Systems Biology Markup Language), einer gebräuchlichen statischen Beschreibungssprache in der Biologie, in bSπ Prozesse vorstellt ([EcLe_06]). Des Weiteren gibt es einige komplexere Modellierungen biologischer Vorgänge in bSπ-Varianten: In [CCDM_05] und [CCDBM_06] wird eine komplette, aber sehr einfache Prokaryotenzelle als bSπ-Modell untersucht. Von Corrado Priami und Paola Lecca wird in [LePr_03] die Kontrolle von Zellzyklen in Eukaryotenzellen modelliert und in [ABCGM_07] widmen sich die Autoren einer Modellierung von Altersprozessen in Bakterienkolonien.

4.3 Anwendungsbeispiel: Genexpression in bSπ

Wir betrachten erneut das System aus Abschnitt 3.4, das Genexpression mit positivem Feedback am Beispiel zweier Gene sowie ihrer Produkte modelliert. Mit Hilfe des biochemischen stochastischen π-Kalküls, den wir in der polyadischen Variante mit Definitionsgleichungen einsetzen, sind wir nun in der Lage, auch die quantitativen Aspekte des Systems korrekt zu beschreiben. Da das System Sys bereits den syntaktischen Einschränkungen, die mit bSπ einhergehen, genügt, brauchen wir lediglich sämtliche Präfixe um stochastische Raten zu erweitern. Da bSπ eine Reduktionssemantik verwendet und daher keine Interaktionen mit externen Prozessen zulässt, können wir auf die Restriktionen im initialen Prozess (hier: bSπ-Sys) verzichten.

$$
\begin{aligned}
\text{bS}\pi\text{-Sys} &= \text{GenA} \mid \text{GenTF} \mid \text{Transkript} \mid \text{Translat} \mid \text{RnaZerf} \mid \text{ProtZerf} \\
\\
\text{GenA} &= tf^-()[4].(\text{GenA} \mid \text{RnaA}) + tf^+()[40].(\text{GenA} \mid \text{RnaA}) \\
\text{RnaA} &= trans()[1].(\text{RnaA} \mid \text{ProteinA}) + zerf_r()[1].\mathbf{0} \\
\text{ProteinA} &= (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\text{BindungA} \mid \text{Kinase}) \\
\text{BindungA} &= \overline{binden}(bb_1, bb_2, bb_3)[0.1].\text{GebundenA} + \\
&\quad zerf_p()[0.1].\overline{bb_3}()[\infty].\mathbf{0} \\
\text{GebundenA} &= \overline{bb_1}()[10].\text{BindungA} + zerf_p()[0.1].\overline{bb_3}()[\infty].\overline{bb_3}()[\infty].\mathbf{0} \\
\text{Kinase} &= \overline{bb_2}(aktiv)[10].\text{Kinase} + bb_3()[\infty].\mathbf{0} \\
\\
\text{GenTF} &= tf^-()[4].(\text{GenTF} \mid \text{RnaTF}) + tf^+()[40].(\text{GenTF} \mid \text{RnaTF}) \\
\text{RnaTF} &= trans()[1].(\text{RnaTF} \mid \text{ProteinTF}) + zerf_r()[1].\mathbf{0} \\
\text{ProteinTF} &= binden(bb'_1, bb'_2, bb'_3)[0.1].\text{GebundenTF} + zerf_p()[0.1].\mathbf{0} \\
\text{GebundenTF} &= bb'_1()[10].\text{ProteinTF} + \\
&\quad bb'_2(aktiv')[10].bb'_1()[10].\text{AktivTF}(aktiv') + bb'_3()[\infty].\mathbf{0} \\
\text{AktivTF}(a) &= \overline{a}()[100].\text{AktivTF}(a) + zerf_p()[0.1].\mathbf{0} \\
\\
\text{Transkript} &= \overline{tf^-}()[4].\text{Transkript} + aktiv()[100].\overline{tf^+}()[40].\text{Transkript} \\
\text{Translat} &= \overline{trans}()[1].\text{Translat} \\
\text{RnaZerf} &= \overline{zerf_r}()[1].\text{RnaZerf} \\
\text{ProtZerf} &= \overline{zerf_p}()[0.1].\text{ProtZerf}
\end{aligned}
$$

Die Genexpression wird nun bei Anwesenheit des aktiven Transkriptionsfaktors tatsächlich verstärkt, da die Rate des Kanals tf^+ in den Prozessen Transkript, GenA und GenTF zehn mal so groß ist wie die Rate des Kanals tf^- und damit auch die Auftrittswahrscheinlichkeit der Transkription erhöht und deren Dauer im Fall der Einleitung über tf^+ deutlich verkürzt ist. Auch das Problem, dass Zerfallsvorgänge von Komplexen durch andere Aktionen unterbrochen werden konnten, besteht nicht mehr. Die Rate des Kanals bb_3 ist ∞, was bedeutet, dass die Wettlaufbedingung im Fall eines eingeleiteten Zerfallsprozesses ignoriert wird, bis alle Komponenten des Komplexes ihre Auflösung abgeschlossen haben.

Beispiel 4.3.1: In diesem Beispiel betrachten wir eine Reduktionsableitung zur Veranschaulichung der Modellierung. Wir ersetzen gegebenenfalls die Prozessbezeichner, die durch einen Reduktionsschritt mit der Regel Struct entfaltet werden, bereits vor den Regelanwendungen in einem Zwischenschritt durch die jeweiligen Prozessterme, um Interaktionsmöglichkeiten explizit sichtbar zu machen. Die zu entfaltenden Bezeichner sind vor diesen Zwischenschritten stets unterstrichen, wie auch die Kommunikationspartner bei Interaktionen.

	$\underline{\text{bS}\pi\text{-Sys}}$
$=$	$\underline{\text{GenA}} \mid \text{GenTF} \mid \underline{\text{Transkript}} \mid \text{Translat} \mid \text{RnaZerf} \mid \text{ProtZerf}$
$\xrightarrow{tf^-,4,2,1}$	$(\text{GenA} \mid \underline{\text{RnaA}}) \mid \text{GenTF} \mid \text{Transkript} \mid \underline{\text{Translat}} \mid \text{RnaZerf} \mid \text{ProtZerf}$
$\xrightarrow{trans,1,1,1}$	$(\text{GenA} \mid (\text{RnaA} \mid \text{ProteinA})) \mid \underline{\text{GenTF}} \mid \underline{\text{Transkript}} \mid \text{Translat} \mid \text{RnaZerf} \mid \text{ProtZerf}$
$\xrightarrow{tf^-,4,2,1}$	$(\text{GenA} \mid \text{RnaA} \mid \text{ProteinA}) \mid (\text{GenTF} \mid \underline{\text{RnaTF}}) \mid \text{Transkript} \mid \underline{\text{Translat}} \mid \text{RnaZerf} \mid \text{ProtZerf}$
$\xrightarrow{trans,1,2,1}$	$(\text{GenA} \mid (\text{RnaA} \mid \text{ProteinA})) \mid (\text{GenTF} \mid \underline{\text{RnaTF}} \mid \text{ProteinTF}) \mid \text{Transkript} \mid \text{Translat} \mid \underline{\text{RnaZerf}} \mid \text{ProtZerf}$
$\xrightarrow{zerf_r,1,2,1}$	$(\text{GenA} \mid \text{RnaA} \mid \underline{\text{ProteinA}}) \mid (\text{GenTF} \mid \text{ProteinTF}) \mid \text{Transkript} \mid \text{Translat} \mid \text{RnaZerf} \mid \text{ProtZerf}$
$=$	$(\text{GenA} \mid \text{RnaA} \mid (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\underline{\text{BindungA}} \mid \text{Kinase})) \mid (\text{GenTF} \mid \underline{\text{ProteinTF}}) \mid \text{Transkript} \mid \text{Translat} \mid \text{RnaZerf} \mid \text{ProtZerf}$
$\xrightarrow{binden,0.1,1,1}$	$(\text{GenA} \mid \text{RnaA} \mid (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\text{GebundenA} \mid \underline{\text{Kinase}})) \mid (\text{GenTF} \mid \underline{\text{GebundenTF}}) \mid \text{Transkript} \mid \text{Translat} \mid \text{RnaZerf} \mid \text{ProtZerf}$
$\xrightarrow{bb_2,10,1,1}$	$(\text{GenA} \mid \text{RnaA} \mid (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\underline{\text{GebundenA}} \mid \text{Kinase})) \mid (\text{GenTF} \mid \underline{bb_1()[10].\text{AktivTF}(aktiv)}) \mid \text{Transkript} \mid \text{Translat} \mid \text{RnaZerf} \mid \text{ProtZerf}$

$$\xrightarrow{bb_1,10,1,1} (\textsc{GenA} \mid \textsc{RnaA} \mid (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\textsc{BindungA} \mid \textsc{Kinase})) \mid (\textsc{GenTF} \mid \textsc{AktivTF}(aktiv)) \mid \underline{\textsc{Transkript}} \mid \textsc{Translat} \mid \textsc{RnaZerf} \mid \textsc{ProtZerf}$$

$$= (\textsc{GenA} \mid \textsc{RnaA} \mid (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\textsc{BindungA} \mid \textsc{Kinase})) \mid (\textsc{GenTF} \mid \underline{\textsc{AktivTF}(aktiv)) \mid (\overline{tf^-}()[4].\textsc{Transkript}+ aktiv()[100].\overline{tf^+}()[40].\textsc{Transkript})} \mid \textsc{Translat} \mid \textsc{RnaZerf} \mid \textsc{ProtZerf}$$

$$\xrightarrow{aktiv,100,2,1} (\underline{\textsc{GenA}} \mid \textsc{RnaA} \mid (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\textsc{BindungA} \mid \textsc{Kinase})) \mid (\textsc{GenTF} \mid \textsc{AktivTF}(aktiv)) \mid \underline{\overline{tf^+}()[40].\textsc{Transkript}} \mid \textsc{Translat} \mid \textsc{RnaZerf} \mid \textsc{ProtZerf}$$

$$\xrightarrow{tf^+,40,1,1} (\textsc{GenA} \mid \textsc{RnaA} \mid \textsc{RnaA} \mid (\nu\, bb_1)(\nu\, bb_2)(\nu\, bb_3)(\textsc{BindungA} \mid \textsc{Kinase})) \mid (\textsc{GenTF} \mid \textsc{AktivTF}(aktiv)) \mid \textsc{Transkript} \mid \textsc{Translat} \mid \textsc{RnaZerf} \mid \textsc{ProtZerf}$$

In den ersten vier Reduktionsschritten werden die beiden Gene jeweils in RNA übersetzt, aus der anschließend je ein Protein generiert wird. Der fünfte Schritt realisiert einen Zerfallsvorgang bei einem der RNA-Stränge. Im darauf folgenden Zwischenschritt wird das Protein A durch seine zwei Komponenten ersetzt und anschließend im sechsten Reduktionsschritt an das Protein TF gebunden. Die nächsten beiden Schritte realisieren die Aktivierung von Protein TF und das Auftrennen des Komplexes. Es folgt ein weiterer Zwischenschritt, der die Möglichkeit der Kommunikation über Kanal *aktiv* und darauffolgend über Kanal tf^+ aufzeigt. Die letzten beiden Reduktionsschritte setzen schließlich die verstärkte Genexpression um (hier beispielhaft am Gen A). □

Die Modellierung liefert in der Simulation (siehe [PRSS_01]) tatsächlich eine realistische Entwicklung, die mit den Ergebnissen von Laborexperimenten im Einklang steht. Auch in anderen Anwendungsbeispielen (siehe Literaturhinweise am Ende des vorigen Abschnitts) hat sich bSπ durchaus als leicht implementierbarer und biologisch glaubwürdiger Ansatz bewährt. Doch die bisher betrachteten Systeme sind alle verhältnismäßig klein und in sich geschlossen, d.h. relativ unabhängig von äußeren Einflüssen. Auch beschränken sich die Modelle auf Gen-Protein- bzw. Protein-Protein-Interaktionen. Es ist leicht vorstellbar, dass ein System, welches auch Membranoperationen enthält, zu einer großen und unübersichtlichen Implementierung führen kann. So wie durch die Homodimerisations-Regel in bSπ umständliche Formulierungen von bidirektionaler Kommunikation umgangen werden, wären auch spezielle Regeln für Membranoperationen als Ergänzung zu den sehr allgemeinen Kommunikations-Regeln des Kalküls sinnvoll. Die Verwendung einer Reduktionssemantik in bSπ erschwert darüber hinaus die Umsetzung von Schnittstellen zu äußeren Vorgängen, die nicht oder erst zu einem späteren Zeitpunkt implementiert werden sollen (wie bei großen Softwareprojekten, werden auch biologische Modelle nach und nach erweitert und überarbeitet.) Wir stellen im folgenden Kapitel eine Erweiterung des π-Kalküls vor, die einen Lösungsansatz für die geschilderten Probleme darstellt.

Kapitel 5

β-Binder

Auf allen Ebenen von Lebensprozessen, seien es intermolekulare Reaktionen, interzellulärer Stoffaustausch oder gar die Organisation von Gewebe zu Organen und Organismen, spielen die räumliche Anordnung sowie die Abgrenzung von Entitäten untereinander eine wesentliche Rolle. Dieser Aspekt wird bei Modellierungen mittels des klassischen π-Kalküls und des stochastischen π-Kalküls vernachlässigt. Es ist zwar möglich, über die Verwendung des Restriktionsoperators bestimmte Interaktionen zu unterbinden und damit einen rudimentären Distanzbegriff umzusetzen, allerdings lässt sich in einem größeren Modell nicht mehr auf einen Blick sagen, welche Teilprozesse eine biologische Einheit bilden. Mit zunehmender Evolution eines Systems werden die abgeleiteten Teilprozesse einer Entität immer schwieriger zu identifizieren. *β-Binder* stellen eine Erweiterung des π-Kalküls dar, die eine explizite Einteilung von Prozessen in sogenannte Kompartimente erlaubt, die über getypte Bindungstellen miteinander in Kontakt stehen. Obwohl der Kalkül eine Reduktionssemantik verwendet, ermöglicht er eine modulare Implementierung mit weitgehend unabhängigen Systemkomponenten. Hierbei dienen die Typen der Bindungsstellen als einfache Schnittstellen zur Modellierung von Interaktionsmöglichkeiten zwischen den einzelnen Kompartimenten.

Der Kalkül der β-Binder wurde von Corrado Priami, der ebenfalls Autor des stochastischen π-Kalküls ist, unter Zusammenarbeit mit Paola Quaglia entwickelt ([PrQu1_05]) und besteht aus einer sehr anpassungsfähigen Zusatzsyntax, in die verschiedene Varianten des π-Kalküls integriert werden können. Die eigentlichen Berechnungen finden dabei in der π-Kalkül-Variante statt, während die β-Binder-Syntax lediglich der Strukturierung von Entitäten und der Umsetzung räumlicher Aspekte dient. Weitere Arbeiten zum Kalkül von den gleichen Autoren behandeln abstrakte Beispielmodellierungen häufig vorkommender biologischer Phänomene ([PrQu2_05]) bzw. eine Übersetzung von Kohn Interaction Maps zu β-Bindern am Beispiel des Zellzyklus von Säugetieren ([CiPQ_05]). Darüber hinaus existiert eine Arbeit von Davide Prandi, die β-Binder verwendet, um Andockvorgänge von Molekülen mit Rezeptoren im Hinblick auf eine Anwendung in der pharmazeutischen Wirkstoffforschung zu formalisieren ([Pra_06]). In den Abschnitten 5.1 und 5.2 stellen wir den Kalkül zunächst in der von Priami und Quaglia vorgestellten Form vor, das heißt unter Verwendung einer reduzierten Variante des π-Kalküls, bevor wir in Abschnitt 5.3 zu einer stochastischen Variante auf Basis des biochemischen stochastischen π-Kalküls kommen und diese in Abschnitt 5.4 erneut auf das Genexpressionsbeispiel anwenden. Der Abschnitt 5.5 behandelt Modellierungen

einiger relevanter biologischer Phänomene, um die Anwendung des Kalküls in der Systembiologie zu verdeutlichen.

5.1 Syntax

Der β-Binder-Kalkül besitzt zwei verschiedene Prozessbegriffe. Die bisher verwendeten π-Prozesse behalten ihre Bedeutung, kommen jedoch nur noch als Bestandteile sogenannter *Bio-Prozesse* vor, die Kompartimente definieren, welche ein durch einen π-Prozess spezifiziertes Verhalten kapseln.

Definition 5.1.1 (Bio-Prozesse und β-Binder): Die Menge der *Bio-Prozesse* wird durch folgende Grammatik definiert.

$$Bio ::= \mathbf{0} \quad | \quad B\langle P\rangle \quad | \quad Bio \,||\, Bio$$

Hierbei bezeichnet P einen Prozess in einer Variante des π-Kalküls und B einen *(zusammengesetzten) β-Binder* (oder kurz *Binder*) gemäß

$$B ::= \beta(x:\Gamma) \quad | \quad \beta^h(x:\Gamma) \quad | \quad \beta(x:\Gamma)B \quad | \quad \beta^h(x:\Gamma)B$$

mit $x \in \mathcal{N}$ und $\Gamma \subset \mathcal{N}$, $\Gamma \neq \emptyset$, $x \notin \Gamma$. Ein β-Binder der Form $\beta(x:\Gamma)$ oder $\beta^h(x:\Gamma)$ heißt *elementar*. In $B = \beta(x:\Gamma)$ wird $x = \mathrm{sub}(B)$ als das *Subjekt* des elementaren β-Binders bezeichnet, während Γ den *Typ* von x darstellt. Wir nehmen stets an, dass die Subjekte der elementaren β-Binder eines Bio-Prozesses paarweise verschieden sind. □

Wir wollen im Folgenden die Buchstaben S, T und U für Bio-Prozesse verwenden und entsprechend B, B', B_1, B_2, ... für β-Binder. Mit B^* kürzen wir eine endliche Sequenz elementarer Binder $B_1 \ldots B_n$ mit $n \geq 0$ ab. Für zusammengesetzte Binder bezeichnet $\mathrm{sub}(B)$ die Menge der Subjekte der elementaren Binder in B.

Der untätige Bio-Prozess $\mathbf{0}$ ist wie der untätige π-Prozess zu keinem beobachtbaren Verhalten fähig. Ob mit $\mathbf{0}$ ein Bio-Prozess oder ein π-Prozess gemeint ist, wird durch den jeweiligen Kontext ersichtlich. Tatsächlich kann ein π-Prozess nicht ohne einen vorangestellten β-Binder existieren, so dass keine Verwechslungen möglich sind.

Ein Bio-Prozess der Form $B\langle P\rangle$ definiert ein Kompartiment mit dem durch P festgelegten Verhalten und den durch B spezifizierten Interaktionsmöglichkeiten mit seiner Umgebung. Dabei ist das Subjekt eines elementaren β-Binders in B ein Name, der in P als Kommunikationskanal vorkommen darf und über den eine Kommunikation mit einem anderen Kompartiment stattfinden kann, falls dieses einen elementaren Binder mit passendem Typ besitzt. Für eine Interaktion dieser Art ist es ausreichend, wenn die Typen der beteiligten elementaren Binder nicht disjunkt sind. Mit $\beta^h(x:\Gamma)$ wird ein β-Binder gekennzeichnet, der aktuell nicht an Interaktionen teilnehmen kann. Auf diese Weise lässt sich zum Beispiel eine versteckte Bindungsstelle eines Proteins modellieren. Wir nennen den Binder in diesem Fall naheliegenderweise *versteckt*. Ein nicht versteckter Binder heißt hingegen *aktiv*.

Mit $S \,||\, T$ wird die parallele Komposition von Bio-Prozessen realisiert. Die mit diesem Prozess verknüpfte Vorstellung ist die zweier biologischer Entitäten S und

T, die in einem System koexistieren. Als Entitäten kommen beispielsweise Proteine oder Zellen, evtl. aber auch nicht materielle Objekte, wie die Umweltprozesse aus dem Anwendungsbeispiel der Abschnitte 3.4 und 4.3 in Frage. Unter bestimmten Bedingungen können sich zwei parallele Kompartimente zusammenfügen (*join*), wobei die eingebetteten π-Prozesse dann als parallele Teilprozesse innerhalb eines einzelnen Kompartiments vorliegen. In ähnlicher Weise kann sich ein Kompartiment in zwei Kompartimente aufteilen (*split*). Wir werden im Zusammenhang mit der Definition der Semantik näher auf diese Operationen eingehen.

Ein π-Prozess, der in einen Bio-Prozess eingebettet ist, kann dessen Interaktionsverhalten von Innen heraus manipulieren. Dazu ist es erforderlich, die π-Kalkül-Syntax um einige Präfixe zu erweitern. Wir stellen in nachfolgender Definition die Syntax eines zunächst um Vergleichsoperatoren, alternative Teilprozesse und τ-Präfixe reduzierten und gleichzeitig um die erforderlichen neuen Präfixe erweiterten π-Kalküls vor, wie er üblicherweise in Arbeiten zu β-Bindern benutzt wird.

Definition 5.1.2 (π^β-Prozesse): Folgende Grammatik definiert die Menge der *π^β-Prozesse*:

$$P ::= \mathbf{0} \quad | \quad \pi.P \quad | \quad (\nu x)P \quad | \quad !\pi.P \quad | \quad P|P$$

wobei π ein Aktionspräfix gemäß folgender Definition darstellt:

$$\pi ::= \overline{x}y \quad | \quad x(y) \quad | \quad \exp(x,\Gamma) \quad | \quad \mathrm{hid}(x) \quad | \quad \mathrm{unh}(x) \qquad \square$$

Sämtliche Operatoren besitzen die gleiche Bedeutung wie im klassischen π-Kalkül. Zu den drei Binder-modifizierenden Präfixen exp, hid und unh sind folgende Erläuterungen angebracht.

- Das $\exp(x,\Gamma)$ (*expose*) Präfix erweitert den zusammengesetzten β-Binder des umgebenden Bio-Prozesses um einen elementaren Binder $\beta(x,\Gamma)$. Das Präfix bindet das Subjekt x des Binders innerhalb des jeweiligen Teilprozesses auf die gleiche Weise wie das Eingabe-Präfix $y(x)$ (siehe Definition 3.1.3).

- Mit $\mathrm{hid}(x)$ (*hide*) wird der elementare β-Binder des umgebenden Bio-Prozesses, der das Subjekt x besitzt, versteckt. Falls der ensprechende Binder bereits versteckt ist, oder kein solcher Binder im Bio-Prozess existiert, so kann das Präfix $\mathrm{hid}(x)$ nicht verarbeitet werden, der betroffene Teilprozess ist also handlungsunfähig, bis ein passender Binder aktiv gemacht oder gebildet wird. Das Präfix bindet den Namen x nicht.

- $\mathrm{unh}(x)$ (*unhide*) ist das Gegenstück zu $\mathrm{hid}(x)$ und aktiviert einen elementaren Binder mit Subjekt x unter der Prämisse, dass dieser existent und versteckt ist. Auch $\mathrm{unh}(x)$ wartet gegebenenfalls auf die Erfüllung der Prämisse und bindet den Namen x im jeweiligen Teilprozess nicht.

Substitutionen und α-Konversionen lassen sich auf die drei neuen Präfixe genauso anwenden, wie auf die Kommunikationspräfixe $\overline{x}y$ und $x(y)$. Dabei werden auch die Namen im Typ Γ eines expose-Präfixes substituiert, da diese frei vorkommen. Dadurch entsteht die Möglichkeit, β-Binder dynamisch über die Kommunikation mit anderen Kompartimenten anzupassen, wie in folgendem Beispiel gezeigt wird.

Beispiel 5.1.3: Der Bio-Prozess

$$S = \beta(x : \{u, u'\})\langle \overline{x}v.P\rangle \mid\mid \beta(y : \{u\})\langle y(z).\exp(y', \{z\}).Q\rangle$$

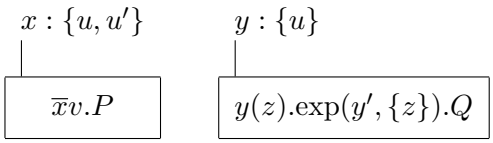
besteht aus zwei Kompartimenten, die jeweils einen elementaren Binder besitzen. Die beiden Binder haben nicht-disjunkte Typen und gestatten daher die Kommunikation der beiden Kompartimente, falls in den eingebetteten π^β-Prozessen komplementäre Kommunikationspräfixe mit den Subjekten der Binder als Kanal in ungeschützter Form vorliegen. (Dass die Subjekte der Binder nicht identisch sind, spielt hierbei keine Rolle, lediglich die Typung entscheidet über die Fähigkeit zur Interaktion.) Da entsprechende Ein- und Ausgabe-Präfixe vorliegen, ergibt sich nach Reduktion der Bio-Prozess

$$S' = \beta(x : \{u, u'\})\langle P\rangle \mid\mid \beta(y : \{u\})\langle \exp(y', \{v\}).Q\{v/z\}\rangle\,.$$

Nun kann das zweite Kompartiment einen neuen Binder bilden und verwendet dabei im Typ den Namen, den das andere Kompartiment ihm gesendet hat. Es ergibt sich der Bio-Prozess

$$S'' = \beta(x : \{u, u'\})\langle P\rangle \mid\mid \beta(y : \{u\})\beta(y' : \{v\})\langle Q\{v/z\}\rangle\,. \qquad \square$$

Um die Kompartimente eines Bio-Prozesses auf einen Blick identifizieren zu können, bietet sich die Verwendung einer grafischen Notation an. Anstatt die Binder eines Kompartiments in Klammern vor den π^β-Prozess zu schreiben, zeichnen wir in den folgenden Beispielen ein Rechteck um den Prozess, der für jeden elementaren Binder eine Bindungsstelle (abgehender vertikaler Strich) mit dem Subjekt des Binders und dessen Typ als Beschriftung besitzt. Um einen Binder als versteckt zu markieren, wird seine Beschriftung in Klammern gesetzt. Für den Bio-Prozess S aus dem Beispiel ergibt sich die Notation



Die relative Position der Binder eines Kompartiments ebenso wie die relative Position der Kompartimente selbst spielt für die semantische Bedeutung eines Bio-Prozesses keine Rolle. Wir werden dies im nächsten Abschnitt durch eine entsprechend erweiterte strukturelle Kongruenzrelation formal ausdrücken.

5.2 Semantik

Neben der strukturellen Kongruenz über π^β-Prozessen gibt es im β-Binder-Kalkül auch eine strukturelle Kongruenz über Bio-Prozessen, die wir durch das gleiche Relationssymbol darstellen. Die Regeln für Vergleichsoperationen und alternative Komponenten fallen aufgrund der eingeschränkten Syntax weg.

Definition 5.2.1 (Strukurelle Kongruenz): Zwei π^β-Prozesse P und Q, die sich durch Anwendung der folgenden vier Regeln ineinander überführen lassen, heißen *strukturell kongruent* (Notation $P \equiv Q$).

(1) $P \equiv Q$, falls $P =_\alpha Q$

(2) $P \mid Q \equiv Q \mid P, \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R, \quad P \mid \mathbf{0} \equiv P$

(3) $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P, \quad (\nu x)(P|Q) \equiv P|(\nu x)Q$ falls $x \notin \mathrm{fn}(P), \quad (\nu x)\mathbf{0} \equiv \mathbf{0}$

(4) $!\pi.P \equiv \pi.(P \mid !\pi.P)$

Die strukturelle Kongruenz zwischen Bio-Prozessen ist auf die gleiche Weise über die nachfolgenden vier Regeln definiert.

(5) $B\langle P\rangle \equiv B\langle Q\rangle$, falls $P \equiv Q$

(6) $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1, \quad S_1 \mid\mid (S_2 \mid\mid S_3) \equiv (S_1 \mid\mid S_2) \mid\mid S_3, \quad S \mid\mid \mathbf{0} \equiv S$

(7) $B_1B_2\langle P\rangle \equiv B_2B_1\langle P\rangle$

(8) $B^*\hat{\beta}(x:\Gamma)\langle P\rangle \equiv B^*\hat{\beta}(y:\Gamma)\langle P\{y/x\}\rangle$, falls $y \notin \mathrm{bn}(P) \cup \mathrm{fn}(P)$, $\hat{\beta} \in \{\beta, \beta^h\}$ □

Wir betrachten als nächstes die Regeln der Reduktionssemantik, die nun statt über π-Prozessen wie beim klassischen π-Kalkül über Bio-Prozessen definiert sind.

Definition 5.2.2 (Reduktion): Ein Bio-Prozess S kann genau dann zu einem Bio-Prozess S' evolvieren, wenn sich $S \longrightarrow^* S'$ mittels einer endlichen Anzahl von Anwendungen der im Folgenden aufgeführten Regeln ableiten läßt. In den Regeln wird mit $(\nu\vec{u})$ eine endliche Sequenz von Restriktionen der Form $(\nu u_1)\ldots(\nu u_n)$ mit $n \geq 0$ abgekürzt. $x \notin \vec{u}$ steht für $x \neq u_1, \ldots, x \neq u_n$.

INTRA: $$\frac{}{B\langle(\nu\vec{u})(\overline{x}y.P_1 \mid x(z).P_2 \mid P_3)\rangle \longrightarrow B\langle(\nu\vec{u})(P_1 \mid P_2\{y/z\} \mid P_3)\rangle}$$

INTER: $$\frac{P \equiv (\nu\vec{u})(\overline{x}y.P_1 \mid P_2), \quad Q \equiv (\nu\vec{v})(x'(y').Q_1 \mid Q_2), \quad \Gamma \cap \Gamma' \neq \emptyset}{\beta(x:\Gamma)B_1^*\langle P\rangle \mid\mid \beta(x':\Gamma')B_2^*\langle Q\rangle \longrightarrow \beta(x:\Gamma)B_1^*\langle P'\rangle \mid\mid \beta(x':\Gamma')B_2^*\langle Q'\rangle}$$
mit $P' = (\nu\vec{u})(P_1 \mid P_2)$, $Q' = (\nu\vec{v})(Q_1\{y/y'\} \mid Q_2)$ und $x, y \notin \vec{u}$, $x', y \notin \vec{v}$

BIOPAR: $$\frac{S \longrightarrow S'}{S \mid\mid T \longrightarrow S' \mid\mid T} \qquad \text{BIOSTRUCT:} \quad \frac{T \longrightarrow T', \; T \equiv S, \; T' \equiv S'}{S \longrightarrow S'}$$

EXPOSE: $$\frac{y \notin \vec{u}, \quad y \notin \mathrm{sub}(B), \quad y \notin \Gamma}{B\langle(\nu\vec{u})(\exp(x,\Gamma).P_1 \mid P_2)\rangle \longrightarrow B\beta(y:\Gamma)\langle(\nu\vec{u})(P_1\{y/x\} \mid P_2)\rangle}$$

HIDE: $$\frac{x \notin \vec{u}}{B^*\beta(x:\Gamma)\langle(\nu\vec{u})(\mathrm{hid}(x).P_1 \mid P_2)\rangle \longrightarrow B^*\beta^h(x:\Gamma)\langle(\nu\vec{u})(P_1 \mid P_2)\rangle}$$

UNHIDE: $$\frac{x \notin \vec{u}}{B^*\beta^h(x:\Gamma)\langle(\nu\vec{u})(\mathrm{unh}(x).P_1 \mid P_2)\rangle \longrightarrow B^*\beta(x:\Gamma)\langle(\nu\vec{u})(P_1 \mid P_2)\rangle}$$

JOIN: $$\frac{f_{\mathrm{join}}(B_1, B_2, P_1, P_2) = (B, \sigma_1, \sigma_2)}{B_1\langle P_1\rangle \mid\mid B_2\langle P_2\rangle \longrightarrow B\langle P_1\sigma_1 \mid P_2\sigma_2\rangle}$$

$$\textsc{Split}: \quad \frac{f_{\text{split}}(B, P_1, P_2) = (B_1, B_2, \sigma_1, \sigma_2)}{B\langle P_1 \mid P_2\rangle \;\longrightarrow\; B_1\langle P_1\sigma_1\rangle \parallel B_2\langle P_2\sigma_2\rangle}$$

In den Regeln JOIN und SPLIT bezeichnen σ_1 und σ_2 beliebige Substitutionen. Die Begriffe *Reduktionsableitung* und *Ableitungsschritt* sind analog zu den entsprechenden Begriffen in der Reduktionssemantik des klassischen π-Kalküls definiert. □

Betrachten wir die Regeln nun etwas genauer:

Intra: Mit dieser Regel werden Reduktionen innerhalb des in einen Bio-Prozess eingebetteten π^β-Prozesses realisiert. Parallele Teilprozesse mit komplementären Kommunikationspräfixen können auf die gleiche Weise reduziert werden wie im klassischen π-Kalkül. Die übrigen fünf Regeln der klassischen Reduktionssemantik sind aufgrund der eingeschränkten Syntax überflüssig. Die Regel (5) der strukturellen Kongruenz gewährleistet, dass strukturelle Umformungen in π^β-Prozessen auf das Level von Bio-Prozessen übertragen werden können.

Inter: Interaktionen zwischen Kompartimenten werden mit dieser Regel umgesetzt. Existieren ungeschützte und komplementäre Eingabe- bzw. Ausgabe-Präfixe in parallelen Bio-Prozessen mit nicht typdisjunkten elementaren β-Bindern, so findet eine Übertragung des zu sendenden Namens über eine entsprechende Substitution im empfangenden Teilprozess statt. Der gesendete Name sowie die verwendeten Kanäle müssen dabei in den jeweiligen Prozessen frei vorkommen.

BioPar, BioStruct: Ähnlich wie bei π-Prozessen ist auch bei Bio-Prozessen eine Reduktion innerhalb einzelner paralleler Komponenten erlaubt. Die Regel BIOPAR setzt dieses Verhalten um. Mit der Regel BIOSTRUCT werden sämtliche Umformungen gemäß der strukturellen Kongruenzrelation sowohl in Bio-Prozessen, wie auch in eingebetteten π^β-Prozessen realisiert.

Expose, Hide, Unhide: Die Binder eines Bio-Prozesses können wie bereits beschrieben durch diese Regeln gezielt manipuliert werden. Wichtig hierbei ist, dass die Subjekte der betroffenen Binder in den Prozessen frei vorkommen.

Join: Über diese Regel ist ein Zusammenschluss zweier Kompartimente möglich. Die Bedingungen einer solchen *join*-Operation werden über eine je nach Modellierung gewählte vierstellige Funktion f_{join} angegeben. Ist die Funktion für die beiden Binder und π^β-Prozesse der beteiligten Kompartimente definiert, so gibt der Funktionswert (ein Tripel) den Binder des neuen Kompartiments, sowie zwei auf die jeweiligen π^β-Prozesse anzuwendenden Substitutionen an. Andernfalls ist der Zusammenschluss nicht möglich.

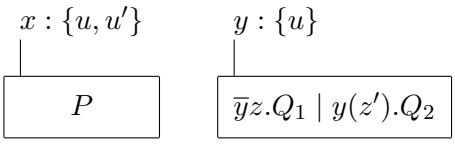
Split: Das Auftrennen eines Kompartiments in zwei Teile wird mit der Regel SPLIT verwirklicht, die auf ähnliche Weise wie JOIN parametrisch bezüglich einer auf den jeweiligen Anwendungsfall zugeschnittenen Funktion ist. In diesem Fall ist die Funktion (f_{split}) dreistellig und liefert für einen gegebenen Binder sowie zweier Teilprozesse ein Quadrupel bestehend aus den beiden Bindern der resultierenden Kompartimente und zwei Substitutionen, die auf die π^β-Teilprozesse anzuwenden sind. Auch die *split*-Operation lässt sich nicht anwenden, wenn f_{split} für das Binder-Prozess-Tripel eines Kompartiments undefiniert ist.

Beispiel 5.2.3: Dieses Beispiel veranschaulicht die Regeln JOIN und SPLIT anhand zweier einfacher Instanzen von f_{join} und f_{split}.

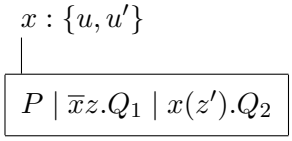
$$f_{\text{join}}(B_1, B_2, P_1, P_2) = \begin{cases} (B_1, \sigma_{id}, \{x/y\}), & \text{falls } B_1 = B_1^*\beta(x:\Gamma) \text{ und} \\ & B_2 = B_2^*\beta(y:\Gamma') \text{ und } \Gamma \cap \Gamma' \neq \emptyset \\ \bot, & \text{sonst.} \end{cases}$$

$$f_{\text{split}}(B, P_1, P_2) = \begin{cases} (B, B, \sigma_{id}, \sigma_{id}), & \text{falls } B = B^*\beta(x:\Gamma) \text{ und } P_2 = x(y).Q \\ \bot, & \text{sonst.} \end{cases}$$

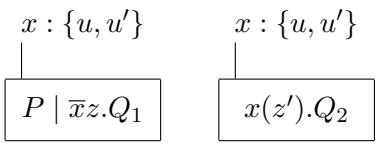
σ_{id} ist die Identitätssubstitution, die keine Namen ersetzt und $\bot$ steht für Undefiniertheit. Wir betrachten die Evolution des Bio-Prozesses



Nach einem Zusammenschluss der beiden Kompartimente ergibt sich der Bio-Prozess



Anschließend ist ein erneutes Auftrennen des Kompartiments mit dem Ergebnis



möglich. Vor beiden Ableitungsschritten wäre alternativ eine Kommunikation mittels der Regel INTRA realisierbar, durch die die split-Operation verhindert würde. □

Im Gegensatz zum klassischen π-Kalkül und zu bSπ lassen sich mit dem β-Binder-Kalkül auch Membranoperationen ohne großen Aufwand modellieren. Ein Bio-Prozess $B\langle P\rangle$ repräsentiert dabei auf natürliche Weise eine membranumschlossene Entität eines biologischen Systems oder auch ein beliebiges anderes abgeschlossenes Gebilde. Bisher erlaubt der Kalkül allerdings keine Modellierung quantitativer Aspekte, weshalb wir im folgenden Abschnitt eine stochastische Erweiterung angeben, die auf der Idee des biochemischen stochastischen π-Kalküls beruht.

5.3 Stochastische β-Binder

Der nun vorgestellte *Kalkül der stochastischen β-Binder* (kurz: Sβ) lässt sich mit dem Vorwissen der vorangehenden Abschnitte auf einfache Weise aus den klassischen β-Bindern entwickeln (siehe auch [DPPQ_06]). Die einzige syntaktische Anpassung, die wir vornehmen, ist die Erweiterung der Präfixe um Raten. Erneut stellt eine Rate $r \in \mathbb{R}^+ \cup \{\infty\}$ den Parameter einer Exponentialverteilung dar. Präfixe mit den

gleichen Kommunikationskanälen besitzen wie auch schon in bSπ die gleiche Rate. Die tatsächliche Rate einer Aktion hängt von der Basisrate des verwendeten Kanals und der Anzahl der ungeschützten Aktionspräfixe mit diesem Kanal im jeweiligen Prozess ab. Im Gegensatz zu bSπ unterscheiden wir nun zwischen drei Typen von Aktionen.

- Die *bimolekulare Aktion* findet zwischen zwei unterschiedlichen Teilprozessen statt, wobei sich die Rate aus dem Produkt $r = r_b * |P_1| * |P_2|$ ergibt. Die Basisrate des verwendeten Kanals wird durch r_b ausgedrückt, während $|P_1|$ und $|P_2|$ die Anzahl der jeweils möglichen Interaktionspartner bezeichnen (beispielsweise die Anzahl der ungeschützten Eingabe- bzw. Ausgabepräfixe mit dem verwendeten Kanal).

- Analog zu bSπ gibt es auch in Sβ eine *Homodimerisationsaktion*, bei der zwei gleichartige Teilprozesse mit Rate $r = 1/2 * r_b * |P| * |P - 1|$ interagieren.

- Als dritte Möglichkeit existiert in Sβ die *monomolekulare Aktion*, in die nur ein einzelner Teilprozess involviert ist. Die Rate wird durch das Produkt $r = r_b * |P|$ beschrieben.

Die semantischen Regeln des β-Binder-Kalküls bleiben intakt, bis auf dass die Reduktionen nun mit 5-Tupeln (t, S, r_b, r_i, r_o) beschriftet werden, über die die Berechnung der tatsächlichen Rate des Ableitungsschrittes möglich ist. Anstatt eine Menge $\mathcal{H} \subseteq \mathcal{N}$ von Namen anzugeben, die für Homodimerisationen reserviert sind, wird nun der Typ einer Aktion explizit im Parameter t mitgeführt. S bezeichnet den Teil-Bio-Prozess auf der linken Regelseite, der für die Zählarbeit zur Ermittlung der Ratenparameter r_i und r_o benötigt wird. r_b ist die Basisrate des verwendeten Kanals x (oder eine anderweitig berechnete Basisrate; siehe unten) und stellt mit r_i und r_o die Faktoren der tatsächlichen Rate dar. Ohne die Reduktionsregeln erneut anzugeben, beschreiben wir nachfolgend die Aktionstypen und Reduktionslabel für jede der Regeln aus Definition 5.2.2. Anschließend werden die benötigten Zählfunktionen In_x, Out_x, Hid_x, Unh_x, Count_t und Num definiert.

Intra: Die Intraaktion stellt stets eine monomolekulare Aktion dar, deren Reduktionslabel durch $(i, S, r_b * (1 + \mathrm{In}_x(P_3)) * (1 + \mathrm{Out}_x(P_3)), 1, 1)$ gegeben ist. (x ist der verwendete Kanal mit Basisrate r_b und P_3 kapselt sämtliche parallelen Komponenten, die nicht in die Regel involviert sind.)

Inter: Für die Interaktion werden zwei Fälle unterschieden. Falls $\beta(x : \Gamma)B_1^*\langle P\rangle \equiv \beta(x' : \Gamma')B_2^*\langle Q\rangle \equiv \beta(x : \Gamma)\beta(x' : \Gamma')B^*\langle P\rangle$, dann wird die Aktion als Homodimerisation aufgefasst und besitzt das Label $(I_h, S, \alpha(\Gamma, \Gamma')/2, 2, 1)$. Andernfalls handelt es sich um eine bimolekulare Aktion mit Label $(I, S, \alpha(\Gamma, \Gamma'), 1, 1)$. Die Funktion α ermöglicht die Modellierung einer Affinität zwischen zwei elementaren β-Bindern in Abhängigkeit ihrer Typen. Sie liefert eine Rate aus $\mathbb{R}^+ \cup \{\infty\}$, die die Basisrate der Kanäle ersetzt. Affinität zwischen bestimmten Molekülen ist ein häufiges biologisches Phänomen, dessen explizite Berücksichtigung in den semantischen Regeln eines Kalküls für viele Modellierungen von Vorteil sein kann. In Modellen, in denen Affinität keine Rolle spielt, lässt sich das Interaktionsverhalten von bSπ simulieren, indem man für die Subjekte zweier elementarer Binder mit nicht-disjunkten Typen Γ_1 und Γ_2 die gleiche Basisrate r wählt und $\alpha(\Gamma_1, \Gamma_2) = r$ definiert.

BioPar: Die Regel BioPar reicht das Label (t, S_p, r_b, n_1, n_2) der Reduktion in ihrer Prämisse weiter und aktualisiert dabei die letzten beiden Komponenten gemäß der Zählfunktion $\mathrm{Count}_t(S_p, T) = (n_1', n_2')$, wobei T die parallele Komponente des Bio-Prozesses auf der linken Regelseite ist, die nicht an der Aktion mit Typ t teilnimmt. Das neue Reduktionslabel berechnet sich zu $(t, S_p, r_b, n_1 + n_1', n_2 + n_2')$.

BioStruct: Die Regel BioStruct reicht das Label der Reduktion in ihrer Prämisse unverändert weiter.

Expose: Expose-Aktionen stellen monomolekulare Aktionen dar, die nicht von der Anzahl anderer ungeschützter exp-Präfixe abhängen. Das Reduktionslabel hat die Gestalt $(e, S, r_b, 1, 1)$.

Hide: Auch Hide-Aktionen sind monomolekular, hängen allerdings sehr wohl von der Anzahl anderer hid-Präfixe mit dem selben Namen ab. Das Reduktionslabel hat die Gestalt $(h, S, r_b * \mathrm{Hid}_x(P_2), 1, 1)$.

Unhide: Analog zu Hide-Aktionen sind Unhide-Aktionen monomolekular mit Label $(u, S, r_b * \mathrm{Unh}_x(P_2), 1, 1)$.

Join: Je nachdem, ob die beiden Kompartimente einer Join-Operation identische Entitäten repräsentieren ($B_1\langle P_1\rangle \equiv B_2\langle P_2\rangle$), kann auch die Reduktion mittels Join entweder als Homodimerisation mit Label $(J_h, S, c_j/2, 2, 1)$ oder als bimolekulare Aktion mit Label $(J, S, c_j, 1, 1)$ aufgefasst werden. Die Basisrate c_j wird durch die jeweilige Instanz der Funktion f_{join} definiert, die nun statt des gewohnten Tripels ein Quadrupel mit c_j als vierter Komponente liefert.

Split: Split-Operationen sind stets monomolekular mit Label $(s, S, c_s, 1, 1)$ und definieren ihre Basisrate c_s über die Funktion f_{split}, die nun ein um die Rate erweitertes 5-Tupel liefert.

Zusätzlich zu den bereits in der Semantik von bSπ definierten Zählfunktionen für jedes Präfix, die wir in Sβ um Hid_x und Unh_x erweitern, benötigen wir die Zählfunktionen Count_t und Num, die die Anzahl gleichartiger Teil-Bio-Prozesse bestimmen. Sei π ein beliebiges Präfix. Für $(G, \pi') \in \{(\mathrm{In}, x(y)), (\mathrm{Out}, \overline{x}y), (\mathrm{Hid}, \mathrm{hid}(x)), (\mathrm{Unh}, \mathrm{unh}(x))\}$ definieren wir

$$
\begin{aligned}
&\mathrm{G}_x(\mathbf{0}) = 0\\
&\mathrm{G}_x(\pi[r].P) = \begin{cases} 1 & \text{falls } \pi = \pi' \\ 0 & \text{sonst} \end{cases}\\
&\mathrm{G}_x(P \mid Q) = \mathrm{G}_x(P) + \mathrm{G}_x(Q)\\
&\mathrm{G}_x((\nu z)P) = \begin{cases} \mathrm{G}_x(P) & \text{falls } x \neq z \\ 0 & \text{sonst} \end{cases}\\
&\mathrm{G}_x(!P) = \mathrm{G}_x(P)\ .
\end{aligned}
$$

Die Funktionen Count_t und Num sind wie folgt definiert.

$$\mathrm{Count}_t(S,S') = \begin{cases} (\mathrm{Num}(S,S'),0) & \text{falls } t \in \{i,e,h,u,s\} \\ (\mathrm{Num}(B_1\langle P_1\rangle,S'),\mathrm{Num}(B_2\langle P_2\rangle,S')) & \text{falls } t \in \{I,J\} \\ & \text{und } S \equiv B_1\langle P_1\rangle \,||\, B_2\langle P_2\rangle \\ (\mathrm{Num}(B\langle P\rangle,S'),\mathrm{Num}(B\langle P\rangle,S')) & \text{falls } t \in \{I_h,J_h\} \\ & \text{und } S \equiv B\langle P\rangle \,||\, B\langle P\rangle \end{cases}$$

$$\mathrm{Num}(B\langle P\rangle,\mathbf{0}) = 0$$

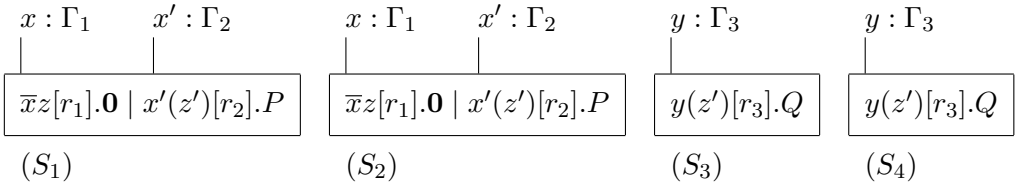
$$\mathrm{Num}(B\langle P\rangle,B'\langle P'\rangle \,||\, S) = \begin{cases} 1+\mathrm{Num}(B\langle P\rangle,S) & \text{falls } B\langle P\rangle \equiv B'\langle P'\rangle \\ \mathrm{Num}(B\langle P\rangle,S) & \text{sonst} \end{cases}$$

Die drei Fälle in der Definition von Count_t entsprechen jeweils monomolekularen Aktionen, bimolekularen Aktionen bzw. Homodimerisationen. Die Auswertung der Wettlaufbedingung und damit die Wahl des nächsten auszuführenden Ableitungsschrittes erfolgt auf die gleiche Weise wie bei bSπ über eine Adaption des Gillespie-Algorithmus mit dem einzigen Unterschied, dass die Ausgangsrate nun in Abhängigkeit der möglichen Reduktionsresultate und nicht nur der involvierten Kommunikationskanäle berechnet wird.

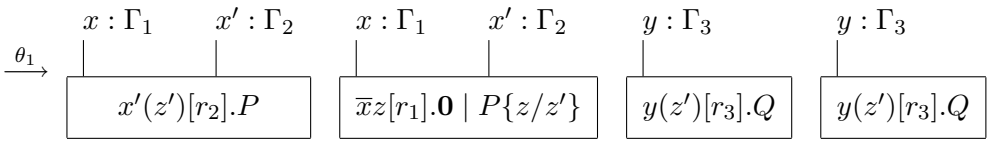
$$er(S) = \sum_{(S',(t,S'',r_b,r_i,r_o))\in\{(S',\theta)\,|\,S\xrightarrow{\theta}S'\}} r_b * r_i * r_o$$

Das folgende Beispiel verdeutlicht den Vorgang.

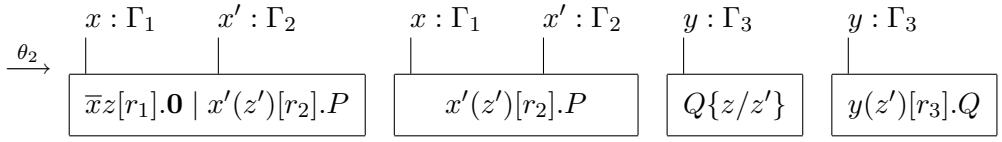
Beispiel 5.3.1: Wir betrachten die Reduktionsmöglichkeiten des Bio-Prozesses



mit $\alpha(\Gamma_1,\Gamma_2) = r_1$ und $\alpha(\Gamma_1,\Gamma_3) = r_3$. Unter der Annahme, dass keine split- oder join-Operationen möglich sind, ergeben sich zwei Reduktionen. Zwischen den ersten beiden Kompartimenten S_1 und S_2 kann eine Homodimerisations-Interaktion realisiert werden oder eines der letzten beiden Kompartimente S_3 bzw. S_4 kann über eine bimolekulare Interaktion einen Namen von einem der ersten beiden Kompartimente empfangen. Im ersten Fall ergibt sich der Bio-Prozess



mit $\theta_1 = (I_h, S_1 \,||\, S_2, r_1/2, 2, 1)$. Im zweiten Fall gibt es vier mögliche Paare von Interaktionspartnern, die alle zu strukturell äquivalenten Resultaten führen. Wir betrachten daher beispielhaft die Interaktion zwischen S_2 und S_3.

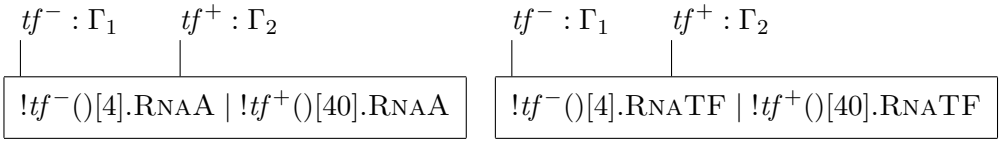


mit $\theta_2 = (I, S_2 \,||\, S_3, r_3, 2, 2)$. Die Regel INTER setzt die letzten beiden Komponenten des Labels zunächst auf 1. Bei Anwendung der Regel BIOPAR wird durch die Funktion Count_I in den Teilprozessen S_1 und S_4 nach Kompartimenten gesucht, die äquivalent zu S_2 und S_3 sind. Von beiden Typen wird ein weiteres Exemplar gefunden, so dass die letzten beiden Komponeten des Labels jeweils auf 2 inkrementiert werden. Für $r_1 = r_3 = 1$ ergibt sich die Ausgangsrate $\frac{r_1}{2} * 2 * 1 + r_3 * 2 * 2 = 5$ Die Wahrscheinlichkeit für eine Reduktion nach dem ersten Fall beträgt 20%, während sich für eine Reduktion gemäß dem zweiten Fall eine Wahrscheinlichkeit von 80% ergibt. Im zweiten Fall wird eines der vier Interaktions-Paare mit jeweils gleicher Wahrscheinlichkeit gewählt. □

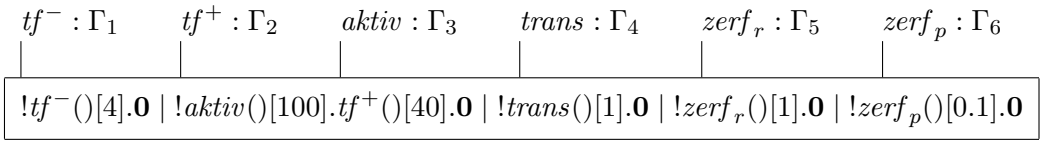
Sowohl in der stochastischen wie auch in der ursprünglichen Variante lässt sich der β-Binder Kalkül mit polyadischen Eingabe- und Ausgabe-Präfixen verwenden. Ein Verzicht auf den Replikationsoperator zu Gunsten von Definitionsgleichungen erscheint allerdings nicht sinnvoll, da sich aufgrund des fehlenden Operators für alternative Teilprozesse eine umständliche und schlecht lesbare Implementierung von Alternativen ergäbe.

5.4 Anwendungsbeispiel: Genexpression mit Kompartimenten

In diesem Abschnitt wenden wir uns einer Modellierung des Genexpressionsbeispiels aus den Abschnitten 3.4 und 4.3 mittels einer polyadischen Variante des stochastischen β-Binder Kalküls zu. Das Modell soll die gleichen quantitativen Resultate liefern, wie das bSπ-Modell und sich dabei wesentlich übersichtlicher und intuitiver präsentieren. Wir geben einige Teilprozesse in separaten Gleichungen an, die hier jedoch als Makros zu verstehen sind und nicht als Definitionsgleichungen, die eine entsprechende Berücksichtigung in der strukturellen Kongruenzrelation benötigen würden. Man beachte, dass die Gleichungen nicht parametrisch sind und keine Rekursion beinhalten. Wir definieren Sβ-SYS wie folgt:



(GENA) (GENTF)



(UMWELTPROZESS)

$$
\begin{aligned}
\textsc{RnaA} &= !\mathit{trans}()[1].\textsc{ProteinA} \mid \mathit{zerf}_r()[1].\mathrm{hid}(\mathit{trans})[\infty].\mathrm{hid}(\mathit{zerf}_r)[\infty].\mathbf{0} \\
\textsc{ProteinA} &= !\overline{\mathit{bb}_2}(\mathit{aktiv})[10].\mathbf{0} \mid \mathit{zerf}_p()[0.1].\mathrm{hid}(\mathit{zerf}_p)[\infty].\overline{\mathit{bb}_3}()[\infty].\mathbf{0} \\
\\
\textsc{RnaTF} &= !\mathit{trans}()[1].\textsc{ProteinTF} \mid \mathit{zerf}_r()[1].\mathrm{hid}(\mathit{trans})[\infty].\mathrm{hid}(\mathit{zerf}_r)[\infty].\mathbf{0} \\
\textsc{ProteinTF} &= \mathit{bb}_2(\mathit{aktiv}')[10].!\overline{\mathit{aktiv}'}()[100].\mathbf{0} \mid \mathit{bb}_3()[\infty].\mathbf{0} \mid \\
&\quad \mathit{zerf}_p()[0.1].\mathrm{hid}(\mathit{zerf}_p)[\infty].\mathrm{hid}(\mathit{aktiv}')[\infty].\mathbf{0} \\
\textsc{AktivTF} &= !\overline{\mathit{aktiv}}()[100].\mathbf{0} \mid \mathit{bb}_3()[\infty].\mathbf{0} \mid \\
&\quad \mathit{zerf}_p()[0.1].\mathrm{hid}(\mathit{zerf}_p)[\infty].\mathrm{hid}(\mathit{aktiv})[\infty].\mathbf{0}
\end{aligned}
$$

Für sämtliche Kanäle x mit zugeordneter Basisrate r_b gilt: falls es einen elementaren Binder $\beta(x:\Gamma)$ gibt, so ist $\alpha(\Gamma,\Gamma) = r_b$. Für alle Typen Γ_i, Γ_j mit $i \neq j$ definieren wir $\alpha(\Gamma_i,\Gamma_j) = 0$. Die Funktionen f_{join} und f_{split} sind wie folgt definiert.

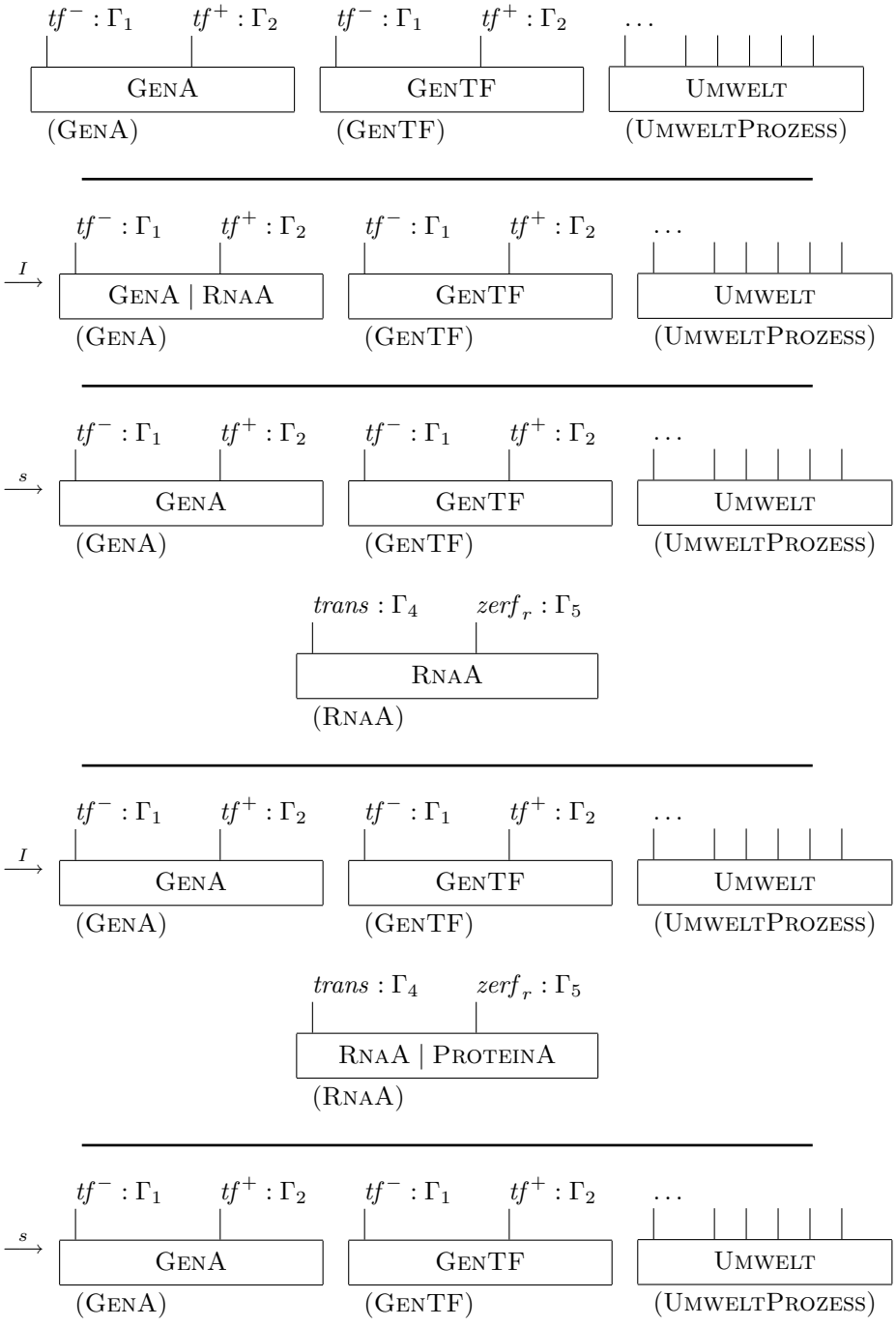
$$
f_{\mathrm{join}}(B_1,B_2,P_1,P_2) = \begin{cases} (B_1,\sigma_{id},\{\mathit{zerf}'_p/\mathit{zerf}_p\},0.1)\,, \\ \qquad\qquad \text{falls } P_1 = \textsc{ProteinA} \wedge P_2 = \textsc{ProteinTF} \\ \bot\,, \qquad \text{sonst.} \end{cases}
$$

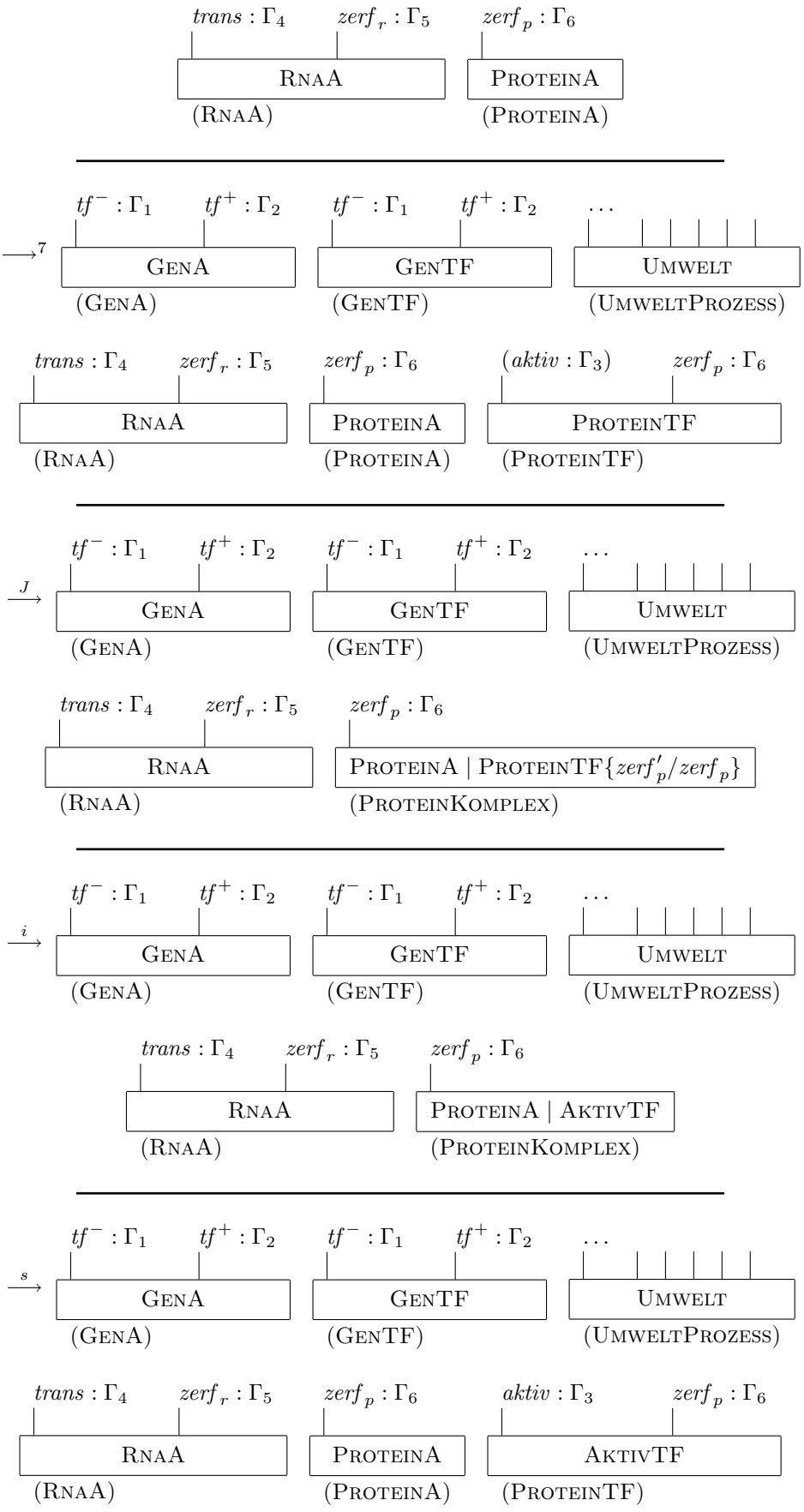
$$
f_{\mathrm{split}}(B,P_1,P_2) = \begin{cases} (B,\beta(\mathit{trans}:\Gamma_4)\beta(\mathit{zerf}_r:\Gamma_5),\sigma_{id},\sigma_{id},\infty)\,, \\ \qquad \text{falls } B = B^*\beta(\mathit{tf}^-:\Gamma_1) \wedge P_2 \in \{\textsc{RnaA},\textsc{RnaTF}\} \\ (B,\beta(\mathit{zerf}_p:\Gamma_6),\sigma_{id},\sigma_{id},\infty)\,, \\ \qquad \text{falls } B = B^*\beta(\mathit{trans}:\Gamma_4) \wedge P_2 = \textsc{ProteinA} \\ (B,\beta^h(\mathit{aktiv}:\Gamma_3)\beta(\mathit{zerf}_p:\Gamma_6),\sigma_{id},\sigma_{id},\infty)\,, \\ \qquad \text{falls } B = B^*\beta(\mathit{trans}:\Gamma_4) \wedge P_2 = \textsc{ProteinTF} \\ (B,\beta(\mathit{aktiv}:\Gamma_3)\beta(\mathit{zerf}_p:\Gamma_6),\sigma_{id},\{\mathit{zerf}_p/\mathit{zerf}'_p\},10)\,, \\ \qquad \text{falls } B = \beta(\mathit{zerf}_p:\Gamma_6)\wedge \\ \qquad P_2 \in \{\textsc{ProteinTF}\{\mathit{zerf}'_p/\mathit{zerf}_p\},\textsc{AktivTF}\} \\ \bot\,, \qquad \text{sonst.} \end{cases}
$$

In der Ausgangssituation setzt sich das System aus drei Kompartimenten zusammen. Die ersten beiden repräsentieren Gen A und Gen TF, während das dritte die vier Umweltprozesse der bSπ-Modellierung vereint. Man beachte, dass alternative Teilprozesse durch modifizierte parallele Teilprozesse ersetzt worden sind, die das gleiche Systemverhalten realisieren. Die Gen-Kompartimente können sich nach Aktivierung über einen der Kanäle tf^- oder tf^+ in ein Gen- und ein RNA-Kompartiment aufsplitten. Hierbei kommt der erste Fall in der Definition der f_{split}-Funktion zum Einsatz. Auf ähnliche Weise können sich die RNA-Kompartimente in ein RNA- und ein Protein-Kompartiment aufsplitten. Ein Protein A und ein Protein TF können sich zu einem Komplex-Kompartiment verbinden und dann über den Kanal bb_2 die Aktivierung des Proteins TF mittels einer Intraaktion bewirken. Danach splitten sich die Kompartimente wieder in zwei getrennte Proteine auf. Der Zerfall eines Moleküls wird über das Verstecken sämtlicher Binder des entsprechenden Kompartiments realisiert. Da in der Modellierung keine exp-Präfixe zur Binder-Generierung und keine unh-Präfixe zum Aktivieren von versteckten Bindern vorkommen, und Kompartimente ohne Binder nicht an join- oder split-Operationen beteiligt sein können, sind solche Kompartimente zu keiner weiteren Interaktion mit ihrer Umgebung fähig und

werden mit dem untätigen Bio-Prozess gleichgesetzt, der über die strukturelle Kongruenz eliminiert werden kann.

Beispiel 5.4.1: Wir betrachten eine Beispielableitung die der Ableitung aus Beispiel 4.3.1 entspricht. Die Prozesse in den drei initialen Kompartimenten kürzen wir durch die Makros GenA, GenTF bzw. Umwelt ab. Von den Reduktionslabeln wird jeweils nur die erste Komponente, die den Typ der Aktion spezifiziert, angegeben. Des Weiteren schreiben wir $\longrightarrow^n$ $(n > 1)$ um mehrere Reduktionen in einem Schritt umzusetzen. Die letzten beiden Schritte aus Beispiel 4.3.1 lassen wir in dieser Ableitung weg.




trans : Γ4
zerf_r : Γ5
zerf_p : Γ6
RnaA
ProteinA
(RnaA)
(ProteinA)
tf− : Γ1
tf+ : Γ2
...
7
GenA
GenTF
Umwelt
(GenA)
(GenTF)
(UmweltProzess)
(aktiv : Γ3)
ProteinTF
(ProteinTF)
J
ProteinA | ProteinTF{zerf′_p/zerf_p}
(ProteinKomplex)
i
AktivTF
s

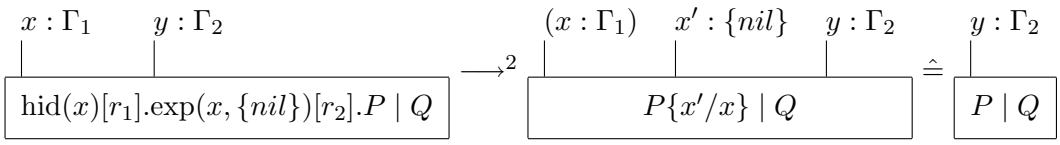
Besonders zu erwähnen ist der fünfte Schritt, der zur Verkürzung des Beispiels sieben Reduktionen zusammenfasst. Dabei wird in je zwei Reduktionen ein RnaTF-Molekül, sowie anschließend ein ProteinTF-Molekül produziert, im Anschluss zerfällt das RnaTF-Molekül in drei Reduktionen. Im sechsten Schritt bildet sich ein Proteinkomplex, wobei der Kanal $zerf_p$ im ProteinTF-Prozess umbenannt wird, um einen separaten Zerfall dieses Moleküls zu verhindern. Stattdessen kann ein gemeinsamer Zerfall der beiden Proteine über Kanal bb_3 umgesetzt werden. □

5.5 Mustermodellierungen biologischer Phänomene

In diesem Abschnitt untersuchen wir die Eignung von Sβ für systembiologische Modellierungen anhand einiger typischer Phänomene aus der Zellbiologie. Wir geben jeweils eine allgemeine, abstrakte und möglichst anpassungsfähige Implementierung in unserem Kalkül an und diskutieren mögliche Probleme und Alternativen.

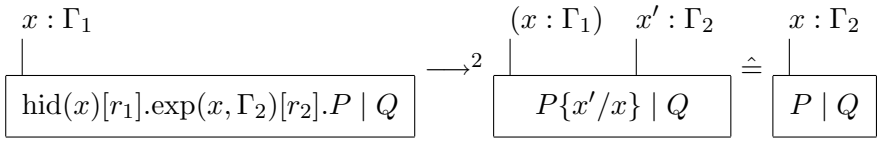
5.5.1 Modifikation von Binder-Typen

In Sβ gibt es keine Möglichkeit, elementare Binder zu löschen oder deren Typ zu modifizieren. Diese Operationen sind bei der Modellierung von Proteininteraktionen von großer Relevanz, da die Bindungsstellen eines Proteins nach einer Phosphorilierung oder der Einwirkung von Kinasen ein völlig verändertes Verhalten aufweisen können. Eine einfache Realisierung des Löschens von Bindern besteht im Verstecken des Binders über ein hid-Präfix. Allerdings kann dies in einer größeren Modellierung zu einer sehr unübersichtlichen Darstellung führen. Um herauszufinden, ob ein Binder tatsächlich als gelöscht gelten soll oder bloß als versteckt, ist es notwendig, den eingebetteten π^β-Prozess nach entsprechenden unh-Präfixen zu durchsuchen. Außerdem müssen die Instanzen von f_{join} und f_{split} auf die Möglichkeit der Generierung eines entsprechenden Binders geprüft werden. Eine elegantere Implementierung, die gleichzeitig beliebige Typänderungen eines Binders ermöglicht, besteht in der Kombination eines hid- und eines exp-Präfixes mit gleichem Subjekt. Im Fall einer Löschung enthält das exp-Präfix den Typ $\{nil\}$, im Fall einer Typänderung den neuen Typ. Um Verwechslungen mit einer Implementierung zu vermeiden, in der tatsächlich bloß ein Verstecken des Binders und Produzieren eines anderen, davon unabhängigen Binders gemeint ist, legen wir fest, dass in letztgenanntem Fall die Subjekte der Präfixe verschieden sein müssen. Als Beispiel für eine Binder-Löschung betrachte man die Ableitung

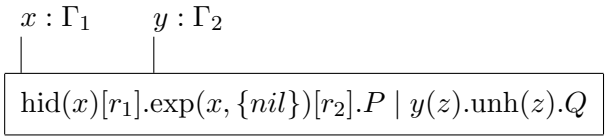


Das Subjekt des neuen Binders wird mittels α-Konversion zu x' geändert, um einen Namenskonflikt mit dem bestehenden Binder zu verhindern. Wichtig ist hierbei, dass der Name x in Q nicht mehr vorkommt, da sonst eventuell eine weitere Verwendung des versteckten Binders mit Typ x, der als gelöscht betrachtet werden soll, möglich ist. In der Darstellung lassen wir den versteckten Binder, sowie den Binder mit Typ $\{nil\}$ weg und verwenden den alten Namen x als Subjekt des neuen Binders,

um die Übersichtlichkeit zu steigern. Für die Illustrierung einer Typänderung diene folgendes Beispiel:



Problematisch bei der beschriebenen Implementierung sind eventuell vorkommende unh-Präfixe, die ihr Subjekt über Interaktionen mit anderen Bio-Prozessen erhalten. Der Prozess



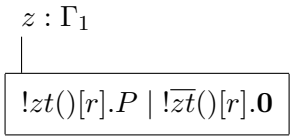
könnte beispielsweise nach Löschung des Binders durch seine erste Komponente den Namen x über einen anderen Bio-Prozess empfangen und anschließend den eigentlich bloß versteckten Binder wieder aktivieren. Auf die Kommunikation von Binder-Subjekten sollte daher völlig verzichtet werden.

Wird die hier vorgeschlagene Verfahrensweise in einem Modell konsequent eingesetzt, so kann zur weiteren Vereinfachung die Kombination aus hid- und exp-Präfix $\mathrm{hid}(x)[r_1].\mathrm{exp}(x, \Gamma)[r_2]$ durch ein Makro-Präfix $\mathrm{del}(x)[r_1]$ (*delete*) bzw. $\mathrm{cht}(x, \Gamma)[r_1]$ (*change type*) abgekürzt werden. Um zu gewährleisten, dass die Ausführung von hid und exp nicht unterbrochen werden kann, wird die Rate r_2 des exp-Präfixes stets auf ∞ gesetzt.

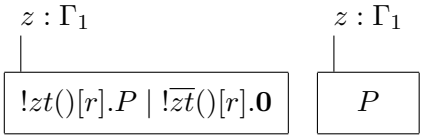
5.5.2 Zellteilung

„Zellen entstehen immer aus Zellen." ([PlHe_06]) Dies ist eines der Hauptcharakteristika lebender Zellen und eine der wesentlichsten Feststellungen in der Zellbiologie. Zellen vermehren sich, indem sie ihre Organellen (insbesondere den Zellkern mit dem Genom) reproduzieren und sich dann in zwei getrennte Zellen mit jeweils der Hälfte der Organellen und je einem Zellkern aufteilen. Ein theoretischer Ansatz, der speziell für die Aufgabe der ganzheitlichen Modellierung des Verhaltens von Zellen entwickelt wurde, muss auch eine Möglichkeit zur Formalisierung von Zellteilung aufweisen.

Auf den ersten Blick scheint der Replikationsoperator in Sβ dafür hervorragend geeignet zu sein. Bei näherer Betrachtung lässt sich jedoch ein gravierendes Problem feststellen. Man betrachte den Bio-Prozess

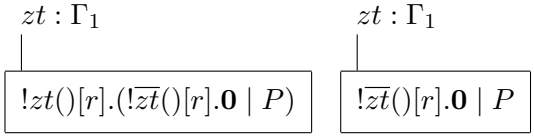


der stellvertretend für eine Zelle mit Verhalten P steht. Eine Zellteilung wird nach der Replikation beider paralleler Komponenten durch eine Kommunikation über den Kanal zt eingeleitet und mittels einer split-Operation ausgeführt. (Es gelte $f_{\mathrm{split}}(B, P, P') = (B, B, \sigma_{id}, \sigma_{id}, r')$ mit $P' = !zt()[r].P \mid !\overline{zt}()[r].\mathbf{0}$.) Der resultierende Bio-Prozess

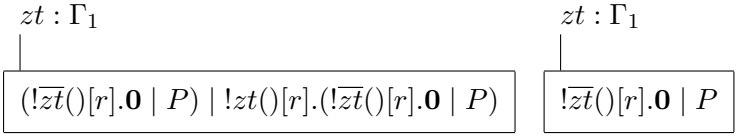


besteht aus zwei Kompartimenten, von denen das erste die unveränderte Zelle darstellt, während das zweite ebenfalls das Verhalten P der beschriebenen Zelle hat, allerdings nicht zur Zellteilung fähig ist, da der Replikationsoperator nicht mitrepliziert werden kann. Dieses Verhalten entspricht also nicht der Vorstellung einer Zellteilung mit dem Ergebnis zweier identischer Zellen. Tatsächlich ist es nicht möglich, ein Kompartiment anzugeben, das sich selbst exakt repliziert.

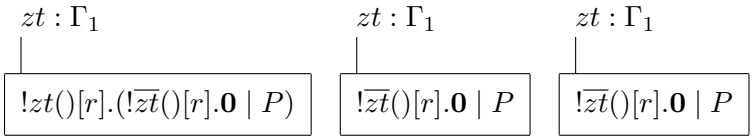
Die Lösung des Problems besteht in der Implementierung eines Umweltprozesses als zusätzliches Kompartiment, welches die neue Zelle auf Anforderung einer bereits existierenden Zelle herstellt. In folgendem Beispiel stellt das erste Kompartiment den Umweltprozess und das zweite die Zelle dar.



Nach Entfaltung der äußeren Replikation im Umweltprozess und Kommunikation über zt ergibt sich der Bio-Prozess



Der Umweltprozess trennt nun eine exakte Kopie der Zelle über eine split-Operation ab. (Es gelte $f_{\text{split}}(B, P_1, P_2) = (B, B, \sigma_{id}, \sigma_{id}, r')$ mit $P_1 = !zt()[r].(!\overline{zt}()[r].\mathbf{0} \mid P)$ und $P_2 = !\overline{zt}()[r].\mathbf{0} \mid P$.)



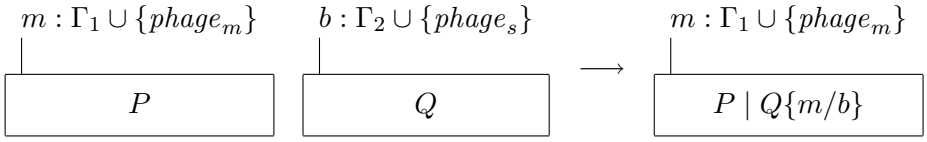
5.5.3 Endocytose und Exocytose

Während sich die Membranoperationen Mate und Mito, also das Verschmelzen bzw. Auftrennen zweier Zellen oder membranumhüllter Organellen direkt über entsprechende Instanzen der f_{join} und f_{split} Funktionen realisieren lassen, erfordern die Operationen der Endocytose und Exocytose, die ein Verschlingen bzw. Ausspeien von Materialien durch eine Zelle darstellen, eine genauere Betrachtung. In Sβ wie auch im klassischen β-Binder Kalkül ist das Vorkommen von Kompartimenten innerhalb eines übergeordneten Kompartiments nicht gestattet. Verschachtelte Membranstrukturen lassen sich jedoch auf einfache Weise mit der gegebenen Syntax simulieren. Um die Modellierung unabhängig von den eingebetteten π^{β}-Prozessen zu machen,

definieren wir zwei Mengen von Namen $\mathcal{M}, \mathcal{S} \subseteq \mathcal{N}$, die nur in den Typen von Bindern Verwendung finden. Alle Namen in $\mathcal{M}$ tragen den Index m (für *Master*), um zu kennzeichnen, dass Binder, die einen solchen Namen im Typ enthalten, in der Lage sind, andere Kompartimente zu verschlingen. Die Namen in $\mathcal{S}$ tragen den Index s (für *Slave*) und kommen in Bindertypen zum Einsatz, die ein zu verschlingendes Kompartiment markieren. Wir legen fest, dass die übrigen Namen in $\mathcal{N}$ keinen der Indices m oder s tragen dürfen, um unnötige Verwirrung zu vermeiden. Es gilt $x_m \in \mathcal{M} \iff x_s \in \mathcal{S}$ und außerdem $f_{\text{join}}(B_1, B_2, P_1, P_2) = (B_1, \sigma_{id}, \{x_1/x_2\})$, falls $B_1 = B_1^*\beta(x_1 : \Gamma_1 \cup \{z_m\})$ und $B_2 = B_2^*\beta(x_2 : \Gamma_2 \cup \{z_s\})$. Ein Kompartiment, das in einem seiner Bindertypen den Namen $z_m \in \mathcal{M}$ besitzt, kann also jedes andere Kompartiment mit einem Namen $z_s \in \mathcal{S}$ in einem seiner elementaren Binder verschlingen. Dabei wird im Slave-Kompartiment eine Namensanpassung vorgenommen, so dass die π^β-Prozesse der beiden Kompartimente nach der Endocytose über die Subjekte der betroffenen Binder kommunizieren können.

Die Exocytose-Operation wird über eine entsprechende Instanz der Funktion f_{split} realisisert. Dabei muss der Teilprozess, der das Kompartiment verlässt, mit passenden Bindern versehen werden. Es ist durchaus denkbar, dass ein verschlungenes Kompartiment in veränderter Form wieder ausgespien wird. Daher ist es nicht möglich oder sinnvoll eine allgemeine Definition für die f_{split} Funktion anzugeben.

Beispiel 5.5.1: Wir betrachten ein kleines Beispiel aus der Zellbiologie. Eine Makrophage ist eine Zelle des Immunsystems, die andere Zellen oder Substanzen, die für den Organismus schädlich sind, aufspüren und verschlingen kann. In folgendem Prozess stellt das erste Kompartiment eine solche Makrophage dar, während das zweite Kompartiment ein schädliches Bakterium darstellt, dass neutralisiert werden soll.

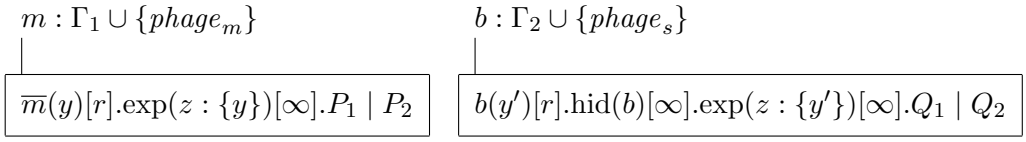


Wie bereits erwähnt, spielen die π^β-Prozesse der Kompartimente bei dieser Operation keine Rolle. Intuitiv entspricht dies der Gegebenheit, dass es einer Makrophage relativ egal ist, welches Verhalten ein feindliches Bakterium zum Zeitpunkt des Verschlingens aufweist. □

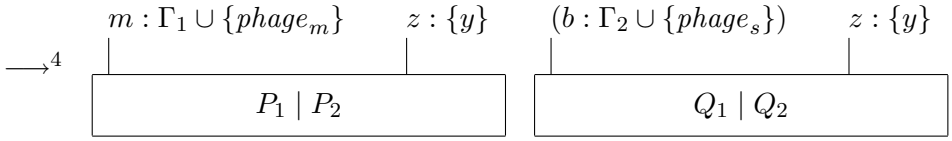
Diese Herangehensweise empfiehlt sich besonders bei Modellen, in denen flache Hierarchien von sich gegenseitig verschlingenden Entitäten bestehen, wie in dem Makrophagenbeispiel. In einer Anwendung, in der eine verschlungene Entität ebenfalls untergeordnete Entitäten enthalten kann, ist es sinnvoll, die jeweiligen Kompartimente intakt zu lassen. Wir behalten die Konvention für die Namens-Teilmengen $\mathcal{M}$ und $\mathcal{S}$ bei, heben allerdings die Vereinbarung für die Funktion f_{join} aus dem letzten Absatz auf. Stattdessen modellieren wir die Endocytose nun über die eingebetteten π^β-Prozesse. Beide involvierten Kompartimente erzeugen einen neuen Binder, in dessen Typ nur Namen vorkommen, die in sämtlichen anderen Bindern des Modells nicht auftauchen. Das verschlungene Kompartiment versteckt zusätzlich seine übrigen Binder, so dass es nur noch mit dem nun übergeordneten Kompartiment kommunizieren kann. Die Aktion wird zunächst über eine Interaktion, bei der der

Typ des jeweils zu erzeugenden Binders ausgetauscht wird, eingeleitet.

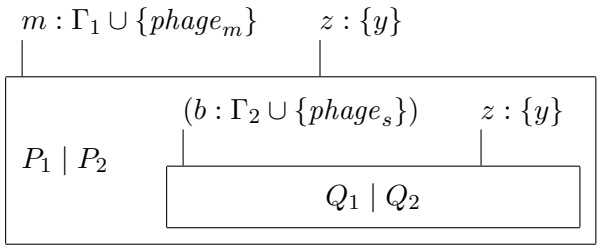
Beispiel 5.5.2: Wir modellieren zur Veranschaulichung erneut das Makrophagenbeispiel, obwohl die Hierarchie hier sehr flach ist.



Nach der Endocytose ergibt sich der Bio-Prozess



Um auch in komplexeren Beispielen eine übersichtliche Darstellung zu gewährleisten, kann die Hierarchie auf folgende Weise grafisch umgesetzt werden:



□

Die Einbeziehung der jeweiligen π^β-Prozesse in die Endocytose-Operation könnte als Nachteil aufgefasst werden, bietet aber andererseits eine feinere Kontrolle über den Vorgang. In den Prozessen lässt sich wie im Beispiel gezeigt eine Fallunterscheidung implementieren, die ein Verhalten für den Fall des Enthaltenseins in einer anderen Entität (im Beispiel Q_1) oder den Fall der Autonomie (im Beispiel Q_2) spezifiziert. Gegebenenfalls ist der Teilprozess Q_2 so zu implementieren, dass er nach der Endocytose zu keiner direkten Reduktion fähig ist.

Auch bei dieser hierarchischen Modellierung mittels spezieller Binder empfiehlt es sich nicht, eine allgemeine Vorgehensweise für die Exocytose anzugeben. Eventuell reicht es aus, die versteckten Binder des untergeordneten Kompartiments wieder zu aktivieren und im Gegenzug die bei der Endocytose gebildeten Binder zu verstecken. In anderen Fällen könnte es notwendig sein, ein Kompartiment mit komplett neuen Bindern auszustatten, um einer Modifikation des Kompartiments durch die übergeordente Entität gerecht zu werden.

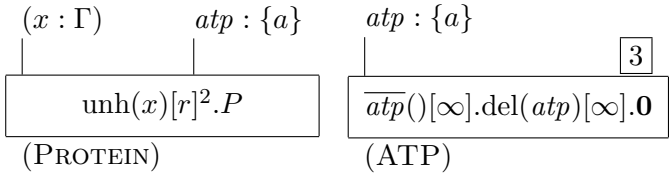
5.5.4 Energiehaushalt in Zellen

Bisher haben wir das Verhalten von biologischen Entitäten völlig entkoppelt von den verfügbaren Ressourcen in ihrem Umfeld betrachtet. In Sβ lassen sich jedoch auch solche Einflüsse problemlos integrieren. Jeglicher Lebensprozess, der als Informationsverarbeitung aufgefasst werden kann, verbraucht irgendeine Form von Energie.

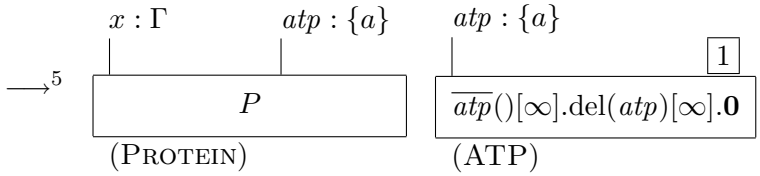
In Zellen wird diese Energie über das Molekül ATP (*Adenosintriphosphat*) bereitgestellt. Unter Mitwirkung der Mitochondrien wird dieses mit Nukleotiden vergleichbare Molekül aus Glukose gewonnen und in das Cytosol abgegeben. Energieverbrauchende Entitäten binden ATP, spalten eine Phosphatgruppe des Moleküls ab und nutzen die dabei frei werdende Energie für verschiedenste Zwecke. Zwei besonders bedeutsame Beispiele sind die Synthetisierung oder der Abbau von Molekülen (chemische Energie) und die Deformation von Molekülen z.B. bei der Kontraktion von Muskelzellen (kinetische Energie).

Wir wollen eine allgemeine Implementierung des Energiehaushalts einer Zelle angeben und am Beispiel der Proteinfaltung veranschaulichen. Jedes ATP-Molekül wird durch ein Kompartiment dargestellt, das über die Kommunikation mittels seines einzigen Binders zur Auflösung gebracht werden kann. Die Auflösung wird dabei durch das Löschen des Binders gemäß der Vorgehensweise aus Abschnitt 5.5.1 beschrieben. Ein anderes Kompartiment, welches Energie verbraucht, besitzt einen elementaren Binder, der mit dem ATP-Binder kompatibel ist. Jedes Präfix, das eine energieverbrauchende Aktion einleitet, wird von einer bestimmten Anzahl Eingabe-Präfixe über den ATP-Binder (mit unendlicher Rate) gefolgt, die dem Energieverbrauch der Aktion entsprechen. Der Einfachheit halber geben wir die Eingabe-Präfixe nicht explizit an, sondern versehen das einleitende Präfix mit einem Exponenten, der die Anzahl der benötigten ATP-Moleküle spezifiziert. Beispielsweise steht $x(y)[r]^3$ für $x(y)[r].atp()[\infty].atp()[\infty].atp()[\infty]$.

Beispiel 5.5.3: Wir betrachten ein Protein, das die Energie von zwei ATP-Molekülen benötigt, um eine im Inneren befindliche Bindungsstelle an die Oberfläche zu bringen. Der Bio-Prozess



modelliert das Protein und drei ATP-Moleküle, die als ein Kompartiment mit einem die Anzahl angebenden Index in der rechten oberen Ecke dargestellt sind. Das Protein aktiviert zunächst seinen versteckten Binder und empfängt anschließend zwei Nachrichten von zwei verschiedenen ATP-Bio-Prozessen, welche ihre Binder verstecken, daher zu keiner Aktion mehr fähig sind und in unserer Darstellung vernachlässigt werden.



□

Eine energieverbrauchende Aktion wird nach diesem Modellierungsansatz auch dann eingeleitet, wenn nicht genügend ATP zu ihrer Vollendung verfügbar ist. In dem Beispiel würde der versteckte Binder des Protein-Prozesses auch dann aktiviert, wenn nur ein oder überhaupt kein ATP-Prozess existierte. Im Regelfall stellt dies

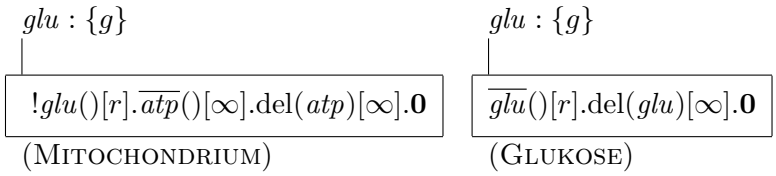
kein Problem dar, denn das Folgeverhalten des Prozesses wird durch die übriggebliebenen ATP-Eingabe-Präfixe behindert. Bei Energiemangel wird die Evolution des Systems also nicht augenblicklich gestoppt, sondern es werden noch einige Schritte durchgeführt, bis schließlich sämtliche Prozesse durch vorangestellte ATP-Eingabe-Präfixe blockiert sind.

Für Systeme, in denen es auf eine exakte Betrachtung von Zuständen des Energiemangels ankommt, besteht eine Alternative in der Umkehrung der Präfix-Reihenfolge. Die ATP-Eingabe-Präfixe werden dem Präfix, das eine Aktion einleitet, vorangestellt, so dass $x(y)[r]^3$ nun für $atp()[\infty].atp()[\infty].atp()[\infty].x(y)[r]$ steht. In diesem Fall ergibt sich ein starker Konkurrenzkampf um das verfügbare ATP. Eine Aktion kann selbst dann scheitern, wenn ausreichend ATP für ihre Durchführung vorhanden ist. In einem System, das zwei Proteine gemäß dem Beispiel 5.5.3 sowie zwei ATP-Prozesse enthält, können beide Proteine je ein ATP verbrauchen und sind anschließend beide blockiert. Welche Präfix-Reihenfolge in einer Implementierung zum realistischeren Verhalten führt, muss von System zu System entschieden werden.

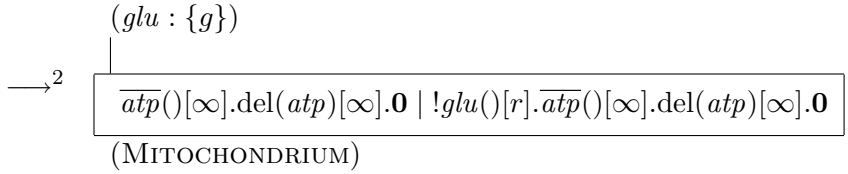
Ein weiterer interessanter Aspekt ist die Abhängigkeit der Evolutionsgeschwindigkeit eines biologischen Systems von der verfügbaren Energie. In unserer Implementierung finden alle Aktionen mit ihren jeweiligen Raten statt, unabhängig davon, ob viel oder wenig ATP zur Verfügung steht. Erst bei völliger Abwesenheit von ATP kommt die Evolution zum Stillstand. Versieht man die ATP-Eingabe-Präfixe mit einer Rate ungleich unendlich, so wird die Geschwindigkeit und Häufigkeit von Aktionen mit sinkender ATP-Molekülanzahl abfallen, da nach Gillespies Algorithmus die tatsächliche Rate von der Anzahl möglicher Interaktionspartner abhängt.

Abschließend geben wir noch eine Möglichkeit an, die ATP-Produktion zu modellieren. Ähnlich der Zellteilungsmodellierung aus Abschnitt 5.5.2 erzeugen wir ATP-Kompartimente mittels split-Operationen aus einem Produktions-Prozess.

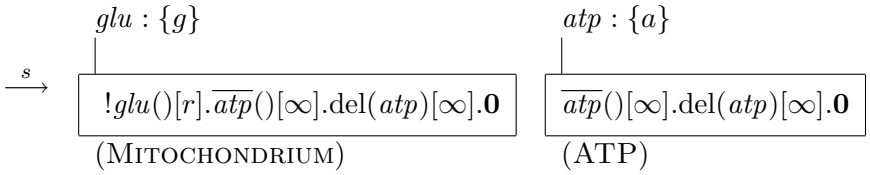
Beispiel 5.5.4: Der folgende Bio-Prozess stellt ein Mitochondrium dar, das aus Glukose-Molekülen ATP erzeugt.



Der Glukose-Prozess wird analog zu den ATP-Prozessen verbraucht, indem er seinen Binder löscht und wird fortan nicht mehr dargestellt.



Im letzten Schritt teilt sich der ATP-Prozess vom Mitochondrium ab. Wir verwenden $f_{\text{split}}(B, P_1, P_2) = (B, \beta(glu : \{g\}), \sigma_{id}, \sigma_{id}, r')$ mit $P_1 = \overline{atp}()[\infty].\text{del}(atp)[\infty].\mathbf{0}$ und $P_2 = !glu()[r].\overline{atp}()[\infty].\text{del}(atp)[\infty].\mathbf{0}$.



Das Beispiel stellt eine grobe Vereinfachung der realen ATP-Produktion dar. Glukose wird bereits im Cytosol einer Zelle verarbeitet und lediglich die anfallenden Nebenprodukte werden durch Mitochondrien weiter zerlegt. Dabei wird insgesamt wesentlich mehr als ein ATP-Molekül pro Glukose-Molekül erzeugt. □

Kapitel 6

Vergleich der Ansätze

Nachdem wir nun einen knappen Einblick in die aktuelle Forschung auf dem Gebiet der Systembiologie vermittelt und einige theoretische Ansätze für Modellierungen biologischer Systeme kennengelernt haben, stellt sich die Frage nach den Vor- und Nachteilen dieser Ansätze. Ist mit dem Kalkül der β-Binder eine ganzheitliche Beschreibung der komplexen Vorgänge in Zellen auf intuitive und effiziente Weise möglich, oder sind weitere Erweiterungen zur Erreichung dieses Ziels nötig? Sind speziellere Ansätze wie der Brane-Kalkül oder der κ-Kalkül, die in Kapitel 2 skizziert wurden, eventuell besser zu diesem Zweck geeignet? Im Abschnitt 6.1 ziehen wir zunächst die in Abschnitt 2.2.1 eingeführten Eigenschaften für einen informellen Vergleich sämtlicher in dieser Arbeit erwähnten Ansätze heran. In Abschnitt 6.2 untersuchen wir hingegen die Expressivität der in den Kapiteln 3, 4 und 5 vorgestellten Kalküle auf Basis des π-Kalküls.

6.1 Vergleich relevanter Eigenschaften

In folgender Tabelle werden die acht in dieser Arbeit vorgestellten Kalküle bezüglich der Eigenschaften aus Abschnitt 2.2.1 bewertet. Ein ✔ bedeutet, dass ein Kalkül die jeweilige Eigenschaft besitzt, während mit (✔) eine eingeschränkte Gültigkeit der Eigenschaft angedeutet wird. Falls ein Kalkül eine Eigenschaft gänzlich vermissen lässt, wird dies durch ✘ gekennzeichnet. Nachfolgend werden einige Erklärungen zu den Einträgen der Tabelle gegeben, die der Reihe nach zu jeder Eigenschaft erfolgen.

	formal	verständlich	intuitiv benutzbar	skalierbar	erweiterbar	anpassungsfähig	stochastisch	quantitativ	validierbar	dynamisch	nebenläufig
π-Kalkül	✔	✘	✘	(✔)	(✔)	✔	✘	✘	✔	✔	✔
Sπ	✔	✘	✘	✔	(✔)	✔	(✔)	✔	✔	✔	✔
bSπ	✔	✘	✘	✔	(✔)	✔	✔	✔	✔	✔	✔
β-Binder	✔	✔	(✔)	(✔)	✔	✔	✘	✘	(✔)	✔	✔
Sβ	✔	✔	(✔)	✔	✔	✔	✔	✔	(✔)	✔	✔
κ-Kalkül	✔	✔	✔	✘	✔	✘	✘	✘	(✔)	✔	✘
Brane-Kalkül	✔	✔	(✔)	✘	✔	✘	(✘)	(✘)	(✔)	✔	✔
Bio-Ambients	✔	✔	(✔)	(✔)	✔	✔	(✘)	(✘)	(✔)	✔	✔

- **formal:** Diese Arbeit befasst sich mit Prozesskalkülen, die naturgemäß formale Beschreibungsmittel darstellen.

- **verständlich:** Wie übersichtlich und lesbar eine Implementierung in einem Kalkül ist, hängt stark von der Möglichkeit ab, die betrachteten Entitäten visuell gegeneinander abzugrenzen. Eine Implementierung im klassischen π-Kalkül oder einer seiner stochastischen Varianten wirkt auf den ersten Blick abschreckend kompliziert und bedarf stets einer zusätzlichen Erklärung zu der Bedeutung der einzelnen Teilprozesse, um sich einem Leser zu erschließen. Für die Visualisierung von biologischen Vorgängen hat sich eine Umrahmung von Prozesstermen, die in den jeweiligen Kalkülen unterschiedlich benannt wird (Membran, Ambient, Binder), als sehr vorteilhaft erwiesen. Dabei ist die Möglichkeit zur Verschachtelung dieser Rahmen im Brane-Kalkül bzw. den Bio-Ambients der einfachen Darstellung bei β-Bindern überlegen, auch wenn Verschachtelung von Kompartimenten im β-Binder Kalkül, wie in Abschnitt 5.5.3 beschrieben, nachgebildet werden kann.

- **intuitiv benutzbar:** Die Schwierigkeiten bei der Implementierung mittels Prozesskalkülen erweisen sich als Nachteil dieser Ansätze. Die wenigen sehr allgemeinen Operationen ermöglichen zwar eine einfache Modellierung einiger Basisfälle, aber schon bei leichten Abweichungen von der Regel werden die Prozessterme sehr groß und die Implementierung umständlich. Der κ-Kalkül stellt in dieser Hinsicht eine Ausnahme dar. Proteininteraktionen lassen sich hier sehr intuitiv über die regelbasierte Syntax implementieren. Auch für eine intuitive Modellierung erweisen sich grafische Notationsmöglichkeiten als vorteilhaft.

- **skalierbar:** In Modellierungen auf Basis von Prozesstermen wird in der Regel von der exakten Ausdehnung und Position einer Entität abstrahiert. Über die Möglichkeit, die Gültigkeit von Kommunikationskanälen zu begrenzen, lassen sich jedoch leicht beliebige Beziehungen zwischen Entitäten umsetzen, die sich aus deren relativer Position oder Größe ergeben. Um auch bezüglich zeitlicher Aspekte von einer Skalierbarkeit sprechen zu können, ist eine quantitative Semantik erforderlich, wie sie bei Sπ, bSπ und Sβ vorliegt. Bei Brane- und κ-Kalkül liegt aufgrund der sehr speziellen Operationen keine gute Skalierbarkeit vor.

- **erweiterbar:** Um ein einmal erstelltes Modell zu einem späteren Zeitpunkt überarbeiten und erweitern zu können, sollte es einen modularen und verständlichen Aufbau besitzen. Die Erweiterbarkeit geht daher einerseits mit der Verständlichkeit des Ansatzes und andererseits mit der Möglichkeit zur Gliederung in Kompartimente einher. Bei sorgfältiger Implementierung lässt sich jedoch auch mit dem klassischen π-Kalkül und seinen stochastischen Varianten eine gewisse Erweiterbarkeit realisieren. An dieser Stelle wollen wir außerdem darauf hinweisen, dass auch Prozesskalküle selbst auf einfache Weise erweitert werden können. Einzelne Komponenten der Syntax lassen sich problemlos hinzufügen oder ersetzen, wie am Beispiel der polyadischen Varianten der π-Kalkül basierten Ansätze gezeigt. Auch Erweiterungen dieser Art können von großer Bedeutung bei der Wahl eines geeigneten Modellierungsansatzes sein.

- **anpassungsfähig:** Hier erweist sich das sehr einfache und allgemeine Interaktionsprinzip des π-Kalküls und ähnlicher Kalküle als großer Vorteil. Prinzipiell lassen sich beliebige Membranoperationen, Protein-Interaktionen oder auch Genexpression über den synchronen Austausch von Information zwischen unabhängigen Agenten (parallelen Teilprozessen) implementieren. Beim klassischen π-Kalkül geht eine Umsetzung sehr komplizierter Operationen wie z.B. der Endocytose mit einem Verlust an Verständlichkeit einher.

- **stochastisch:** Diese Eigenschaft ist für Simulationszwecke von äußerster Wichtigkeit. Bei Sπ ist in der Tabelle eine eingeschränkte Gültigkeit der Eigenschaft angegeben, da hier die Umsetzung von Auftrittswahrscheinlichkeiten nicht der Vorstellung eines Molekular- oder Zellbiologen entspricht. Brane-Kalkül und Bio-Ambients werden in dieser Arbeit ohne stochastische Semantik vorgestellt. Es sei jedoch angemerkt, dass durchaus stochastische Varianten dieser Kalküle existieren oder geeignete Erweiterungen in Diskussion sind.

- **quantitativ:** Die in dieser Arbeit vorgestellten stochastischen Kalküle ermöglichen über den Verteilungsparameter stets eine Modellierung quantitativer Aspekte. Es existieren allerdings auch Ansätze wie beispielsweise probabilistische Prozesskalküle, die Aktionen explizit eine Auftrittswahrscheinlichkeit zuordnen und daher trotz stochastischer Semantik nicht quantitativ sind.

- **validierbar:** Zum π-Kalkül existert reichhaltige Literatur, die sich mit der Verifiaktion von Prozesstermen bzgl. einer gegebenen Spezifikation befasst. Die Anwendung der Ergebnisse auf Erweiterungen des π-Kalküls muss allerdings noch genauer untersucht werden. Generell sind Prozesskalküle aufgrund ihrer simplen Syntax für Verifikation und Model Checking hervorragend geeignet.

- **dynamisch:** Die Beschreibbarkeit der dynamischen Evolution eines Systems ist eine von Prozesskalkülen naturgemäß unterstützte Eigenschaft.

- **nebenläufig:** Die Autonomie der Komponenten eines Modells auf Basis von Prozesstermen lässt stets ein hohes Maß an Parallelität zu. Der κ-Kalkül stellt mit seiner regelbasierten Semantik eine Ausnahme dar.

Die Beispielmodellierungen in dieser Arbeit zeigen auf, dass für eine Implementierung zellbiologischer Vorgänge ein stochastischer und quantitativer Ansatz unerlässlich ist. Der Nichtdeterminismus des klassischen π-Kalküls reicht zur Beschreibung der Dynamik biologischer Systeme nicht aus. Weiterhin haben wir festgestellt, dass aufgrund der enormen Modellgröße, die sich schon für sehr einfache Systeme ergibt, der Verständlichkeit und intuitiven Handhabbarkeit eines Ansatzes große Bedeutung zukommt. Als Fazit dieser Betrachtung lässt sich der Kalkül der stochastischen β-Binder als der am besten geeignete Ansatz für sehr allgemeine und komplexe Modellierungen in der Zellbiologie herausstellen. Mit entsprechender stochastischer Erweiterung sind die Bio-Ambients als gleichwertig anzusehen. Spezialkalküle wie der Brane-Kalkül müssen um eine ganze Reihe zusätzlicher Konzepte erweitert werden, um sämtliche Operationen der drei abstrakten Maschinen (siehe Abschnitt 2.1) nachbilden zu können. Für Modelle von vielzelligen Systemen (beispielsweise Komponenten des Immunsystems), bei denen die exakte Beschreibung von Membranen und ihren Interaktionen im Zentrum der Aufmerksamkeit steht, ist ein Ansatz über erweiterte Brane-Kalküle sicherlich nicht uninteressant.

6.2 Betrachtungen zur Expressivität

In der theoretischen Informatik ist seit jeher die Frage, welches Verhalten (d.h. welche Berechnungen) mit einem Kalkül oder sprachtheoretischen Ansatz ausgedrückt werden kann, von großer Bedeutung. Doch auch in der Systembiologie gewinnt diese Fragestellung an Bedeutung. Einerseits sind Biologen noch weit davon entfernt, die komplette Vielfalt der Lebensprozesse zu verstehen und andererseits ist das volle Ausmaß der Interaktionsmöglichkeiten zwischen verschiedenen biologischen Entitäten sowie deren Integration in höhere Organisationsstrukturen nicht ganzheitlich verstanden. Ein Modellierungsansatz, der heute bekannte Phänomene effektiv und elegant beschreibt, kann sich in der Zukunft aufgrund neuer Erkenntnisse über den tatsächlichen Komplexitätsgrad dieser Phänomene als nutzlos oder unelegant erweisen. Ein objektives Maß für die Ausdrucksstärke eines theoretischen Ansatzes bezüglich biologischer Modellierungen lässt sich selbstverständlich nicht angeben, da sich die Ergebnisse empirischer Studien von biologischen Phänomenen auf die unterschiedlichste Weise interpretieren lassen und dementsprechend zu verschiedenen Modellierungen führen können. Dennoch ist es möglich, die Ansätze untereinander zu vergleichen, indem man Übersetzungen zwischen verschiedenen Kalkülen oder Sprachen konstruiert.

Wie bereits in Kapitel 3 erwähnt, ist die Transitionssemantik des π-Kalküls mächtiger als die Reduktionssemantik, da sie sämtliche Reduktionen allein durch τ-Transitionen modellieren kann. Umgekehrt ist es jedoch nicht möglich, jegliches durch Transitionen ausdrückbare Verhalten mit der Reduktionssemantik nachzubilden. Der stochastische π-Kalkül ist wiederum ausdrucksstärker als der klassische π-Kalkül mit Transitionssemantik, da das Formulieren von quantitativen Aspekten eines biologischen Systems ohne die Einbeziehung einer mathematischen Verteilung unmöglich ist. Weder die Inkorporation des Zeitbegriffs, noch die Umsetzung von Wahrscheinlichkeiten für bestimmte Verhaltensweisen sind unter Verzicht auf die stochastische Erweiterung darstellbar. Der biochemische stochastische π-Kalkül stellt zwar im Vergleich zu Sπ eine besser an biologische Gegebenheiten angepasste Beschreibungssprache dar, geht aber mit einem gewissen Verlust an Ausdrucksstärke einher. Dies ist sowohl durch die eingeschränkte Syntax bedingt, wie auch durch das Verwenden einer Reduktionssemantik und der Assoziation der Raten mit den Kommunikationskanälen anstatt den einzelnen Präfixen.

Ein Vergleich des Kalküls der β-Binder mit dem π-Kalkül erweist sich sowohl für die klassischen als auch die stochastischen Versionen beider Ansätze als deutlich schwieriger als Vergleiche der π-Kalkül-Varianten untereinander. Wir wollen daher versuchen, Übersetzungen zwischen diesen Ansätzen anzugeben. Abschnitt 6.2.1 beschäftigt sich zunächst mit der Übersetzung von Sβ in bSπ. In Abschnitt 6.2.2 widmen wir uns einigen Überlegungen zu den problematischen join- und split-Operationen in Sβ. Schließlich skizzieren wir in Abschnitt 6.2.3 die Übersetzung von bSπ in Sβ und erläutern dabei auftretende Probleme.

6.2.1 Übersetzung von Sβ in bSπ

Eine Übersetzung in diese Richtung hat sowohl theoretische wie auch praktische Relevanz. Ließe sich ein beliebiges β-Binder-Modell in ein äquivalentes π-Kalkül-Modell transformieren, so könnte das umfangreiche Theoriewissen und die Erfahrungen mit

der Analyse und Verifikation von π-Kalkül-Modellen auch für den β-Binder-Kalkül zur Anwendung kommen. Es wäre damit außerdem nachgewiesen, dass sich durch die Erweiterung kein Zuwachs an Ausdrucksstärke ergibt. Ein praxisbezogener Vorteil wäre die Möglichkeit, Simulatoren für π-Prozesse auch für Übersetzungen von Bio-Prozessen einsetzen zu können. Leider erweist sich eine Übersetzung der join- und split-Operationen des β-Binder-Kalküls als sehr schwierig. Der Grund hierfür liegt in deren unbeschränkter Definierbarkeit in Abhängigkeit von Bindern sowie Prozesstermen. Auch die beiden Substitutionen, die bei jeder der Operationen zum Einsatz kommen, erschweren die Übersetzung. Wir werden uns in diesem Abschnitt daher auf eine Übersetzung von Modellen ohne join und split beschränken. Da sich eine Übersetzung vom klassischen β-Binder-Kalkül in den klassischen π-Kalkül mit Reduktionssemantik auf einfache Weise aus der Übersetzung der stochastischen Varianten ableiten lässt, behandeln wir nur den stochastischen Fall. Darüber hinaus verwenden wir beide Kalküle in ihrer polyadischen Variante.

Zunächst benötigen wir einen Äquivalenzbegriff für Prozesse, um das Verhalten eines Prozesses mit seiner Übersetzung vergleichen zu können. In der Theorie zum π-Kalkül kommen dabei je nach Variante verschiedenartige *Bisimulationsrelationen* zum Einsatz, die zwei Teilprozesse in Beziehung setzen, falls sie in jedem Kontext (d.h. umgebenden Prozess) nur zu Folgeprozessen evolvieren können, die wieder über die Relation in Beziehung stehen. Mit anderen Worten weisen zwei Prozesse das gleiche Verhalten auf, wenn jeder Ableitungsschritt in dem einen durch einen vergleichbaren Ableitungsschritt in dem anderen simuliert werden kann. Selbst für den klassischen π-Kalkül ist die Theorie zu diesen Relationen bereits so umfassend, dass wir davon absehen, eine geeignete Relation zwischen Sβ und bSπ einzuführen. Wir orientieren uns stattdessen intuitiv am Bisimulationsbegriff und geben unsere Übersetzung ohne formalen Korrektheitsbeweis an.

Ein wichtiger Begriff im Zusammenhang mit der Expressivität von Prozesskalkülen ist die *Beobachtbarkeit* von Verhalten. Damit zwei Prozesse als äquivalent betrachtet werden können, dürfen sie für einen externen Beobachter nicht unterscheidbar sein. Dabei nimmt der Beobachter lediglich die ausgeführten Aktionen der jeweiligen Ableitungsschritte wahr. Zwei äquivalente Prozesse müssen in der gleichen Reihenfolge vergleichbare beobachtbare Aktionen ausführen und im stochastischen Fall auch mit der jeweils gleichen Rate. Allerdings besitzt Sβ neben den Eingabe- und Ausgabe-Präfixen noch drei weitere Präfixe, um die Binder eines Kompartiments zu manipulieren. Wir führen daher in der Übersetzung Kanalnamen *exp*, hid_x und unh_x ein, die im Sβ-Modell nicht vorkommen dürfen und deren Verwendung in Eingabe- und Ausgabe-Präfixen im bSπ-Modell als äquivalentes Verhalten zu einem exp-, hid- bzw. unh-Präfix betrachtet werden soll. Wir definieren darüber hinaus weitere Namen, deren Verwendung in Kommunikationspräfixen als nicht beobachtbar gilt, da wir manche Aktionen eines Sβ-Modells durch mehrere Aktionen im bSπ-Modell übersetzen, von denen nur eine beobachtbar sein soll. Wir fassen die Namen für die Verwendung in nicht beobachtbaren Aktionen in der Menge $\mathcal{I}$ zusammen. Sämtliche Namen in $\mathcal{I}$ kommen im Sβ-Modell nicht vor und besitzen die Rate ∞, damit sie keine zusätzliche Zeit verbrauchen und keinen Einfluss auf das quantitative Verhalten des übersetzten Prozesses ausüben.

Neben dem Verzicht auf join- und split-Operationen vereinbaren wir der Einfachheit halber einige weitere Einschränkungen für Sβ-Modelle.

1. Die Subjekte von Bindern dürfen nicht kommuniziert werden und kommen nicht als Kanäle für Intraaktionen innerhalb eines Kompartiments vor.
2. Die Subjekte von elementaren Bindern aller Kompartimente sind paarweise verschieden.
3. Die Basisraten von Interaktionen sind nur von den Kanalnamen abhängig, d.h. die Affinitätsfunktion α weist zwei nicht disjunkten Bindertypen die Rate der Bindersubjekte zu, wobei angenommen wird, dass die Subjekte bei nicht disjunkten Bindern die gleiche Rate besitzen.
4. Die Namen in Bindertypen werden nicht als Kommunikationskanäle verwendet.
5. In einem Kompartiment frei vorkommende Namen kommen in anderen Kompartimenten nicht vor.

Die Einschränkungen 1. und 2. gehen sicherlich nicht mit einem Verlust an Ausdrucksstärke einher. Für 5. gilt dies unter der Prämisse des Verzichts auf join und split ebenfalls. 3. und 4. sind vermutlich mit einem gewissen Verlust an Expressivität verbunden, den wir hier in Kauf nehmen.

Definition 6.2.1 (Übersetzung Sβ zu bSπ): Die Übersetzungsfunktionen von Bio-Prozessen, Bindern bzw. π^β-Prozessen in bSπ Prozesse werden durch $[\![\,]\!]$ zusammengefasst und sind wie folgt definiert.

$$[\![B_1\langle P_1\rangle \,||\, B_2\langle P_2\rangle \,||\, \ldots \,||\, B_n\langle P_n\rangle\,]\!] \quad =$$
$$[\![P_1]\!]_{B_1} \mid [\![P_2]\!]_{B_2} \mid \ldots \mid [\![P_n]\!]_{B_n} \mid [\![B_1]\!] \mid [\![B_2]\!] \mid \ldots \mid [\![B_n]\!] \mid$$
$$!e()[\infty].(\nu\, z)(\nu\, hid_z)(\nu\, unh_z)(\nu\, com_z)(\nu\, cancel_z)(\nu\, sc_x)(\nu\, r_z)$$
$$\overline{exp}(z, hid_z, unh_z, com_z, cancel_z, sc_x)[\infty].(Bin_z \mid !r_z()[\infty].Bin_z)$$

$$[\![\beta(x:\{x_1,\ldots,x_m\})B^*]\!] \quad = \quad \overline{r_x}()[\infty].\mathbf{0} \mid !r_x()[\infty].Bin_x \mid [\![B^*]\!]$$

$$[\![\beta^h(x:\{x_1,\ldots,x_m\})B^*]\!] \quad =$$
$$sc_x(s')[\infty].\mathbf{0} \mid unh_x()[r].\overline{r_x}()[\infty].\mathbf{0} \mid !r_x()[\infty].Bin_x \mid [\![B^*]\!]$$

$$[\![\,]\!] \quad = \quad \mathbf{0}$$

$$[\![\mathbf{0}]\!]_B \quad = \quad \mathbf{0}$$

$$[\![\hat{x}(y_1,\ldots,y_k)[r].P]\!]_B \quad =$$
$$\begin{cases} \tau[\infty].(A_{\hat{x}_1} + \ldots + A_{\hat{x}_n}) & \text{falls } B \equiv \beta(x:\{x_1,\ldots,x_n\})B^* \\ \hat{x}(y_1,\ldots,y_k)[r].[\![P]\!]_B & \text{sonst} \end{cases}$$
$$\text{wobei } A_{\hat{x}_i} = (\nu\, c)(\overline{c}()[\infty].\mathbf{0} \mid !c()[\infty].\overline{com_x}()[\infty].(\hat{x}_i(y_1,\ldots,y_k)[r].[\![P]\!]_B + cancel_x()[\infty].\overline{c}()[\infty].\mathbf{0}))$$
$$\text{mit } \hat{x}_i \in \{\overline{x_i}, x_i\} \text{ für } i = 1,\ldots,n \text{ und } \hat{x} \in \{\overline{x}, x\}$$

$$[\![\exp(x,\Gamma)[r].P]\!]_B \quad =$$
$$\tau[r].\overline{e}()[\infty].exp(z, hid_z, unh_z, com_z, cancel_z, sc_z)[\infty].[\![P\{z/x\}]\!]_{B\beta(z:\Gamma)}$$

$$[\![\mathrm{hid}(x)[r].P]\!]_B \quad =$$
$$\overline{hid_x}()[r].([\![P]\!]_B \mid \overline{sc_x}(s)[\infty].\mathbf{0} \mid !sc_x(s')[\infty].[s'=s]\overline{cancel_x}()[\infty].\overline{sc_x}(s)[\infty].\mathbf{0})$$

$$
\begin{aligned}
[\![\mathrm{unh}(x)[r].P]\!]_B &= \overline{unh_x}()[r].([\![P]\!]_B \mid \overline{sc}_x(f)[\infty].\mathbf{0}) \\
[\![(\nu\, x)P]\!]_B &= (\nu\, x)[\![P]\!]_B \\
[\![!P]\!]_B &= ![\![P]\!]_B \\
[\![P \mid Q]\!]_B &= [\![P]\!]_B \mid [\![Q]\!]_B
\end{aligned}
$$

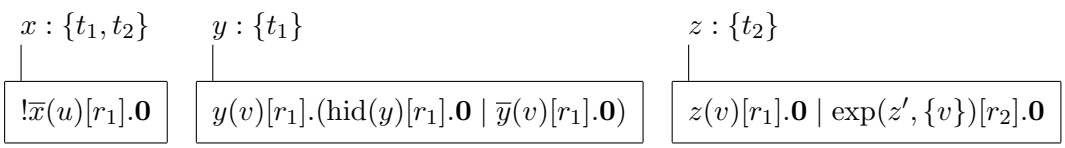
Es gilt $n, m, k \in \mathbb{N}$ und $\mathcal{I} = \{c, e, exp\} \cup \{com_x, cancel_x, sc_x, r_x \mid \forall x \in \mathcal{N}\}$. Die Übersetzungsfunktion für π^β-Prozesse weist als zusätzliches Argument einen Binder auf, der bei der Übersetzung von Kommunikationspräfixen analysiert wird. Die Makros Bin_x sind für alle Namen x definiert als

$$Bin_x = hid_x()[r].unh_x()[r].\overline{r_x}()[\infty].\mathbf{0} + com_x()[\infty].\overline{r_x}()[\infty].\mathbf{0}\ ,$$

wobei r die Basisrate des Kanals x ist. □

Die Übersetzung soll anhand eines einfachen Beispiels veranschaulicht werden. Zuvor machen wir jedoch einige allgemeine Anmerkungen. Ein Binder der Form $\beta(x : \Gamma)$ wird im bSπ-Modell durch einen parallelen Prozess Bin_x repräsentiert, der dessen Zustand modelliert und an den von außen Anfragen gestellt werden können. Im verfügbaren Zustand kann er über eine Interaktion mit Kanal hid_x versteckt werden, oder über com_x bestätigen, dass er verfügbar ist, so dass zwei Prozesse über den Kanal x kommunizieren können. Im versteckten Zustand kann der Binder lediglich über unh_x verfügbar gemacht werden. Der Binder-Prozess wird ständig repliziert, wobei das Präfix $\overline{r_x}()[\infty]$ sicherstellt, dass nicht zwei Binder-Prozesse mit gleichem Subjekt zur gleichen Zeit aktiv sind. Ein Bio-Prozess wird entsprechend seiner Kompartimente $B_i\langle P_i\rangle$ $(i = 1, ..., n)$ in eine Menge paralleler Teilprozesse $[\![P_i]\!]_{B_i}$, sowie eine Menge paralleler Binder-Prozesse $[\![B_i]\!]$ übersetzt, die je nachdem, ob ein elementarer Binder verfügbar oder versteckt ist, unterschiedlich übersetzt werden. Der Fall $[\![\,]\!] = \mathbf{0}$ beendet die Übersetzung eines zusammengesetzten Binders. (Man bedenke, dass B^* auch eine leere Sequenz von elementaren Bindern darstellen kann.) Als letzte parallele Komponente enthält die Übersetzung eines Bio-Prozesses einen Teilprozess, der das Entstehen neuer Binder modelliert. Eine Anfrage über Kanal e startet die Replikation eines neuen Binder-Prozesses, dessen neu erstellte Kanäle z, hid_z, unh_z, com_z, $cancel_z$ und sc_z an den anfordernden Prozess zurückgesandt werden. Bei der Übersetzung der eingebetteten π^β-Prozesse ist vor allem der Fall einer Interaktion zwischen Kompartimenten interessant. Im übersetzten Prozess prüft jedes Kompartiment über einen Kanal com_x, ob sein jeweiliger Binder im verfügbaren Zustand ist. Gegebenenfalls muss eine Interaktion über den Kanal $cancel_x$ abgebrochen werden. Wir werden in Beispiel 6.2.3 genauer darauf eingehen.

Beispiel 6.2.2: Der folgende Bio-Prozess soll in einen äquivalenten bSπ-Prozess übersetzt werden.



Man beachte, dass das erste Kompartiment mit den beiden anderen kommunizieren kann, während diese untereinander zu keiner Kommunikation fähig sind, obwohl

sie geeignete Präfixe besitzen. Wir werden diese Gegebenheit in der Übersetzung dadurch umsetzen, dass wir die Namen in den Bindertypen als Kanäle verwenden. Im Folgenden kürzen wir die Binder der drei Kompartimente der Reihe nach durch B_1, B_2 und B_3 ab und ensprechend die eingebetteten π^β-Prozesse durch P_1, P_2 und P_3. Nach Anwendung der Übersetzungsfunktion für Bio-Prozesse ergibt sich zunächst

$$[\![P_1]\!]_{B_1} \mid [\![P_2]\!]_{B_2} \mid [\![P_3]\!]_{B_3} \mid [\![B_1]\!] \mid [\![B_2]\!] \mid [\![B_3]\!] \mid E \ ,$$

wobei das Makro E für die parallele Komponente zur Erzeugung neuer Binder steht, die nicht vom jeweiligen Modell abhängt. Die Übersetzungen der Binder ergeben jeweils einen parallelen Teilprozess

$$\overline{r_a}()[\infty].\mathbf{0} \mid !r_a()[\infty].Bin_a \mid \mathbf{0}$$

mit $a \in \{x, y, z\}$. Der interessante Teil der Übersetzung ist jedoch die Behandlung der π^β-Prozesse. P_1 wird zu

$$\begin{aligned}
&!\tau[\infty].(A_{\overline{t_1}} + A_{\overline{t_2}}) \quad = \\
&!\tau[\infty].((\nu\, c)(\overline{c}()[\infty].\mathbf{0} \mid !c()[\infty].\overline{com_x}()[\infty].(\overline{t_1}(u)[r_1].\mathbf{0} + cancel_x()[\infty].\overline{c}()[\infty].\mathbf{0}))+ \\
&(\nu\, c)(\overline{c}()[\infty].\mathbf{0} \mid !c()[\infty].\overline{com_x}()[\infty].(\overline{t_2}(u)[r_1].\mathbf{0} + cancel_x()[\infty].\overline{c}()[\infty].\mathbf{0})))
\end{aligned}$$

übersetzt. Die Erzeugung des τ-Präfixes bei der Übersetzung von $\overline{x}(u)[r_1]$ ist notwendig, um sicherzustellen, dass der äußerste Operator unter der Replikation ein Präfix ist. Die beiden alternativen Komponenten des Prozesses ermöglichen entweder eine Kommunikation über t_1 oder über t_2. Dies entspricht der Auswahl des zweiten bzw. dritten Kompartiments als Kommunikationspartner im betrachteten Bio-Prozess. Die Übersetzung von P_2 ergibt

$$\begin{aligned}
&\tau[\infty].(\nu\, c)(\overline{c}()[\infty].\mathbf{0} \mid !c()[\infty].\overline{com_y}()[\infty].(\overline{t_1}(v)[r_1].(\overline{hid_y}()[r_1].(\mathbf{0} \mid \overline{sc_y}(s)[\infty].\mathbf{0} \mid \\
&!sc_y(s')[\infty].[s' = s]\overline{cancel_y}()[\infty].\overline{sc_y}(s)[\infty].\mathbf{0}) \mid A_{\overline{t_1}}) + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0}).
\end{aligned}$$

$A_{\overline{t_1}}$ hat die gleiche Gestalt, die bereits in der Übersetzung von P_1 angegeben wurde, wobei x durch y und u durch v zu ersetzen ist. Schließlich lässt sich P_3 zu

$$\tau[\infty].A_{t_2} \mid \tau[r_2].\overline{e}()[\infty].exp(z, hid_z, unh_z, com_z, cancel_z, sc_z)[\infty].\mathbf{0}$$

übersetzen. Das τ-Präfix, das bei der Übersetzung des exp-Präfixes entsteht, dient dem Erhalt des stochastischen Verhaltens. Die beobachtbare Komponente der Aktion ist die Kommunikation über den Kanal *exp*, die mit Rate ∞ stattfindet. □

Als nächstes gehen wir anhand eines Beispiels genauer auf das etwas komplizierte Zusammenspiel der Kanäle $cancel_x$ und sc_x (stop canceling) in den Übersetzungen von Kommunikationspräfixen ein.

Beispiel 6.2.3: Wir betrachten die Übersetzung des Bio-Prozesses aus Beispiel 6.2.2 und interessieren uns insbesondere für das zweite Kompartiment. Wir übersetzen stets nur die für die Ableitung benötigten Teilprozesse, um die Darstellung zu verkürzen. Das $\tau[\infty]$-Präfix aus der Übersetzung von $y(v)[r_1]$ wurde bereits durch eine entsprechende Aktion verbraucht.

$[\![P_1]\!]_{B_1} \mid (\nu\, c)(\overline{c}()[\infty].\mathbf{0} \mid\, !c()[\infty].\overline{com_y}()[\infty].(t_1(v)[r].[\![(\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0})]\!]+ cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid [\![P_3]\!]_{B_3} \mid [\![B_1]\!] \mid [\![B_2]\!] \mid [\![B_3]\!] \mid E$

Im Folgenden kürzen wir $[\![P_1]\!]_{B_1} \mid [\![P_3]\!]_{B_3} \mid [\![B_1]\!] \mid [\![B_2]\!] \mid [\![B_3]\!] \mid E$ durch P ab. Zunächst wird über den privaten Kanal c eine kontrollierte Replikation des Kommunikationsprozesses eingeleitet. Anschließend wird über Kanal com_y die Verfügbarkeit des Binders bestätigt. (Der Binder-Prozess wird entsprechend reduziert.)

$$\begin{aligned}
&\xrightarrow{c,\infty,1,1} && (\nu\, c)(\overline{com_y}()[\infty].(t_1(v)[r].[\![(\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0})]\!]+ \\
& && cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0}) \mid\, !c()[\infty].\overline{com_y}()[\infty].(t_1(v)[r]. \\
& && [\![(\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0})]\!] + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid P \\
&\xrightarrow{com_y,\infty,1,1} && (\nu\, c)((t_1(v)[r].[\![(\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0})]\!] + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0}) \mid \\
& && !c()[\infty].\overline{com_y}()[\infty].(t_1(v)[r].[\![(\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0})]\!]+ \\
& && cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid P'
\end{aligned}$$

Da die Übersetzung des ersten Kompartiments nach einer ähnlichen Ableitung ein ungeschütztes $\overline{t_1}(u)[r_1]$-Präfix enthält, kann nun eine Kommunikation zwischen den Kompartimenten erfolgen.

$$\begin{aligned}
&\xrightarrow{t_1,r,2,1} && (\nu\, c)([\![(\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0})]\!] \mid\, !c()[\infty].\overline{com_y}()[\infty].(t_1(v)[r].\mathbf{0}+ \\
& && cancel_y()[\infty].\overline{c}()[\infty].[\![(\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0})]\!])) \mid P''
\end{aligned}$$

Der Teilprozess unter dem Replikationsoperator kann nun nicht weiter reduziert werden, da unter der Restriktion kein freies $\overline{c}()[\infty]$-Präfix mehr existiert. Wir entfernen den Teilprozess und den bedeutungslos gewordenen Restriktionsoperator daher aus der Darstellung. Des Weiteren übersetzen wir den Teilprozess $\mathrm{hid}(y)[r_1].\mathbf{0} \mid \overline{y}(v)[r_1].\mathbf{0}$, um die Ableitung fortzusetzen. Das $\tau[\infty]$-Präfix aus der Übersetzung von $\overline{y}(v)[r_1]$ verbrauchen wir, ohne einen Zwischenschritt anzugeben.

$$\begin{aligned}
= \quad & \overline{hid_y}()[r_1].(\mathbf{0} \mid \overline{sc_y}(s)[\infty].\mathbf{0} \mid\, !sc_y(s')[\infty].[s' = s]\overline{cancel_y}()[\infty].\overline{sc_y}(s)[\infty].\mathbf{0}) \mid \\
& (\nu\, c)(\overline{c}()[\infty].\mathbf{0} \mid\, !c()[\infty].\overline{com_y}()[\infty].(\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid P''
\end{aligned}$$

Wir haben nun bereits eine erfolgreiche Kommunikation über einen Binder-Kanal gezeigt und wollen jetzt den Fehlschlag einer Kommunikation betrachten. Erneut wird über die Kanäle c und com_y die Kommunikation vorbereitet.

$$\begin{aligned}
\longrightarrow^2 \quad & \overline{hid_y}()[r_1].(\mathbf{0} \mid \overline{sc_y}(s)[\infty].\mathbf{0} \mid\, !sc_y(s')[\infty].[s' = s]\overline{cancel_y}()[\infty].\overline{sc_y}(s)[\infty].\mathbf{0}) \mid \\
& (\nu\, c)((\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0}) \mid\, !c()[\infty].\overline{com_y}()[\infty]. \\
& (\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid P'''
\end{aligned}$$

Nun soll der Binder des zweiten Kompartiments versteckt werden. Über Kanal hid_y wird zunächst der Binder-Prozess darüber informiert. Anschließend müssen auch die übrigen parallelen Komponenten des Kompartiments informiert werden, falls sie bereits eine Kommunikation über den Binder-Kanal initiiert haben. Dafür findet über sc_y eine kontrollierte Replikation des *cancel*-Prozesses statt.

$$\xrightarrow{hid_y,r_1,1,1} \quad \overline{sc_y}(s)[\infty].\mathbf{0} \mid !sc_y(s')[\infty].[s'=s]\overline{cancel_y}()[\infty].\overline{sc_y}(s)[\infty].\mathbf{0} \mid (\nu\, c)((\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0}) \mid !c()[\infty].\overline{com_y}()[\infty].(\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid P^{(4)}$$

$$\xrightarrow{sc_y,\infty,1,1} \quad [s=s]\overline{cancel_y}()[\infty].\overline{sc_y}(s)[\infty].\mathbf{0} \mid !sc_y(s')[\infty].[s'=s]\overline{cancel_y}()[\infty].\overline{sc_y}(s)[\infty].\mathbf{0} \mid (\nu\, c)((\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0}) \mid !c()[\infty].\overline{com_y}()[\infty].(\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid P^{(4)}$$

Der Vergleichsoperator stellt die Übereinstimmung der Namen fest, so dass nun der Abbruch der Kommunikation über Kanal $cancel_y$ vollzogen werden kann.

$$\xrightarrow{cancel_y,\infty,1,1} \quad \overline{sc_y}(s)[\infty].\mathbf{0} \mid !sc_y(s')[\infty].[s'=s]\overline{cancel_y}()[\infty].\overline{sc_y}(s)[\infty].\mathbf{0} \mid (\nu\, c)(\overline{c}()[\infty].\mathbf{0} \mid !c()[\infty].\overline{com_y}()[\infty].(\overline{t_1}(v)[r_1].\mathbf{0} + cancel_y()[\infty].\overline{c}()[\infty].\mathbf{0})) \mid P^{(4)}$$

Die Kommunikation über den Binder wurde damit auf ihren Anfangszustand zurückgesetzt. Der *cancel*-Prozess bleibt weiter aktiv und könnte weitere Kommunikationsversuche über den Binder-Kanal zurücksetzen. Erst ein *unhide*-Prozess, der den Namen f über den Kanal sc_y versendet, würde den *cancel*-Vorgang stoppen, da der Vergleichsoperator im Fall $[f=s]$ blockiert. □

6.2.2 Übersetzung von join- und split-Operationen

Sicherlich lässt sich für ein Sβ-Modell mit einer konkreten Instanziierung der Funktionen f_{join} und f_{split} leicht eine Übersetzung in bSπ angeben. Man betrachte beispielsweise die Beispielmodellierungen der Genexpression mit positivem Feedback aus den Abschnitten 4.3 und 5.4. Das beobachtbare Verhalten der beiden Modelle ist äquivalent, wenn man sich darauf einigt, einige der Kanäle für nicht beobachtbare Kommunikationen zu reservieren, wie wir es im letzten Abschnitt bei der allgemeinen Übersetzung mit der Menge $\mathcal{I}$ getan haben.

Ob eine Übersetzung von join und split auch allgemein und nicht bloß an der konkreten Modellierung gelingen kann, wollen wir im Folgenden untersuchen. Wir skizzieren zunächst einen Ansatz zur Übersetzung einer sehr allgemeinen Klasse von join- und split-Operationen und diskutieren anschließend die dabei auftretenden Probleme. Sei $f_{\text{join}}(B_1, B_2, P_1, P_2) = (B, \sigma_1, \sigma_2, r)$ mit festen Prozessen P_1, P_2 und Substitutionen σ_1, σ_2 ein Fall der join-Funktion in einem gegebenen Modell (die Binder dürfen sowohl feste Komponenten wie auch Platzhalter enthalten). Wir versuchen die Prozesse P_1 und P_2 wie folgt mit einer Funktion $\hat{[\![}\,\hat{]\!]}$ zu übersetzen, wobei $[\![\,]\!]$ die Übersetzungsfunktion für π^β-Prozesse aus Definition 6.2.1 ist.

$$\hat{[\![}P_1\hat{]\!]}_{B_1} = [\![P_1]\!]_{B_1} + (join_i()[r].[\![P_1]\!]_B\,\sigma_1 \mid [\![B]\!])$$
$$\hat{[\![}P_2\hat{]\!]}_{B_2} = [\![P_2]\!]_{B_2} + (\overline{join_i}()[r].[\![P_2]\!]_B\,\sigma_2)$$

Für jeden Fall der join-Funktion wird hierbei ein neuer Name $join_i$ verwendet. Ein Fall der split-Funktion der Gestalt $f_{\text{split}}(B, P_1, P_2) = (B_1, B_2, \sigma_1, \sigma_2, r)$ für feste Prozesse und Substitutionen könnte durch

$$\hat{[\![}P_1\hat{]\!]}_B = [\![P_1]\!]_B + (\mathit{split}_i()[r].[\![P_1]\!]_{B_1}\,\sigma_1 \mid [\![B_1]\!])$$
$$\hat{[\![}P_2\hat{]\!]}_B = [\![P_2]\!]_B + (\overline{\mathit{split}_i}()[r].[\![P_2]\!]_{B_2}\,\sigma_2 \mid [\![B_2]\!])$$

übersetzt werden. Zunächst ergibt sich das Problem, dass Binder im Laufe einer Ableitung entstehen, versteckt und wieder sichtbar gemacht werden können. Um sicherzustellen, dass sich in dem Moment, in dem eine join- oder split-Operation über einen Kanal join_i bzw. split_i eingeleitet wird, alle benötigten Binder im korrekten Zustand befinden, könnten Testkanäle wie com_x in der Übersetzungsfunktion $[\![\,]\!]$ zum Einsatz kommen. Auch ein Test, ob zwei Prozesse sich im gleichen oder in verschiedenen Kompartimenten befinden, wäre auf ähnliche Weise möglich. Wir verzichten auf eine detaillierte Ausarbeitung und kommen nun zu einem größeren Problem.

Die Funktion $\hat{[\![}\,\hat{]\!]}$ übersetzt zwar sämtliche Vorkommen von Prozessen P_1 und P_2, die im initialen Prozess einer Ableitung vorkommen, sie erfasst aber nicht solche Vorkommen, die zu einem späteren Zeitpunkt entstehen. Beispielsweise könnte ein Prozess $Q \mid P_2$ innerhalb eines Kompartiments durch einen Ableitungsschritt $Q \longrightarrow P_1$ zu $P_1 \mid P_2$ evolvieren und anschließend wäre im Sβ-Modell eine split-Operation möglich, im übersetzten bSπ-Modell allerdings nicht. Hier wäre ein Test auf Gleichheit von Prozesstermen erfoderlich, der in bSπ sicherlich nicht möglich ist. Die Frage, ob trotzdem eine allgemeine Übersetzung von join und split möglich ist, bleibt an dieser Stelle offen.

6.2.3 Übersetzung von bSπ in Sβ

Auch eine Übersetzung in diese Richtung ist nicht uninteressant, da sie einen Nachweis darstellen würde, dass Sβ mindestens die gleiche Expressivität besitzt wie bSπ. Doch auch hier stoßen wir auf ein Problem. Die einzigen syntaktischen Elemente von bSπ, die in Sβ nicht vorkommen, sind alternative Teilprozesse und der Vergleichsoperator. Eine Übersetzung für alternative Teilprozesse ist leicht anzugeben:

$$\dot{[\![}P + Q\dot{]\!]} = (\nu\, a)(a()[\infty].P \mid a()[\infty].Q \mid \overline{a}()[\infty].\mathbf{0})$$

Über den Kanal a findet genau eine Kommunikation statt, die einen der Teilprozesse P oder Q aktiviert und den anderen für die restliche Ableitung blockiert. Die Kommunikation über den Auswahlkanal a sei hierbei nicht beobachtbar.

Der Vergleichsoperator stellt allerdings ein sehr mächtiges Konstrukt dar, welches im Allgemeinen schwer zu übersetzen ist. Wir betrachten daher eine Einschränkung, die eine einfache Übersetzung ermöglicht. Falls die beiden Namen, die verglichen werden sollen, nicht als Kommunikationskanäle Verwendung finden, gilt die folgende Übersetzung:

$$\dot{[\![}[x = y]P\dot{]\!]} = \overline{x}()[\infty].\mathbf{0} \mid y()[\infty].P$$

Nur für den Fall, dass die Namen identisch sind, können die beiden parallelen Komponenten der Übersetzung kommunizieren, wobei P aktiviert wird. Eine weitere Voraussetzung zur Wahrung des quantitativen Verhaltens besteht darin, dass die Kommunikation über x bzw. y nicht beobachtbar ist und mit Rate ∞ stattfindet.

Bei der Übersetzung eines bSπ-Prozesses in einen Bio-Prozess reicht es aus, sämtliche alternativen Teilprozesse und Auswahloperatoren gemäß der Funktion $\dot{[\![}\,\dot{]\!]}$ zu übersetzen und den resultierenden π^β-Prozess mit einem beliebigen Binder in ein Kompartiment zu integrieren, um einen syntaktisch korrekten Bio-Prozess zu erhalten.

Abschließend kann man sagen, dass Vergleiche der Expressivität von Prozesskalkülen stets mit erheblichen Schwierigkeiten verbunden sind. Die Frage, welcher Kalkül in der molekular- und zellbiologischen Praxis am vielversprechendsten ist, wird wohl nicht nur durch theoretische Betrachtungen, sondern hauptsächlich durch die Erfahrungen mit konkreten Implementierungen entschieden werden müssen.

Kapitel 7

Bewertung und Ausblick

Mit wachsendem Verständnis zellbiologischer Prozesse wächst auch die Komplexität und die Detailliertheit der von Biologen aufgestellten Modelle und gleichzeitig die damit verbundenen Anforderungen an die formalen Ansätze für biologische Modellierungen. Insbesondere spielt die Simulation von biologischen Vorgängen zur Gewinnung von Vorhersagen über die Entwicklung eines Systems unter verschiedenen Bedingungen eine immer größere Rolle. Einerseits lassen sich durch Computersimulation reale Experimente einsparen bzw. zumindest besser planen, zum anderen erhofft man sich Aussagen über Systeme mit sehr vielen Molekülen zu gewinnen, die einer experimentellen Betrachtung nur schwer zugänglich sind. Wir haben in dieser Arbeit aus Sicht der theoretischen Informatik einige aktuell diskutierte Ansätze zur biologischen Modellierung auf Prozesskalkülbasis vorgestellt, an einfachen Beispielmodellierungen veranschaulicht und bezüglich relevanter Eigenschaften verglichen. Darüber hinaus wurde versucht, den Kalkül der stochastischen β-Binder, einen vielversprechenden Vertreter aus der Gruppe der Prozesskalküle, in seiner Ausdrucksstärke mit dem π-Kalkül, der einen der wohluntersuchtesten Ansätze darstellt, zu vergleichen.

Einige der diskutierten Modelleigenschaften können aufgrund der in den Beispielen dieser Arbeit gemachten Erfahrungen als essentiell eingestuft werden. Dazu gehören neben der Verständlichkeit und intuitiven Handhabbarkeit eines Ansatzes die Möglichkeiten zur stochastischen und quantitativen Erfassung des Sachbestandes. Im Zusammenhang mit einer einheitlichen Beschreibung der Prozesse in lebenden Zellen hat sich die Anpassungsfähigkeit eines Ansatzes an verschiedenartige Operationen als sehr bedeutend herausgestellt. Obwohl die Wahl eines geeigneten Modellierungsansatzes für ein konkretes biologisches System durch die Resultate dieser Arbeit erleichtert wird, lässt sich nicht bedingungslos ein Ansatz als die optimale Lösung benennen. Der Reichtum an Varianten und Erweiterungen, die zu den einzelnen Kalkülen existieren, erschwert die Auswahl genauso wie eine unterschiedliche und schwer vergleichbare Vorstellung des Berechnungsbegriffs. (Während π-Kalkül-Ansätze auf dem Prinzip der Kommunikation über gemeinsame Kanäle basieren, werden Berechnungen im Brane-Kalkül direkt durch Membranen mittels Operationen realisiert, die auf physiaklischen Vorgängen beruhen.) Viele der Ansätze werden ständig an neue Erkenntnisse aus der Laborforschung angepasst und verbessern so ihre Eignung für biologische Modellierungen. Ob sich einer der Ansätze durchsetzt, oder auch weiterhin eine Vielzahl verschiedener Kalküle zur Anwendung kommt, werden schließlich die Erfahrungen mit den stets größer werdenden Implementierungen und Simulationsmodellen entscheiden.

Nach derzeitigem Stand sind die Ansätze der β-Binder und Bio-Ambients besonders hervorzuheben, da beide die essentiellen Anforderungen zellbiologischer Modellierung gut erfüllen und bereits Simulatoren auf Basis dieser Ansätze existieren, die biologisch glaubwürdige Ergebnisse für einfache und wohluntersuchte Systeme liefern.

Neben der Weiterentwicklung der Syntax und Semantik der Kalküle ist für zukünftige Arbeiten auf dem Gebiet auch eine Untersuchung der Anwendbarkeit von theoretischem Wissen, wie es für den klassischen π-Kalkül existiert, eine vielversprechende Aufgabe. Für die Validierung von Modellen und Modelleigenschaften wäre beispielsweise ein exakt definierter Bisimulationsbegriff für den Kalkül der stochastischen β-Binder wünschenswert. Auch die Anwendung von Verfahren des Model Checking gewinnt in der Systembiologie derzeit an Bedeutung und stellt neue Herausforderungen an theoretische Wissenschaftler in diesem Bereich.

Da für Biologen und andere Wissenschaftler außerhalb der theoretischen Informatik Prozesse typischer Prozesskalküle relativ schwierig zu lesen und zu verstehen sind und dieser Umstand die Akzeptanz und Verbreitung der Herangehensweise behindert, besteht eine weitere Aufgabe für die Informatik in der Schaffung besserer Möglichkeiten zur Gliederung von Implementierungen. Beispielsweise lassen sich Konzepte aus der objektorientierten Programmierung in die Kalküle integrieren, wie in [KuLN_06], [KuDu_06], [KuNi_06] und [Kut_06] anhand des biochemischen stochastischen π-Kalküls demonstriert wird.

Literatur

[**ABCGM_07**] Bruno Apolloni, Simone Bassis, Alberto Clivio, Sabrina Gaito, Dario Malchiodi. *Modelling individual's aging within a bacterial population using a pi-calculus paradigm.* Natural Computing 6:33-53, 2007

[**BuGo_06**] Nadia Busi, Roberto Gorrieri. *On the Computational Power of Brane Calculi.* Transactions on Computational Systems Biology VI (LNBI 4220), 2006

[**Bus_06**] Nadia Busi. *Deciding Behavioural Properties in Brane Calculi.* Proc. Computational Methods in Systems Biology 2006 (LNBI 4210), 2006

[**Car1_05**] Luca Cardelli. *Abstract Machines of Systems Biology.* Transactions on Computational Systems Biology III (LNBI 3737), 2005

[**Car2_05**] Luca Cardelli. *Brane Calculi - Interactions of Biological Membranes.* Proc. Computational Methods in Systems Biology 2004 (LNBI 3082), 2005

[**CCDBM_06**] D. Chiarugi, M. Curti, P. Degano, G. Lo Brutto, R. Marangoni. *Feedbacks and Oscillations in the Virtual Cell VICE.* Proc. Computational Methods in Systems Biology 2006 (LNBI 4210), 2006

[**CCDM_05**] D. Chiarugi, M. Curti, P. Degano, R. Marangoni. *VICE: A VIrtual CEll.* Proc. Computational Methods in Systems Biology 2004 (LNBI 3082), 2005

[**ChFa_03**] Nathalie Chabrier, François Fages. *Symbolic Model Checking of Biochemical Networks.* Proc. Computational Methods in Systems Biology 2003 (LNCS 2602), 2003

[**CiPQ_05**] Federica Ciocchetta, Corrado Priami, Paola Quaglia. *Modeling Kohn Interaction Maps with Beta-Binders: An Example.* Transactions on Computational Systems Biology III (LNBI 3737), 2005

[**DaLa_04**] Vincent Danos, Cosimo Laneve. *Formal Molecular Biology.* Theoretical Computer Science 325(1):69-110, 2004

[**DaPr_05**] Vincent Danos, Sylvain Pradalier. *Projective Brane Calculus.* Proc. Computational Methods in Systems Biology 2004 (LNBI 3082), 2005

[**DPPQ_06**] Pierpaolo Degano, Davide Prandi, Corrado Priami, Paola Quaglia. *Beta-binders for biological quantitative experiments.* Electronic Notes in Theoretical Computer Science 164(3):101-117, 2006

[**EcLe_06**] Claudio Eccher, Paola Lecca. *Translating SBML Models into the stochastic π-calculus.* Transactions on Computational Systems Biology VII (LNBI 4230), 2006

[**Gil_77**] Daniel T. Gillespie. *Exact Stochastic Simulation of Coupled Chemical Reactions.* J. Phys. Chemistry 81 (25):2340-2361, 1977

[**HKNPT_06**] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn. *Probabilistic model checking of complex biological pathways.* Proc. Computational Methods in Systems Biology 2006 (LNBI 4210), 2006

[**Hof_03**] Christian Hofmann. *Typsysteme in Prozessalgebren am Beispiel des Join-, π- und Fusionkalküls.* Diplomarbeit, Technische Universität Dresden, 2003

[**KuLN_06**] Céline Kuttler, Joachim Niehren, Cédric Lhoussaine, Denys Duchier. *A stochastic Pi Calculus for Concurrent Objects.* Technical Report, INRIA, 2006

[**KuDu_06**] Céline Kuttler, Denys Duchier. *Biomolecular agents as multi-behavioural concurrent objects.* In Proc. 1st Int. Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord'05), Electronic Notes in Theoretical Computer Science 150:31-49, 2006

[**KuNi_06**] Céline Kuttler, Joachim Niehren. *Gene Regulation in the Pi Calculus: Simulating Cooperativity at the Lambda Switch.* Transactions on Computational Systems Biology VII (LNBI 4230), 2006

[**Kut_06**] Céline Kuttler. *Simulating Bacterial Transcription and Tanslation in a Stochastic π Calculus.* Transactions on Computational Systems Biology VI (LNBI 4220), 2006

[**LePr_03**] Paola Lecca, Corrado Priami. *Cell Cycle Control in Eukaryotes: A BioSpi model.* In Proceedings of BioConcur 2003, Electronic Notes in Theoretical Computer Science, 2003

[**MaVa_05**] Radu Mardare, Oleksandr Vagin, Paola Quaglia, Corrado Priami. *Model Checking Biological Systems described using Ambient Calculus.* Proc. Computational Methods in Systems Biology 2004 (LNBI 3082), 2005

[**MiBa_06**] Marino Miculan, Giorgio Bacci. *Modal Logics for Brane Calculus.* Proc. Computational Methods in Systems Biology 2006 (LNBI 4210), 2006

[**Mil_99**] Robin Milner. *Communicating and Mobile Systems: the π-calculus.* Cambridge University Press, 1999

[**NOMK_99**] Masao Nagasaki, Shuichi Onami, Satoru Miyano, Hiroaki Kitano. *Bio-calculus: Its concept and Molecular Interaction.* Genome Informatics, vol. 10(133-143), 1999, Universal Academy Press.

[**Par_01**] Joachim Parrow. *An Introduction to the π-Calculus.* In: Handbook of Process Algebra. Elsevier Health Sciences, 2001

[PeCo_03] Sabine Pérès, Jean-Paul Comet. *Contribution of CTL to Biological Regulatory Networks.* Proc. Computational Methods in Systems Biology 2003 (LNCS 2602), 2003

[PeRo_06] Mario J. Pérez-Jiménez, Francisco José Romero-Campero. *P Systems, a new computational Modelling Tool for Systems Biology.* Transactions on Computational Systems Biology VI (LNBI 4220), 2006

[PhCa_04] Andrew Phillips, Luca Cardelli. *A correct abstract machine for the Stochastic Pi-calculus.* Proceedings of Concurrent Models in Molecular Biology (Bioconcur'04), 2004

[PhCa_05] Andrew Phillips, Luca Cardelli. *A Graphical Representation for the Stochastic Pi-calculus.* Proceedings of Concurrent Models in Molecular Biology (Bioconcur'05), 2005

[PhCC_06] Andrew Phillips, Luca Cardelli, Giuseppe Castagna. *A Graphical Representation for Biological Processes in the Stochastic pi-Calculus.* Transactions on Computational Systems Biology VII (LNBI 4230), 2006

[PlHe_06] Helmut Plattner, Joachim Hentschel. *Zellbiologie. (3. Auflage)* Georg Thieme Verlag, 2006

[Pra_06] Davide Prandi. *A Formal Approach to Molecular Docking.* Proc. Computational Methods in Systems Biology 2006 (LNBI 4210), 2006

[Pri_95] Corrado Priami. *Stochastic π-calculus.* The Computer Journal, 6:578-589, 1995

[PrQu1_05] Corrado Priami, Paola Quaglia. *Beta Binders for Biological Interactions.* Proc. Computational Methods in Systems Biology 2004 (LNBI 3082), 2005

[PrQu2_05] Corrado Priami, Paola Quaglia. *Operational Patterns in Beta-Binders.* Transactions on Computational Systems Biology (LNBI 3380), 2005

[PRSS_01] Corrado Priami, Aviv Regev, Ehud Shapiro, William Silverman. *Application of a stochastic name-passing calculus to representation and simulation of molecular processes.* Information Processing Letters 80:25-31, 2001

[Reg_01] Aviv Regev. *Representation and simulation of molecular pathways in the stochastic π-calculus.* In Proc. 2nd workshop on Computation of Biochemical Pathways and Genetic Networks, 2001

[ReSh_02] Aviv Regev, Ehud Shapiro. *Cells as Computation - Cellular Abstractions.* Nature Vol. 419, 2002

[ReSh_04] Aviv Regev, Ehud Shapiro. *The π-calculus as an abstraction for biomolecular systems.* Published in Modelling in Molecular Biology (G. Ciobanu, G. Rozenberg), Natural Computing Series, Springer, 2004

[ReSS_00] Aviv Regev, William Silverman, Ehud Shapiro. *Representing Biomolecular Processes with Computer Process Algebra: π-Calculus programs of Signal Transduction Pathways.* Proc. Pacific Symp. of Biocomputing, 2000

[**ReSS_01**] Aviv Regev, William Silverman, Ehud Shapiro. *Representation and simulation of biochemical processes using the π-calculus process algebra.* Pacific Symposium on Biocomputing 6:459-470, 2001

[**RPSCS_04**] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, Ehud Shapiro. *BioAmbients: An abstraction for biological compartments.* Theoretical Computer Science 325(1), 2004

[**SaWa_01**] Davide Sangiorgi, David Walker. *The π-calculus: a theory of Mobile Processes.* Cambridge University Press, 2001

[**StBW_06**] Jason Steggles, Richard Banks, Anil Wipat. *Modelling and Analysing Genetic Networks: From Boolean Networks to Petri Nets.* Proc. Computational Methods in Systems Biology 2006 (LNBI 4210), 2006