SEBASTIAN MAUSER

SYNTHESE VON PETRINETZEN AUS HALBGEORDNETEN ABLÄUFEN

SYNTHESE VON PETRINETZEN AUS HALBGEORDNETEN ABLÄUFEN

Dissertation
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

der Fakultät für Mathematik und Informatik
der FernUniversität in Hagen

vorgelegt von Sebastian Mauser
aus Dachau

Hagen, September 2010

Sebastian Mauser:
Synthese von Petrinetzen aus halbgeordneten Abläufen,
Dissertation, September 2010

BERICHTERSTATTER:
Prof. Dr. Jörg Desel
Prof. Dr. Gabriel Juhás

ABSTRACT

The present thesis deals with the problem of synthesising a Petri net from a description of the behaviour of a system or a process given in the form of a set of partially ordered runs. More precisely, as an initial situation a specification of the possible runs of a system by a set of so called labelled partial orders (LPOs) is considered. A single LPO hereby consists of several events, which represent the occurrence of actions. However, in contrast to sequential models an LPO does not simply describe a sequence of events but it includes arbitrary dependencies and independencies resp. concurrency relations of events. Consequently, a set of partially ordered runs allows a precise and intuitive specification of the behaviour of a concurrent or a distributed system.
Based on such a specification it is searched for a model of the system in the form of a Petri net. Petri nets and domain specific dialects of Petri nets constitute a very well established formalism to model concurrent systems in an integrated way and to subsequently analyze and implement a system. The task of this thesis is the development of algorithms, which automatically generate a Petri net from a set of LPOs such that the behaviour of the net coincides with the given runs.
This topic is first motivated and introduced by different possible applications. Afterwards, a detailed theoretical discussion of the problem is presented. On the one hand the different facets of the problem are extensively investigated and on the other hand possible solution algorithms for the problem are elaborated and analyzed in detail.

ZUSAMMENFASSUNG

Die vorliegende Arbeit behandelt das Problem der Synthese eines Petrinetzes aus einer Verhaltensbeschreibung eines Systems oder Prozesses in der Form halbgeordneter Abläufe. Genauer gesagt wird als Ausgangssituation eine Spezifikation der möglichen Abläufe eines Systems durch eine Menge sog. beschrifteter partieller Ordnungen (BPOs) betrachtet. Eine einzelne BPO besteht hierbei aus mehreren Ereignissen, welche jeweils die Durchführung einer Aktion repräsentieren. Eine BPO beschreibt im Gegensatz zu sequentiellen Ablaufmodellen aber nicht einfach eine Folge von Ereignissen, sondern es können beliebige Abhängigkeiten und Unabhängigkeiten bzw. Nebenläufigkeiten von Ereignissen angegeben werden. Daher lässt sich das Verhalten eines nebenläufigen oder verteilten Systems mithilfe einer Menge von BPOs einerseits sehr genau und andererseits auch auf eine intuitive und leicht verständliche Art und Weise spezifizieren.
Ausgehend von einer derartigen Spezifikation wird dann ein Petrinetzmodell des Systems gesucht. Petrinetze und bereichsspezifische Dialekte von Petrinetzen stellen einen sehr weit verbreiteten und anerkannten Formalismus dar, um nebenläufige Systeme in einer integrierten Form zu modellieren und das modellierte System anschließend zu analysieren und gegebenenfalls zu implementieren. Die Aufgabe, mit der sich diese Arbeit befasst, besteht nun darin, Algorithmen zu entwickeln,

welche ausgehend von einer Menge von BPOs automatisch ein Petrinetz generieren, dessen Verhalten den gegebenen Abläufen entspricht.
Dieses Thema wird zunächst anhand verschiedener Anwendungsbeispiele motiviert und eingeführt. Anschließend folgt eine detaillierte theoretische Diskussion des Problems. Diese umfasst einerseits eine ausführliche Untersuchung der verschiedenen Facetten des Problems und andererseits eine sehr genaue Betrachtung und Analyse möglicher Lösungsverfahren für das Problem.

... Ich schlug ihm deshalb eine andere Modellierungstechnik vor, die Zuse so gut gefiel, dass er ein Buch darüber schrieb (Zuse, K.: Anwendungen von Petri-Netzen, Vieweg, 1982). Dies war die „Netztheorie verteilter Systeme", die bereits viele verschiedenartige erfolgreiche Anwendungen gefunden hatte, so im Bankwesen, Ökonomie, Telekommunikation, Workflow Management, Konfliktlösung, Prozess-Steuerung und Biochemie, nur leider noch nicht an ihrem Geburtsort, der Physik. Durch ihre äußerst einfachen Grundsätze (Axiome) und ihre graphische Ausdrucksweise macht sie komplizierte Zusammenhänge auch dem Nicht-Wissenschaftler zugänglich und bietet zugleich tiefgehende mathematische Analyse-Methoden an (für die Anwendung mittels Computer). ...

— Carl Adam Petri [184]

VORWORT

Der erste Eindruck eines Lesers über die vorliegende Dissertation ist sicherlich häufig, dass sie sehr umfangreich ist. Zumindest ist die Arbeit länger als dies für eine Dissertation im Fach Informatik normalerweise üblich ist. Eine lange Arbeit wird von vielen Lesern als eher negativ bewertet. So schrieb beispielsweise Johann Wolfgang von Goethe in einem Brief an seine Schwester: "Da ich keine Zeit habe, Dir einen kurzen Brief zu schreiben, schreibe ich Dir einen langen ...".
Dies ist bei meiner Dissertation allerdings keineswegs der Fall. Ziel meiner Arbeit ist es, das interessante Thema der Synthese von Petrinetzen aus halbgeordneten Abläufen umfassend zu behandeln. Im Laufe der Forschungsarbeiten hat sich das Thema als sehr facettenreich und vielseitig herausgestellt. Meine Arbeit versucht, dem gerecht zu werden, indem eine abgerundete Darstellung des Themas, welche alle wichtigen Aspekte berücksichtigt, präsentiert wird. Daraus resultiert der große Umfang der Dissertation. Dieser bedeutet aber nicht nur einen erhöhten Aufwand für den Leser der Arbeit, sondern ganz im Gegenteil zu Goethes Zitat erforderte die ausführliche Bearbeitung des Dissertationsthemas auch sehr viel Aufwand und Zeit für die Erstellung der Arbeit.
Glücklicherweise wurde ich dabei sehr gut unterstützt. Ich möchte hier zuerst dem Betreuer meiner Dissertation Prof. Dr. Jörg Desel danken. Er unterstützte mich nicht nur mit vielen wichtigen inhaltlichen Ideen, sondern stellte mir an seinem Lehrstuhl auch eine exzellente Arbeitsumgebung zur Verfügung. Auch dem Zweitgutachter der Dissertation Prof. Dr. Gabriel Juhás fühle ich mich sehr verbunden. Durch ihn bin ich ursprünglich erst zu dem Fach Informatik gekommen, so dass er sicherlich mehr Anteil an meiner Dissertation hat als dies für einen Zweitgutachter üblich ist. Darüberhinaus hat er mir auch inhaltlich wichtige Anregungen gegeben. Für die inhaltliche Unterstützung danke ich vor allem aber meinem ehemaligen Kollegen Prof. Dr. Robert Lorenz und meinem aktuellen Kollegen Robin Bergenthum. Beide haben in vielerlei Hinsicht sehr zu den Ergebnissen dieser Arbeit beigetragen. Des Weiteren möchte ich unseren ehemaligen Studenten Leo von Klenze, Thomas Irgang, Andreas Klett und Robin Löscher danken, welche mir als Hiwis bei dem Implementierungsteil der Dissertation geholfen haben. Der wichtigste Dank gilt schließlich aber meiner Familie, meinen

engen Freunden und meiner Lebensgefährtin, die mich ausnahmslos in meiner wissenschaftlichen Tätigkeit unterstützt und bekräftigt haben.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1

EINLEITUNG

Wir beginnen mit einer kleinen Geschichte (in Anlehnung an [131]). Sie beginnt auf dem ca. 20 Lichtjahre von der Erde entfernten Planten „Gliese 581 d“. Dieser 2007 entdeckte Planet ist der erste ernsthafte Kandidat für eine „Wasserwelt“, welche außerirdisches Leben beherbergen könnte. In unserer Geschichte lebt auf diesem Planeten, wie viele Alien-Fans es seit der Entdeckung vermuten, tatsächlich eine außerirdische Rasse. Ein junger, unerschrockener Einwohner des Planeten macht sich nun mit seinem UFO auf eine Abenteuerreise zur Erde. Bei der Erde angekommen, sieht er ein rotes Auto auf einer Straße stehen und setzt neben diesem zur Landung an. Als der in dem Auto sitzende Mann das UFO erblickt, erschrickt er natürlich sehr. Der Mann wählt mit seinem Handy sofort die Nummer der Polizei, gibt der Polizei den Standort der UFO-Sichtung durch und startet dann den Motor seines Autos, um sich in Sicherheit zu bringen. Der außerirdische Pilot des UFOs beobachtet dies und erstattet bei seinem Heimatplaneten Meldung: „Die Raumschiffe der Erdbewohner sind rot. Um sie zu starten, muss zuerst eine Nummer in ein Handy eingegeben werden und dann muss noch mit dem Handy gesprochen werden.“ Unser tapferer Außerirdischer ist mit seinen Schlussfolgerungen offensichtlich ein wenig voreilig. Die Nachricht des Außerirdischen ist, wie wir alle wissen, nicht ganz korrekt. Zuerst einmal handelt es sich um ein Auto und kein Raumschiff und natürlich sind auch nicht alle Autos auf der Erde rot. Uns interessiert hier aber der von dem Außerirdischen vermutete kausale Zusammenhang der Aktionen des Mannes im Auto (siehe Abbildung 1 rechts, wobei Abhängigkeiten durch Pfeile dargestellt sind). Er nimmt an, dass das Wählen der Nummer für das Sprechen mit dem Handy und das Sprechen mit dem Handy für das Starten des Autos notwendig ist. Tatsächlich ist es aber ja so, dass der Mann unabhängig vom Wählen der Polizei-Nummer und vom Melden des UFO-Standortes den Motor seines Autos starten kann (siehe Abbildung 1 links). Allerdings hat der Außerirdische bei seiner Nachricht zumindest insoweit Recht, dass das Wählen der Nummer vor dem Durchgeben des Standortes stattfinden muss, d.h. hier besteht tatsächlich eine kausale Abhängigkeit.

Das Problem des Außerirdischen in dieser Situation ist, dass Unabhängigkeit von Ereignissen typischerweise nicht beobachtet werden kann. Solange der Außerirdische nur sequentielle Abfolgen beobachtet, kann er die wahren Abhängigkeiten der Ereignisse nicht verstehen. Bestenfalls kann der Außerirdische durch die Beobachtung mehrerer sequentieller Abläufe feststellen, dass das Starten des Motors nicht nur nach dem Wählen und dem Melden, sondern auch nach dem Wählen und vor dem Melden sowie vor beiden Aktionen, dem Wählen und dem Melden, möglich ist (siehe Abbildung 2). Durch sequentielle Abfolgen lässt sich also Nichtdeterminismus beschreiben, d.h. in diesem Beispiel,

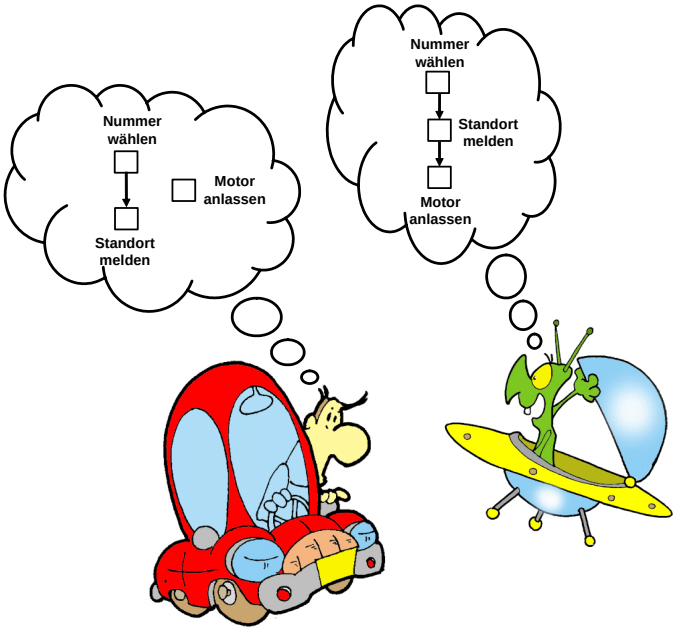

Abbildung 1: Begegnung mit einem Außerirdischen (Die Cliparts stammen von `www.clipproject.info`).

dass das Starten des Autos in beliebiger Reihenfolge zu den anderen beiden Ereignissen stattfinden kann. Aber auch hieraus lässt sich, wie wir gleich noch sehen werden, nicht unbedingt auf Unabhängigkeit der Ereignisse schließen. Zuerst kommen wir aber wieder zu unseren beiden Protagonisten zurück.

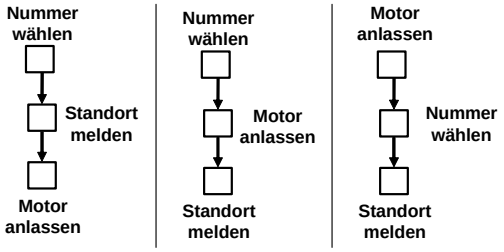

Abbildung 2: Nichtdeterminismus.

Der außerirdische Abenteurer begibt sich nun auf die Heimreise nach „Gliese 581 d“ und hinterlässt dem freundlichen Erdenbewohner mit dem roten Auto noch eine Dankes-Nachricht über seine gewonnenen Erkenntnisse. Da der Mann mit der Fehlinterpretation des Außerirdischen über das Verhalten der Menschen unzufrieden ist, will er das Missverständnis aufklären. Er möchte dem Außerirdischen über das populäre „Hello From Earth“-Projekt der NASA/CSIRO (`www.hellofromearth.net`), bei welchem jedermann eine Nachricht zu „Gliese 581 d“ schicken kann, die tatsächlichen Zusammenhänge seiner drei Aktionen, zu dem Zeitpunkt als der Außerirdische neben ihm landete, erklären. Die einzige Möglichkeit hierzu besteht darin, darzulegen, dass zwar eine Abhängigkeit zwischen der Aktion des Wählens der Nummer der Polizei und des Sprechens mit dem Handy besteht, jedoch das Anlassen des Autos zu diesen beiden Aktionen unabhängig

ist (siehe Abbildung 1 links). Eine solche Unabhängigkeit von Ereignissen wird auch als Nebenläufigkeit bezeichnet. Nebenläufigkeit kann mit einer sequentiellen Beschreibung von Aktionen nicht dargestellt werden. Insbesondere unterscheidet sich Nebenläufigkeit fundamental von Nichtdeterminismus. Es geht nicht nur um die Möglichkeit des Stattfindens von Ereignissen in verschiedenen Reihenfolgen, sondern um das unabhängige Stattfinden. Nebenläufige Ereignisse können auch (müssen aber nicht) parallel bzw. simultan auftreten. Beispielsweise ist es im Falle unseres Mannes im Auto möglich, dass er gleichzeitig telefoniert und den Motor anlässt.
An dieser Stelle könnten Frauen jetzt einwenden, dass Männern, wie es oft behauptet wird, die Multi-Tasking-Fähigkeit zur parallelen Durchführung zweier Aufgaben fehlt. Ohnehin ist es ja beispielsweise auch – vermutlich nicht nur aufgrund der mangelnden Multi-Tasking-Fähigkeiten von Männern – verboten, während des Autofahrens ein Handy zu benutzen. Soll also die Einschränkung berücksichtigt werden, dass der Mann im Auto niemals zwei Aktionen parallel zueinander ausführen kann, so ergibt sich gerade die zuvor dargestellte Situation des Nichtdeterminismus der Aktion des Startens des Autos gegenüber den zwei Aktionen des Wählens und Sprechens. Diese Situation wird dann durch die drei möglichen sequentiellen Abfolgen von Ereignissen aus Abbildung 2 geeignet beschrieben. Es kann aber nun mit einer solchen sequentiellen Beschreibung nicht zwischen dieser Situation und unserer ursprünglichen Situation unabhängiger Ereignisse unterschieden werden. Nebenläufigkeit lässt sich folglich mit sequentiellen Abläufen nicht geeignet darstellen. Außerdem ist es auch sehr umständlich sowie alles andere als intuitiv und natürlich, alle möglichen sequentiellen Abfolgen eines nebenläufigen Ablaufes zur Beschreibung eines solchen Ablaufes aufzuzählen. Insgesamt stellt Nebenläufigkeit also ein eigenes sehr bedeutsames Phänomen dar. Es lässt sich vereinfachend sagen, dass Nebenläufigkeit ein Eintreten von Ereignissen in beliebiger Reihenfolge sowie ein gleichzeitiges Eintreten von Ereignissen erlaubt. Nebenläufigkeit darf aber keinesfalls auf Gleichzeitigkeit reduziert werden. So zeigt das Beispiel deutlich, dass Nebenläufigkeit im Gegensatz zu Gleichzeitigkeit nicht transitiv ist. Das Anlassen des Motors ist zum Wählen und zum Melden nebenläufig, letztere zwei Aktionen sind aber voneinander abhängig.
Durch das Außerirdischen-Beispiel sollte deutlich geworden sein, dass sequentielle Modelle das reale Gefüge von Prozessen und Vorgängen, bei denen Nebenläufigkeit eine Rolle spielt, kaum widerspiegeln können. Es ist bei sequentiellen Modellen nicht ersichtlich, ob zwei Ereignisse nacheinander eintreten, weil das erste eine Voraussetzung des zweiten ist (wie im Falle des Wählens und des Meldens des UFOs) oder ob eine derartige zeitliche Ordnung nur zufällig besteht (wie im Falle des Meldens des UFOs und des Anlassens des Motors). Nun gewinnt aber Nebenläufigkeit in einer zunehmend dezentralen und verteilten Welt mehr und mehr an Bedeutung. Dem Studium der Nebenläufigkeit in der Informatik kommt vor allem aufgrund der Vielzahl von verteilten Systemen, Multiprozessorsystemen, Kommunikationsprotokollen und parallelen Arbeitsabläufen gerade in den letzten Jahren sehr viel Bedeutung zu, denn alle genannten Beispiele beinhalten fundamental nebenläufiges Verhalten.
Das einführende Beispiel zeigt, dass eine korrekte und verständliche Darstellung von Nebenläufigkeit schwierig sein kann. Daher sind zum

Studium großer komplizierter Systeme, welche das Phänomen der Nebenläufigkeit aufweisen, geeignete formale Beschreibungsmöglichkeiten unerlässlich. Die Modellierungstechnik der sog. Petrinetze ist einer der bekanntesten Formalismen, um solche Systeme konzeptuell darzustellen und wirkliche nebenläufige Systeme verschiedenster Anwendungsgebiete zu modellieren und zu analysieren. Insbesondere sind etliche moderne, teilweise weit verbreitete, anwendungsorientierte Modellierungssprachen für nebenläufige Systeme oder Prozesse nur leichte Abwandlungen des Formalismus der Petrinetze. Zumindest werden vielfach analoge Modellierungsprinzipien verwendet, so dass sich entsprechende Modellierungssprachen in Petrinetze übersetzen lassen. Bekannte Beispiel für Modellierungssprachen, welche sich auf Petrinetze zurückführen lassen, sind die von der OMG (Object Management Group) standardisierten Aktivitätsdiagramme der UML (Unified Modeling Language), deren Semantik sogar über Petrinetze definiert ist, die von Professor Scheer entwickelten EPKs (Ereignisgesteuerte Prozessketten) des ARIS-Konzepts (Architektur Integrierter Informationssysteme), die von IBM, BEA Systems und Microsoft eingeführte Sprache BPEL (Business Process Execution Language) bzw. WS-BPEL (für Webservices) oder die von der OMG und der BPMI (Business Process Management Initiative) gepflegte Spezifikationssprache BPMN (Business Process Modeling Notation). Vor allem im Bereich der Modellierung und Analyse von Geschäftsprozessen haben sich Dialekte von Petrinetzen in Theorie und Praxis de facto zum Standard entwickelt.
Das Konzept der Petrinetze hat seinen Ursprung in der Dissertation von Carl Adam Petri [183] aus den frühen sechziger Jahren des zwanzigsten Jahrhunderts. Das Ziel des damals neuartigen Ansatzes war es, wie Petri es formulierte, „möglichst viele Erscheinungen bei der Informationsübertragung und Informationswandlung in einheitlicher und exakter Weise zu beschreiben“ [183]. Seit der Arbeit von Petri wurde das Konzept der Petrinetze als formales Modell für nebenläufige Berechnungen in vielen tausend wissenschaftlichen Publikationen ausführlich untersucht und weiterentwickelt. Dennoch hat die Aussage von Petri noch immer Gültigkeit. So ist der wichtigste Grund für den Erfolg des Formalismus der Petrinetze sicherlich die Möglichkeit, das feine Zusammenspiel von Nebenläufigkeit und Nichtdeterminismus in Systemen geeignet darstellen zu können. Es können auf natürliche Weise Abhängigkeiten von Aktionen, Nebenläufigkeit und Alternativen repräsentiert werden. Ein weiterer Erfolgsgrund ist wohl auch die Kombination einer intuitiven graphischen Modellierungssprache und einer zugehörigen präzisen mathematischen Beschreibung. Während durch den ersten Aspekt die Vorteile der visuellen Wahrnehmung ausgenutzt werden, ermöglicht der zweite Aspekt formale Analysen von Petrinetzmodellen.
Abbildung 3 zeigt ein Beispiel eines sog. Stellen/Transitions-Netzes (S/T-Netzes). Ein solches Petrinetz ist ein gerichteter Graph mit Kantengewichten und zwei Arten von Knoten, den durch Kreise dargestellten Stellen und den durch Quadrate dargestellten Transitionen. Schwarze Marken in einer Stelle repräsentieren den lokalen Zustand einer Stelle. Eine Transition kann schalten, wenn in allen Stellen, welche durch eine ausgehende Kante mit der Transition verbunden sind, mehr Marken enthalten sind als durch das Gewicht der Kante angegeben ist. Durch einen Schaltvorgang werden entsprechend viele Marken aus diesen Stellen entfernt. Zu jeder Stelle, welche durch eine eingehende Kante

mit der Transition verbunden ist, wird dabei die durch das entsprechende Kantengewicht gegebene Anzahl an Marken hinzugefügt. So kann in dem Beispiel-Netz anfangs alternativ die Transition a oder die Transition b schalten, nicht jedoch die Transition c. Nach einem Schalten von a liegen zwei Marken in der Stelle p_2, so dass dann sogar a und b nebenläufig zueinander schalten können, d.h. es sind genug Marken für beide Schaltvorgänge vorhanden. Wir nehmen hier an, dass der Leser grundsätzlich mit Petrinetzen vertraut ist. Ansonsten sei auf Abschnitt 3.2 verwiesen. In diesem Abschnitt werden die notwendigen Grundlagen zu Petrinetzen kurz dargestellt.

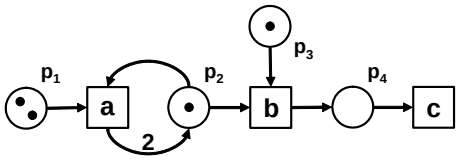

Abbildung 3: Ein markiertes S/T-Netz.

Das Verhalten eines Systems ist durch einzelne Ausführungen bzw. Abläufe des Systems gegeben. Ein einzelner Ablauf ist dabei deterministisch. Um das gesamte Verhalten eines Systems darzustellen, müssen alle alternativen Abläufe eines Systems angegeben werden. Eine bedeutsame Möglichkeit zur Repräsentation des Verhaltens eines nebenläufigen Systems stellen partielle Ordnungen bzw. genauer gesagt beschriftete partielle Ordnungen (BPOs) dar. Eine BPO modelliert einen Ablauf eines Systems, indem Abhängigkeiten zwischen Ereignissen spezifiziert werden. Ein Beispiel für eine BPO, bei der zwei Ereignisse voneinander abhängig sind und ein drittes Ereignis zu diesen beiden Ereignissen unabhängig ist, haben wir schon im Rahmen der Außerirdischen-Geschichte betrachtet (siehe Abbildung 1 links). Dabei wurde insbesondere deutlich, dass BPOs im Rahmen der Beschreibung des Verhaltens nebenläufiger Systeme wesentlich geeigneter sind als beispielsweise sequentielle Ablaufmodelle. So ist der wichtigste Punkt sicherlich, dass BPOs eine intuitive, ausdrucksmächtige und natürliche Art der Repräsentation nebenläufigen Verhaltens darstellen. Vor allem kann mit BPOs wahre Nebenläufigkeit repräsentiert werden, d.h. es können beliebige Abhängigkeiten und daraus resultierende Nebenläufigkeitsbeziehungen spezifiziert werden. Dahingegen lassen sich mit sequentiellen Verhaltensbeschreibungen gar keine Nebenläufigkeitsbeziehungen explizit darstellen und es ist sehr umständlich und auch nur heuristisch möglich, Nebenläufigkeitsbeziehungen aus derartigen Verhaltensbeschreibungen zu erzeugen. Insbesondere ist die Ausdrucksmächtigkeit sequentieller Modelle nicht ausreichend, um Nebenläufigkeit von Nichtdeterminismus, beispielsweise aufgrund einer geteilten Ressource, unterscheiden zu können. Andere Verhaltensbeschreibungen wie Schrittfolgen, Schritt-Transitionssysteme oder Terme erlauben nur die Darstellung von eingeschränkten Nebenläufigkeitsbeziehungen. Aus solchen Darstellungen dann die wahren Nebenläufigkeiten zu erkennen, ist ebenfalls nur schwer möglich. Neben diesem zentralen für BPOs sprechenden Modellierungsargument ist eine Darstellung von Verhalten in der Form von BPOs häufig auch sehr effizient. So repräsentiert eine einzelne BPO eine Menge von sequentiellen Abläufen, welche sehr groß, sogar exponentiell, sein kann, wenn nebenläufiges Verhalten

modelliert wird. Dies gilt aber wiederum nicht nur für sequentielle Abläufe, sondern beispielsweise auch für Schrittfolgen oder Terme. Gerade aus diesem Grund sind vielfach Analysetechniken für BPOs effizienter als für andere, vor allem sequentielle, Verhaltensmodelle.
Es ist daher nicht verwunderlich, dass sich im Rahmen der Modellierung von Abläufen nebenläufiger Systeme in Anwendungen hauptsächlich auf partiellen Ordnungen basierende Modellierungssprachen entwickelt haben. Allerdings wird in Anwendungskontexten meist von Szenarien und weniger von Abläufen gesprochen. Die Begriffe Szenario und Ablauf werden in der Literatur häufig synonym verwendet, wobei der Begriff Szenario manchmal in dem Sinne weniger streng als der Ablaufbegriff verwendet wird, dass ein Szenario beispielsweise im Kontext von UML-Sequenzdiagrammen auch bestimmte Arten von Nichtdeterminismus beinhalten kann. Tatsächlich lassen sich nun die meisten Anwendungsformalismen zur Beschreibung von Szenarien nebenläufiger Systeme auf BPOs abbilden. Beispiele hierfür sind die schon genannten UML-Sequenzdiagramme, die von der ITU-T – dem Telekommunikations-Standardisierungs-Sektor der Internationalen Fernmeldeunion – standardisierten MSCs (Message Sequence Charts) oder die sog. Instanz-EPKs von ARIS. Insbesondere zur Darstellung des Verhaltens von Petrinetzmodellen verteilter Systeme werden partiell geordnete Abläufe sowohl aus theoretischer als auch aus praktischer Hinsicht häufig als der intuitivste und natürlichste Modellierungsansatz angesehen.
Abbildung 4 zeigt zwei BPOs, welche das gesamte Verhalten des in Abbildung 3 dargestellten Netzes repräsentieren. Die beschrifteten Knoten modellieren Schaltereignisse des Netzes. Die Kanten modellieren die Reihenfolge bzw. die Ordnung der Schaltereignisse. Ungeordnete Ereignisse sind unabhängig bzw. nebenläufig zueinander. So stellt die erste BPO dar, dass zuerst die Transition b und danach dann die Transition c in dem Netz schalten kann. Die zweite BPO repräsentiert einen alternativen Ablauf des Netzes. Es kann zuerst a schalten, danach können dann a und b nebenläufig schalten und nach dem Schalten von b aber immer noch nebenläufig zu dem zweiten a-Ereignis kann die Transition c schalten. Für Leser, welchen an dieser Stelle ein intuitives Verständnis für die Darstellung von Abläufen durch BPOs nicht ausreicht, sei auf Abschnitt 3.3 verwiesen. Dort werden entsprechende Grundlagen insbesondere im Rahmen der Modellierung von Abläufen von Petrinetzen erläutert. An dieser Stelle sei auch noch darauf hingewiesen, dass das in Abbildung 3 dargestellte Netz und die zwei in Abbildung 4 dargestellten Abläufe des Netzes in den weiteren Ausführungen der Arbeit immer wieder als Beispiele herangezogen werden.[1]
Nachdem wir nun das für diese Arbeit zentrale Phänomen der Nebenläufigkeit motiviert haben und in diesem Kontext die für uns wesentlichen Modellierungskonzepte des Petrinetzes und der BPO eingeführt haben, soll im folgenden Unterabschnitt anhand eines im Gegensatz zu dem UFO-Beispiel sehr irdischen und realen illustrativen Beispiels zur eigentlichen Problemstellung dieser Arbeit hingeführt werden. Während wir gerade diskutiert haben, wie sich das Verhallten eines gegebenen Petrinetzes durch BPOs beschreiben lässt, werden wir nun die

1 Für die Gutachter werden der Arbeit zwei lose Blätter mit diesen zwei Abbildungen und einigen weiteren besonders häufig oder in verschiedenen Abschnitten referenzierten Abbildungen beigelegt.

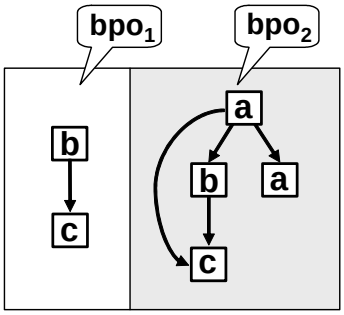

Abbildung 4: Zwei BPOs.

umgekehrte Frage, nämlich wie sich zu einem gegebenen Verhalten in der Form von BPOs ein zugehöriges Petrinetz finden lässt, motivieren.

1.1 EINFÜHRENDES BEISPIEL

Als Beispiel soll hier ein einfacher Geschäftsprozess betrachtet werden. Passend zu einer Dissertation betrachten wir einen vereinfachten Begutachtungsprozess für eine Doktorarbeit. Das Ziel ist es, ein Petrinetzmodell dieses Prozesses zu erstellen. Wir nehmen an, dass Beschreibungen der möglichen Szenarien dieses Prozesses vorliegen. Dies ist eine typische Ausgangssituation, da Szenarien häufig als das einfachste, intuitivste und geeignetste Konzept für eine erste meist noch informale Beschreibung eines Systems oder eines Prozesses angesehen werden. Insbesondere können einzelne Szenarien typischerweise leichter verstanden und dargestellt werden als der Gesamtprozess bzw. der Gesamtprozess ist einzelnen Beteiligten oft ohnehin gar nicht bekannt. Außerdem können Szenarien auch verteilt von verschiedenen Personen und Informationsquellen gesammelt werden.
Es gibt drei mögliche Begutachtungsszenarien. Diese stellen sich wie folgt dar. In jedem Fall wird mit der Einreichung der Dissertation begonnen. Anschließend wird von dem Erstgutachter und von dem Zweitgutachter unabhängig voneinander jeweils ein Gutachten erstellt. Sind beide Gutachten positiv, so wird die Dissertation angenommen und anschließend das Promotionsverfahren entsprechend der geltenden Regelungen abgeschlossen. Letztere Aufgabe kann beispielsweise eine mündliche Prüfung, die Erstellung eines Abschlusszeugnisses und die Veröffentlichung der Dissertation umfassen. Nun ist es aber natürlich auch möglich, dass eines der Gutachten negativ ausfällt. In diesem Fall wird die Dissertation abgelehnt. Es gibt zwei mögliche Szenarien, welche zur Ablehnung der Dissertation führen. Einerseits kann das Erstgutachten negativ sein. Dann wird die Dissertation unabhängig von dem Zweitgutachten abgelehnt. Andererseits soll für das Zweitgutachten genau dasselbe gelten, d.h. ein negatives Zweitgutachten führt ebenfalls unabhängig von dem Erstgutachten zu einer Ablehnung. Die Situation stellt sich also so dar, dass eine Ablehnungsentscheidung für die Dissertation jeweils abhängig von nur einem negativen Gutachten getroffen wird, d.h. eine solche Entscheidung ist dann unabhängig von dem jeweils anderen Gutachten. Auch nach der Ablehnung der Dissertation muss das Promotionsverfahren regelgerecht abgeschlossen werden, beispielsweise durch die Anfertigung eines abschließenden

Urteilsschreibens oder Belehrungen bzgl. einer Neuzulassung zur Promotion. Obwohl das Ablehnen der Dissertation, wie erläutert, schon nach der Fertigstellung nur eines Gutachtens möglich ist, soll das Abschließen des gesamten Promotionsverfahrens allerdings erst nach der Fertigstellung beider Gutachten möglich sein.
Ausgehend von einer derartigen Beschreibung der möglichen Szenarien des Begutachtungsprozesses bietet sich eine Übersetzung der Szenarien in eine formale Form an. Als Szenario-Modell eignen sich, wie zuvor diskutiert, insbesondere BPOs. Eine Repräsentation der beschriebenen Szenarien durch BPOs ist intuitiv möglich und sollte selbst unerfahrenen Modellierern wenig Probleme bereiten. Dies liegt daran, dass die Instanzebene einzelner Abläufe sehr einfach und klar ist. Eine BPO repräsentiert einen einzelnen Ablauf des Prozesses, indem Ordungsbeziehungen zwischen Ereignissen angegeben werden, wobei Ereignisse das Eintreten entsprechender Aktivitäten modellieren. Wie oben beschrieben, besitzt der Begutachtungsprozess drei Abläufe, nämlich einen Ablauf der zur Annahme der Dissertation führt und zwei mögliche Abläufe, welche zur Ablehnung der Dissertation führen. Die letzteren zwei Abläufe unterscheiden sich darin, aus welchem der beiden Gutachten die Ablehnungsentscheidung resultiert. Die drei Abläufe des Prozesses sind in Abbildung 5 als BPOs dargestellt.

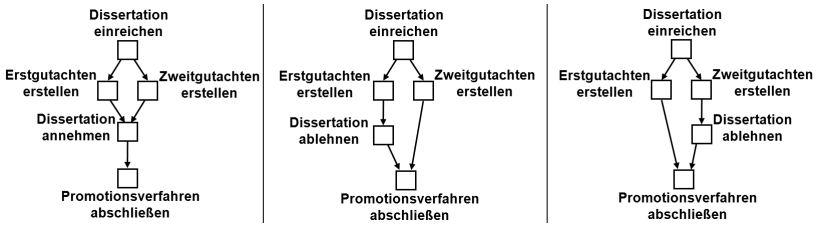

Abbildung 5: Abläufe des Begutachtungsprozesses.

Das Problem ist nun, dass eine solche Sammlung einzelner Ablaufmodelle eines Prozesses das Gesamtverhalten des Prozesses nicht in einer übersichtlichen und strukturierten Art und Weise abbilden kann. Daher sind für die typischen Ziele bei der Modellierung von Prozessen, wie Dokumentation, Analyse, Simulation, Optimierung oder Implementierung eines Prozesses, normalerweise integrierte zustandsbasierte Prozessmodelle erforderlich. Die Erstellung eines solchen Prozessmodells ist aber häufig eine schwierige, aufwändige und fehleranfällige Aufgabe.
Abbildung 6 zeigt ein entsprechendes Petrinetzmodell des beschriebenen Begutachtungsprozesses. Während die einzelnen zuvor diskutierten Szenarien des Prozesses noch einfach verständlich sind, weist das integrierte Modell des Gesamtprozesses einen komplexen und eher schwer einzusehenden Kontrollfluss auf. Dies wird insbesondere durch die komplizierte Stelle mit drei eingehenden und drei ausgehenden Kanten deutlich. So produzieren die zwei Gutachten-Transitionen je eine Marke in der Stelle. „Dissertation annehmen“ konsumiert dann diese zwei Marken aus der Stelle, während „Dissertation ablehnen“ nur eine Marke konsumiert. Dies liegt daran, dass die erstere Aufgabe erst nach Erstellung beider Gutachten stattfinden kann, während die zweite Aufgabe nur ein Gutachten erfordert. „Dissertation annehmen“ produziert zudem auch eine Marke in der Stelle, so dass sich nach der

Erstellung beider Gutachten und der Annahme oder der Ablehnung der Dissertation immer genau eine Marke in der Stelle befindet. „Promotionsverfahren abschließen“ konsumiert schließlich diese Marke aus der Stelle. Dies ist notwendig, da diese Aufgabe neben der Annahme oder der Ablehnung der Dissertation auch beide Gutachten erfordert. Während „Dissertation annehmen“ zwar ohnehin sicherstellt, dass beide Gutachten fertiggestellt sind, ist dies bei der Aufgabe „Dissertation ablehnen“ nicht der Fall.

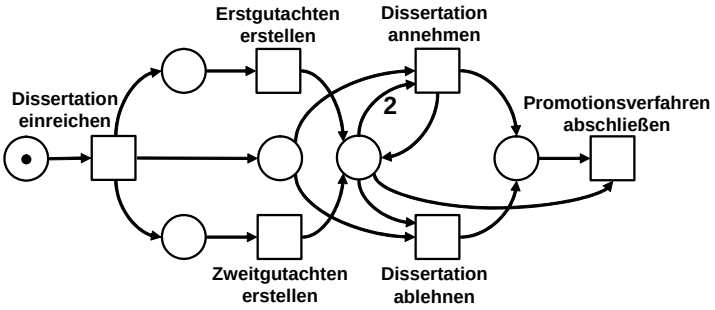

Abbildung 6: Petrinetzmodell des Begutachtungsprozesses.

Es zeigt sich also, dass schon bei diesem kleinen Beispiel eine hohe Komplexität des integrierten Prozesses vorliegt. Das beschriebene Verhalten lässt sich auch nicht auf eine einfachere oder kompaktere Art und Weise durch ein Petrinetz darstellen. Es wird daher deutlich, dass selbst bei einem eher kleinen Beispiel eine händische Konstruktion eines entsprechenden Petrinetzmodells sehr schwierig sein kann. Dahingegen ist die Erstellung der BPO-Modelle für die drei Szenarien des Prozesses relativ einfach. Aus diesen Überlegungen ergibt sich nun die Fragestellung dieser Arbeit: Wie lässt sich ein zugehöriges Petrinetzmodell automatisch aus entsprechenden BPOs generieren? Diese Frage wird nun losgelöst von dem betrachteten Beispiel diskutiert und näher erläutert.

1.2 PROBLEMSTELLUNG

Generell gilt, dass für die ersten Schritte zur Modellierung eines Systems Szenarien sehr häufig das intuitivste und geeignetste Modellierungskonzept darstellen. Die Instanzebene einzelner Szenarien, bei denen noch keine Konflikte berücksichtigt werden, ist meist sehr leicht verständlich. Allerdings lassen sich mithilfe lose gekoppelter Szenario-Modelle sowohl händische als auch automatische Analysen nur schwer durchführen. Daher sind für typische Anwendungen, bei denen Modellierung eine Rolle spielt, normalerweise integrierte Systemmodelle erforderlich. Damit ergibt sich das Problem, wie die Kluft zwischen der benutzerorientierten Szenariosicht eines Systems und einem implementierungsnahen integrierten Model des Systems überbrückt werden soll. Zur Lösung des Problems sind vor allem automatische Verfahren zur Konstruktion eines Systemmodells aus einer Menge von Szenarien, welche Konsistenz zwischen dem Systemmodell und den Szenarien garantieren, vielversprechend. Das Problem der Diskrepanz zwischen Szenariospezifikationen und integrierten Systemmodellen wurde insbesondere im Bereich des Softwareengineering diskutiert, aber auch in

anderen Anwendungsfeldern wie der Geschäftsprozessmodellierung, dem Hardwaredesign oder der Kontrollersynthese finden sich derartige Fragestellungen.
In all diesen Bereichen sind Petrinetze und anwendungsspezifische Dialekte von Petrinetzen eine beliebte Wahl zur integrierten Modellierung von Systemen. Dies gilt umso mehr, wenn Nebenläufigkeit eine wichtige Rolle spielt. In diesem Kontext stellt sich nun die Frage nach Algorithmen zur Konstruktion eines Petrinetzes, dessen Verhalten einer gegebenen Menge von Szenarien entspricht. Diese Fragestellung ist im Bereich der Petrinetztheorie als Syntheseproblem bzgl. Sprachäquivalenz bekannt. Dabei wird eine Menge von Szenarien als Sprache bezeichnet. Es muss allerdings beachtet werden, dass es verschiedene Möglichkeiten gibt, Szenarien zu spezifizieren. Szenarien können durch Sequenzen von Ereignissen spezifiziert sein, es können zusätzlich nebenläufige Schritte von Ereignissen berücksichtigt werden oder es können halbgeordnete Abläufe von Ereignissen betrachtet werden. Halbgeordnete Abläufe werden oft als die natürlichste und geeignetste Repräsentation des Verhaltens von Petrinetzmodellen nebenläufiger Systeme bezeichnet. Insbesondere erlauben sie eine intuitive Spezifikation von wahrer (nicht nur auf Schritte beschränkter) Nebenläufigkeit in Szenarien. Daher werden in dieser Arbeit Szenarien betrachtet, welche durch halbgeordnete Abläufe in der intuitiven Form von BPOs gegeben sind. Eine Menge solcher Szenarien wird im Folgenden als partielle Sprache bezeichnet. Ziel dieser Arbeit ist es, das Problem der Synthese eines Petrinetzes aus einer partiellen Sprache, so dass das Ablaufverhalten des Netzes der partiellen Sprache entspricht, umfassend zu behandeln.
Das Beispiel des Begutachtungsprozesses hat gezeigt, dass es oftmals sehr herausfordernd ist, unmittelbar händisch ein Petrinetzmodell eines Systems zu erstellen. Dahingegen ist das Modellieren von Szenarien in der Form von BPOs häufig relativ einfach. Mit entsprechenden Syntheseverfahren lässt sich dann schnell und ohne weiteren Aufwand automatisch ein Petrinetz erzeugen, welches die Szenarien getreu widerspiegelt. Ein solches Vorgehen, bei welchem die Hauptaufgabe für einen Modellierer in der Sammlung von Szenarien und der Erstellung von zugehörigen BPO-Modellen liegt, das eigentlich gesuchte integrierte Modell dann aber automatisch mithilfe entsprechender Synthesewerkzeuge generiert wird, bezeichnen wir als streng ablauforientierten Modellierungsansatz. Die Vorteile dieses Modellierungsansatzes wurden schon in dem kleinen Beispiel des letzten Unterabschnittes deutlich und gelten für größere Fallbeispiele umso mehr. Insbesondere in einem realen typischerweise sehr komplexen, informalen und verteilten Umfeld ist die Sammlung einzelner Szenarien und die Erstellung zugehöriger valider Szenario-Modelle meist wesentlich einfacher als die Konstruktion von Modellen, welche das gesamte System valide abbilden. Dies liegt vor allem daran, dass Szenarien einerseits ein System aus der Sicht der Nutzer beschreiben bzw. für Nutzer intuitiv verständlich sind und andererseits unabhängig voneinander von verschiedenen auch modellierungsunerfahrenen Nutzern spezifiziert werden können. Die Fragestellung der Synthese eines Petrinetzes aus einer partiellen Sprache ist in diesem Kontext gerade auch deswegen ein sehr interessantes und bedeutsames Problem, da Petrinetze, wie erläutert, eine sehr wichtige Modellierungssprache für nebenläufige Systeme und Prozesse darstellen und eine partielle Sprache ein entsprechend geeigneter

Formalismus zur Spezifikation von Szenarien solcher Systeme ist. In der Literatur werden auch etliche andere Syntheseprobleme für Petrinetze betrachtet. Da BPOs das natürlichste Konzept zur Beschreibung des Verhaltens von Petrinetzen darstellen, hat die Fragestellung dieser Arbeit im Rahmen von Syntheseproblemen für Petrinetze allerdings besondere Relevanz.

1.3 LITERATURÜBERSICHT

Generell stellt die Synthese von Petrinetzen aus Verhaltensbeschreibungen seit den 1990ern ein erfolgreiches und viel beachtetes Forschungsgebiet dar. In der Literatur finden sich auf der einen Seite eine Vielzahl sehr interessanter nichttrivialer theoretischer Ergebnisse und auf der anderen Seite auch einige wichtige industrielle Anwendungen, insbesondere in den Bereichen der Hardware-Synthese und der Geschäftsprozessmodellierung. Schließlich gibt es auch einige fortschrittliche Synthese-Werkzeuge.
Das klassische Syntheseproblem besteht in der Frage, ob es zu einer gegebene Verhaltensspezifikation ein Petrinetz (mit eindeutigen Transitionsnamen) gibt, welches das betrachtete Verhalten aufweist. Im positiven Fall wird dann die Konstruktion eines entsprechenden Zeugen-Netzes gefordert. Ursprünglich bezog sich das Problem der Synthese eines Petrinetzes auf sequentielle Beobachtungen. So wurde es erstmals in [89, 90] ausgehend von endlichen Graphen betrachtet. Die Autoren entwickelten eine Charakterisierung derjenigen Graphen, welche isomorph zu dem Erreichbarkeitsgraph eines elementaren Petrinetzes sind. Aufbauend auf diesen Arbeiten wurden einige interessante Synthesealgorithmen entwickelt [36, 117, 78, 60], wobei letztere Arbeit Bisimilarität anstelle von Isomorphie fordert (auch die Aufteilung einer einzelnen Beschriftung auf mehrere Transitionen wird in dieser Arbeit untersucht). Es stellte sich heraus, dass das betrachtete Problem NP-vollständig ist [14]. Daher wurden auch etliche heuristische Optimierungen der Algorithmen vorgenommen [58, 59]. Dies führte zu Verfahren, welche für viele praktische Beispiele sehr effizient sind.
Die dargestellten ursprünglichen Ergebnisse zur Synthese von Petrinetzen wurden im Wesentlichen in zwei Richtungen weiterentwickelt. So wurden einerseits neben elementaren Netzen auch etliche allgemeinere Netzklassen betrachtet und andererseits wurden auch andere Arten von Verhaltensbeschreibungen untersucht. Die Verhaltensbeschreibungen betreffend muss insbesondere unterschieden werden, ob ein Netz synthetisiert werden soll, dessen Erreichbarkeitsgraph isomorph zu einem gegebenen Graphen ist oder ob ein Netz gesucht ist, dessen mögliche Abläufe einer gegebenen Sprache entsprechen. Das erstere Problem wird als Syntheseproblem bzgl. Isomorphie, das zweite als Syntheseproblem bzgl. Sprachäquivalenz bezeichnet. Die Forderung des Syntheseproblems bzgl. Isomorphie ist dabei substantiell strenger als diejenige des Syntheseproblems bzgl. Sprachäquivalenz [13]. Grundlegende Charakterisierungen im Rahmen der Synthese bzgl. Isomorphie finden sich in [90, 176] für elementare Netze bzw. Bedingungs-Ereignis-Netze und in [173] für Stellen/Transitions-Netze. Weiterführende Überlegungen zu letzterer Arbeit finden sich in [85, 37, 13]. In [16, 17, 18] wurden sog. Netztypen betrachtet, welche eine in der Netzklasse parametrische Petrinetzdefinition ermöglichen. Diese wurden in [68] auch noch um sog. Schritt-Schalt-Politiken verallgemeinert. Weiter wurde

in [47, 185, 186] die Synthese elementarer Netze mit Inhibitorkanten untersucht und in [146] wurden zusätzlich noch Lesekanten und Lokalitäten berücksichtigt. In den meisten dieser Arbeiten wird nicht nur sequentielles Verhalten betrachtet, sondern auch Schrittfolgen, d.h. es werden Schritt-Transitionssysteme [173] verwendet. Die Synthese bzgl. Sprachäquivalenz wurde insgesamt in der einschlägigen Literatur weniger untersucht. Eine wichtige Arbeit hier ist [120], welche sog. Trace-Sprachen für Stellen/Transitions-Netze diskutiert. Weiter wurde die Synthese von S/T-Netzen aus sequentiellen Sprachen in den Arbeiten [13, 18] angesprochen und ausführlich dann in [63, 65] dargestellt. Für uns sind schließlich Mengen von BPOs, also partielle Sprachen, besonders interessant. BPOs werden auch als partielle Worte [108] oder partiell geordnete Multimengen [188] bezeichnet. Partielle Sprachen wurden im Rahmen der Synthese erstmals in [132] im Zusammenhang mit S/T-Netzen untersucht. Diese Arbeit wurde in [153] fortgesetzt. Es gibt mehrere hierzu verwandte Ansätze. So beschäftigen sich einige Arbeiten mit der Beziehung von sog. Ereignis-Strukturen und Petrinetzen [175, 226]. Hier ist insbesondere der Artikel [121], welcher S/T-Netze ohne Selbst-Nebenläufigkeit und sog. lokale Ereignis-Strukturen untersucht, zu nennen. Außerdem betrachten die Arbeiten [213, 177] den Zusammenhang zwischen elementaren Netzen und regulären Ereignis-Strukturen und die Arbeit [103] diskutiert wiederum S/T-Netze und Ereignis-Strukturen. In [147] werden partielle Sprachen, welche von sicheren Netzen mit beschrifteten Transitionen generiert werden, charakterisiert. Eine spezielle Klasse von partiellen Sprachen wird in [151, 152] in Beziehung zu sog. Verzweigungs-Automaten, welche wiederum als spezielle Klasse von beschrifteten Netzen interpretiert werden können, gesetzt.
An algorithmischen Weiterführungen dieser Arbeiten ist insbesondere [13] zu nennen. Dort wird das Problem der Synthese eines puren, beschränkten S/T-Netzes aus endlichen Graphen und regulären Sprachen untersucht. In diesem Kontext war es nun mithilfe linearer Programmierungsverfahren [207] sogar möglich, polynomielle Synthesealgorithmen zu entwickeln. In [17] wurden diese Verfahren im Kontext von Graphen auf allgemeine beschränkte S/T-Netze unter der Berücksichtigung von Schritt-Verhalten erweitert. Eine gute Zusammenfassung derartiger Algorithmen findet sich in [18]. Verfahren für unbeschränkte Netze sind in [63, 65] dargestellt. In diesen Arbeiten wird auch die Entscheidbarkeit des Syntheseproblems für S/T-Netze bzgl. etlicher Klassen von unendlichen Sprachen und Graphen geklärt. Insbesondere die Synthese bzgl. Sprachäquivalenz wird hier detailliert und systematisch untersucht. In [50] wird noch einmal speziell auf High-Level Message-Sequence-Charts als Eingabe im Rahmen der Synthese von Petrinetzen eingegangen. Die Arbeiten [54, 53] verfolgen einen anderen Forschungsstrang. In diesen Arbeiten wird der Synthese-Ansatz aus [60] für elementare Netze derart erweitert, dass k-beschränkte S/T-Netze aus Transitionssystemen synthetisiert werden können. Das resultierende Verfahren ist zwar wiederum exponentiell, hat aber für kleine k oftmals eine sehr gute Laufzeit verglichen mit den polynomiellen Verfahren aus [18]. Schließlich ist in der Arbeit [153] ein algorithmischer Ansatz zur Synthese von S/T-Netzen aus endlichen partiellen Sprachen skizziert. Des Weiteren beschäftigt sich noch die Arbeit [180] in dem Kontext von partiellen Sprachen mit einigen Entscheidbarkeitsproblemen für k-beschränkte Netze.

Neben den Erweiterungen der ursprünglichen Synthese-Konzepte auf weitere Netzklassen und Verhaltensbeschreibungen, wurden in einigen Arbeiten auch darüber hinausgehende neue Synthese-Fragestellungen und Synthese-Aspekte untersucht. So werden in [63, 64] obere und untere Verhaltens-Schranken für Netze betrachtet. In [48, 65, 69] werden Einschränkungen bzgl. der Anzahl und der Struktur der Stellen des synthetisierten Netzes berücksichtigt. Die Arbeit [65] spricht das Problem approximativer Synthese-Lösungen an. In [48, 69, 21, 193, 199, 67] werden Optimierungen bzw. Kostenaspekte für das zu erstellende Netz diskutiert. Der Artikel [19] behandelt die Spezifikation einer Menge von Graphen zu Synthesezwecken. In [64, 15, 66] wird ein Bezug zur Kontroller-Synthese hergestellt und die Spezifikation verteilter Komponenten bei der Synthese berücksichtigt. Schließlich finden sich noch einige Varianten der klassischen Synthesefragestellungen in Arbeiten, welche sich im Rahmen des sog. Process-Mining mit Synthese beschäftigen [1, 69, 7, 84, 211, 51, 53]. Die Haupt-Aufgabe des Process-Mining besteht darin, ein Petrinetz oder ein ähnliches Prozessmodell aus einem sog. Ereignis-Log, welches eine sequentielle Sprache definiert, zu generieren.
Verlassen wir den Bereich der Synthese von Petrinetzen, so gibt es etliche verwandte Syntheseansätze für verschiedenste Arten von asynchronen und verteilten Systemen. Ein guter Überblick über derartige Ansätze findet sich in [67]. In Anwendungen spielt insbesondere die Synthese von zustandsbasierten Modellen aus Szenarien eine wichtige Rolle. Die Arbeiten [9, 150] geben einen Überblick über derartige Syntheseansätze.
Aufbauend auf den genannten theoretischen Resultaten im Bereich der Petrinetz-Synthese gibt es auch etliche industrielle Anwendungen und Fallstudien. Das am weitesten entwickelte Anwendungsfeld ist das Design von Hardware-Systemen, genauer gesagt asynchronen Kontroll-Schaltungen [59, 130]. Ein weiteres wichtiges Anwendungsfeld ist die Geschäftsprozessmodellierung [6, 72], insbesondere das schon angesprochene Process-Mining [1, 69, 7, 84, 211, 51, 53]. Daneben gibt es noch Anwendungen im Bereich der Steuerung von Produktionssystemen [227] bzw. generell zur verteilten Steuerung von Systemen [189, 66, 64, 102, 193, 21] sowie im Rahmen der Modellierung von Kommunikations-Protokollen und verteilter Software [64, 15, 49].
In der vorliegenden Arbeit diskutieren wir Anwendungsmöglichkeiten von Syntheseverfahren im Rahmen des schon kurz skizzierten streng ablauforientierten Modellierungsansatzes. Es gibt einige Vorarbeiten zum Thema der ablauforientierten Modellierung, hauptsächlich aus dem Bereich des Softwareengineering. Die grundlegende Arbeit in diesem Kontext ist [123]. Einen guten Überblick über die Problematik bieten die Bücher [187, 110] sowie die Überblicksarbeiten [9, 150]. Wichtige, über den Bereich des Softwareengineering hinausgehende, Vorarbeiten sind [72, 73, 3, 105]. In dieser Arbeit konzentrieren wir uns auf drei Anwendungsgebiete für Modellierung. Neben den schon angesprochenen Gebieten der Geschäftsprozessmodellierung und des Softwareengineering, diskutieren wir auch den Anwendungsbereich der Lernprozessmodellierung. Die spezifische, relevante Literatur zu all diesen drei Feldern diskutieren wir nicht in dieser allgemeinen Literaturübersicht, sondern diese wird im nächsten Kapitel, welches sich mit den Anwendungsbereichen im Detail beschäftigt, dargestellt.

Trotz der vielfältigen Anwendungsmöglichkeiten für Petrinetz-Synthese, gibt es bisher eher wenig Werkzeugunterstützung. Das bedeutsamste, vielfach in Anwendungen eingesetzte Werkzeug ist Petrify [58]. Es stellt sehr effizient implementierte, auf den Überlegungen aus [60] basierende Methoden für das ursprüngliche Problem der Synthese von elementaren Netzen aus Transitionssystemen zur Verfügung. Es enthält auch einige Heuristiken zur Aufteilung von Beschriftungen. In dem Werkzeug Genet [52] sind Verallgemeinerungen [54, 53] der Verfahren von Petrify implementiert, welche die Synthese von k-beschränkten S/T-Netzen aus Transitionssystemen ermöglichen. Es beinhaltet auch einige Adaptionen der Grundalgorithmen, um den Anforderungen des Process-Mining gerecht zu werden. Das Werkzeug Synet [49] stellt Implementierungen der polynomiellen Verfahren aus [13, 15, 17] zur Synthese beschränkter, auch verteilter, S/T-Netze aus Transitionssystemen zur Verfügung. Zuletzt ist noch das Werkzeug ProM [2] zu nennen, welches eigentlich kein Petrinetzsynthese-Werkzeug, sondern ein Process-Mining-Werkzeug ist. Allerdings enthält es einige der genannten auf Synthesemethoden basierenden Process-Mining-Verfahren [69, 7, 84].
Neben den bisher betrachteten „präzisen“ Syntheseverfahren, gibt es in der Literatur auch noch etliche mehr oder minder heuristische Konstruktionsmethoden, um ein Petrinetz aus einer Verhaltensbeschreibung zu gewinnen. Solchen Verfahren liegt eine ähnliche Problemstellung wie den eigentlichen Syntheseverfahren zu Grunde. Sie stellen somit ein verwandtes Themengebiet dar und sollen hier nicht unerwähnt bleiben. Insbesondere Faltungsverfahren, welche versuchen, das Prinzip der Netz-Entfaltung umzukehren, weisen Ähnlichkeiten zu Synthesealgorithmen auf. Gerade im Bereich des schon mehrfach erwähnten Process-Mining gibt es eine Vielzahl unterschiedlicher Petrinetz-Konstruktionsverfahren [3, 4]. Ein interessantes Beispiel für einen typischen Faltungsansatz ist in der Arbeit [83] vorgestellt. Ausgehend von einer partiellen Sprache wird hier ein Petrinetz konstruiert, indem die gleich beschrifteten Ereignisse der einzelnen BPOs geeignet verschmolzen werden. Die Arbeit [128] skizziert einen weiteren interessanten Faltungsansatz, welcher verschiedenste Arten von Verhaltensbeschreibungen einschließlich BPOs berücksichtigt. Eine grundlegende Arbeit, welche sich mit der Faltung von Prozessnetzen befasst, ist [210]. Derartige im Gegensatz zu Petrinetz-Syntheseverfahren einfachere Konstruktionsmethoden spielen in dieser Arbeit allerdings keine weitere Rolle und stellen ein eigenes Forschungsfeld dar.
Im Gegensatz zu solchen einfacheren Vorgehensweisen basieren alle präzisen Syntheseverfahren auf der sog. Regionentheorie. Sie folgen damit grob gesagt alle denselben grundsätzlichen Ideen. Regionen wurden in den wegweisenden Arbeiten [89, 90] als Mengen von Zuständen eines Transitionssystems, welche gewisse Eigenschaften erfüllen, eingeführt. In den weiteren Arbeiten zur Synthese von Petrinetzen wurde das Konzept hinter dem Begriff der Region wiederverwendet. Auf diese Weise hat sich eine Vielzahl von Regionendefinitionen in den obig aufgezählten, entsprechend abgewandelten Ausgangssituationen entwickelt. So gibt es inzwischen Regionen für verschiedenste Arten von Verhaltensbeschreibungen und Petrinetzklassen. Die grundlegende Idee ist dabei immer dieselbe: Aus einer Verhaltensbeschreibung lassen sich unmittelbar die Transitionen eines entsprechenden Petrinetzes ablesen. Das Verhalten des zu synthetisierenden Petrinetzes lässt sich

dann durch das Hinzufügen von Stellen einschränken. Dabei gibt es Stellen, welche das Verhalten so stark einschränken, dass das spezifizierte Verhalten verhindert wird. Solche Stellen sollen natürlich nicht zu dem Netz hinzugefügt werden. Von Interesse sind also nur diejenigen Stellen, welche das spezifizierte Verhalten zulassen. Eine Regionendefinition soll nun gerade diese Stellen charakterisieren. Hierzu soll die Struktur der gegebenen Verhaltensspezifikation genutzt werden, so dass sich Regionen dann auch effektiv berechnen lassen.
Mithilfe einer geeigneten Regionendefinition lässt sich ein Syntheseproblem folgendermaßen algorithmisch lösen. Es wird versucht ein Netz zu konstruieren, welches das spezifizierte Verhalten aufweist. Die Transitionen des Netzes sind dabei durch die Verhaltensspezifikation gegeben. Für ein solches Netz kommen nun nur solche Stellen in Frage, welche durch Regionen gegeben sind. Im Prinzip ist die Idee hier, alle Regionen zu berechnen. Werden die zugehörigen Stellen zu dem Netz hinzugefügt, so lässt sich das Verhalten des Netzes anschließend nicht mehr weiter einschränken, ohne das spezifizierte Verhalten zu verhindern. Es gilt also dann, dass das synthetisierte Netz entweder das spezifizierte Verhalten aufweist oder es kein solches Netz gibt. Allerdings gibt es hierbei das Problem, dass in vielen Fällen unendlich viele Regionen existieren. Diese können natürlich nicht effektiv berechnet werden. Somit ergibt sich im Rahmen der Entwicklung von Synthesealgorithmen eine wichtige Herausforderung. Es müssen Berechnungsverfahren entworfen werden, welche eine geeignete endliche Menge an Regionen konstruieren und dennoch das obig beschriebene Vorgehen ermöglichen, d.h. für ein synthetisiertes Netz soll gelten, dass es entweder das Syntheseproblem positiv löst, oder aber das Problem ein negative Antwort besitzt.
Bei einem solchen Syntheseverfahren steht abschließend noch eine letzte Aufgabe aus. Es muss implizit oder explizit überprüft werden, ob das synthetisierte Netz nun das spezifizierte Verhalten aufweist oder nicht. In letzterem Fall hat das Syntheseproblem entsprechend obigen Überlegungen dann eine negative Antwort.
Wie die letzten drei Absätze zeigen, besteht ein Syntheseverfahren aus drei wesentlichen Bestandteilen. Es muss eine geeignete Regionendefinition gefunden werden, ein entsprechendes Berechnungsverfahren entwickelt werden und schließlich eine Überprüfung auf Übereinstimmung des synthetisierten Netzes mit der Verhaltensspezifikation stattfinden. Die vielen in diesem Unterabschnitt genannten Arbeiten zur Synthese von Petrinetzen füllen diese Aspekte auf unterschiedliche Arten aus.
An dieser Stelle kommen wir nun zu der für diese Arbeit relevanten Problemstellung der Synthese eines Petrinetzes aus einer partiellen Sprache zurück. Im Gegensatz zu anderen Syntheseproblemen wird hier halbgeordnetes Verhalten von Petrinetzen betrachtet. Diese, wie die Motivationen in den letzten Unterabschnitten gezeigt haben, sehr interessante Synthese-Fragestellung wurde ursprünglich in [132] aufgeworfen. Ausgehend von den Überlegungen aus [132] wurde im zweiten Teil der Habilitationsschrift [153] dann ein erster Lösungsansatz für das Problem erarbeitet. An weiteren Arbeiten zu diesem Thema sind nur noch die eher theoretischen Ausführungen zu k-beschränkten Netzen in [180] zu nennen.
Der Lösungsalgorithmus aus [153] beschäftigt sich mit dem Problem der Synthese eines S/T-Netzes aus einer endlichen partiellen Spra-

che. Die Arbeit verwendet die auf dem Konzept der Markenflüsse aus [131, 133] basierende Regionendefinition aus [132]. Interessant dabei ist, dass diese Definition sich wesentlich von den bisherigen Regionenbegriffen unterscheidet. So werden Regionen ansonsten zumeist als Mengen oder Multimengen von Zuständen von Transitionssystemen, als Vektoren zur Repräsentation von Kantengewichten bzw. Anfangsmarkierungen von Stellen oder als Kombinationen dieser Konzepte definiert. Weiter wird in [153] ein Ansatz zur Berechnung entsprechender Regionen vorgeschlagen. Dieser ist zwar aus theoretische Sicht zur Lösung des Problems geeignet, aus praktischer Sicht aufgrund einer schlechten Komplexität jedoch so kaum anwendbar. Ein Verfahren zur Überprüfung, ob ein entsprechend synthetisiertes Netz das Syntheseproblem schließlich löst, wird in der Arbeit kurz skizziert, ist allerdings in der dargestellten Form nicht korrekt. Insbesondere ist daher die Entscheidbarkeit des Syntheseproblems noch nicht vollständig bewiesen. Zusammenfassend lässt sich also sagen, dass in [153] zwar schon wesentliche Aspekte zur Lösung des betrachteten Syntheseproblems behandelt sind, allerdings noch kein vollständiger und insbesondere kein für praktische Belange zufriedenstellender Synthesealgorithmus vorliegt. Die Vorarbeiten aus [153] bilden nun die Grundlage für die hier vorliegende Arbeit.

1.4 ZIELSETZUNG

Das Ziel dieser Arbeit ist es, das Problem der Synthese eines Petrinetzes aus einer partiellen Sprache umfassend zu untersuchen und zu lösen. Wir beschränken uns dabei zunächst auf den in [153] ausschließlich betrachteten, eingeschränkten Fall des klassischen Syntheseproblems für Stellen/Transition-Netze und endliche partielle Sprachen.
Die Entwicklung von Lösungsverfahren für dieses Problem wird in dieser Arbeit systematisch diskutiert. Hierzu untersuchen wir jeweils detailliert die drei im letzten Unterabschnitt identifizierten Bestandteile eines Syntheseverfahrens, nämlich Regionendefinitionen, Berechnungsverfahren und Übereinstimmungstest-Algorithmen. Wir gehen hier entsprechend eines Art Baukastenprinzips vor. Das bedeutet, dass wir jedes der drei Teilprobleme im Rahmen des Entwurfes eines Syntheseverfahrens als einzelnen Baustein isoliert betrachten. Dabei entwickeln und diskutieren wir verschiedene Herangehensweisen zur Lösung der einzelnen Teilprobleme. Wir versuchen insbesondere für jedes Teilproblem bzw. jeden Baustein, viele interessante Syntheseideen aus der Literatur in unserem Kontext anzuwenden. Dieses Vorgehen erweist sich als sehr fruchtbar. So entwickeln wir vier verschiedene Regionendefinitionen für partielle Sprachen, wir stellen drei verschiedene Berechnungsverfahren für derartige Regionen vor und wir erstellen ausgehend von den Berechnungsverfahren zwei verschiedene Übereinstimmungstests, um ein synthetisiertes Netz mit der spezifizierten partiellen Sprache zu vergleichen. Die Idee ist dann, dass die einzelnen Herangehensweisen zur Lösung der Teilprobleme entsprechend der Baustein-Idee beliebig zu einem Synthesealgorithmus für das gesamte Syntheseproblem kombiniert werden können, d.h. eine beliebige der Regionendefinitionen zusammen mit einem beliebigen Berechnungsverfahren und einem Übereinstimmungstest ergibt einen Lösungsalgorithmus für das Gesamtproblem. Insgesamt entwickeln wir mit der dargestellten klaren Strukturierung der Problemstellung somit eine Vielzahl interessanter

Lösungsalgorithmen für die betrachtete Synthesefragestellung. Die drei verschiedenen Bausteine für Lösungsalgorithmen subsumieren wir unter dem Begriff der Lösungsdimension der Synthese von Petrinetzen aus halbgeordneten Abläufen. Das Baukastenprinzip der Lösungsdimension ist in Abbildung 7 links illustriert.

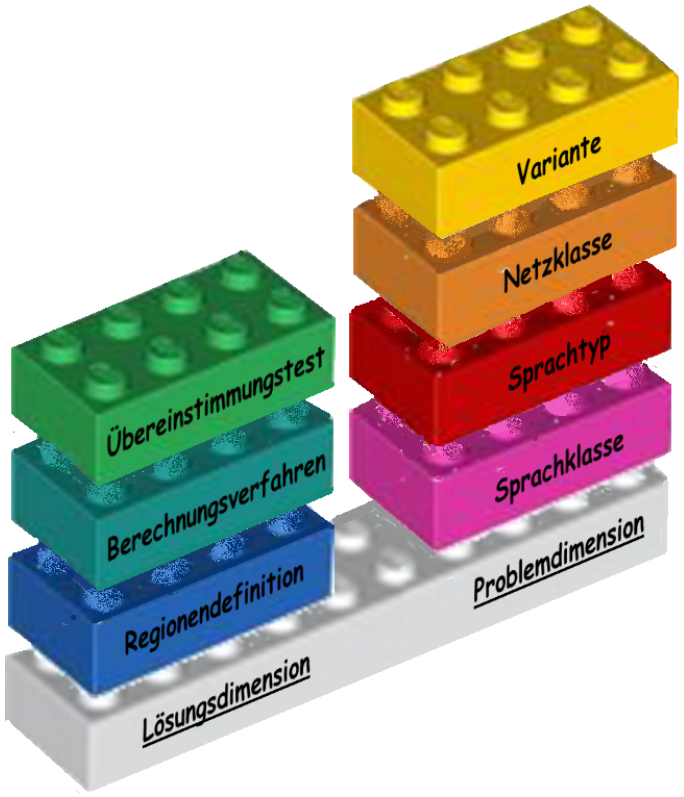

Abbildung 7: Baukasten für die Synthese von Petrinetzen aus halbgeordneten Abläufen.

Die aus dem Baukastenprinzip resultierenden Algorithmen weisen teilweise große konzeptuelle Unterschiede auf. Um die Unterschiede verstehen zu können, werden die verschiedenen Lösungsansätze für die drei Teilprobleme in dieser Arbeit detailliert untersucht und verglichen. Es werden sowohl Komplexitätsvergleiche als auch experimentelle Tests durchgeführt. Es zeigt sich hierbei, dass es insgesamt kein bestes Lösungsverfahren gibt, sondern die Ergebnisse für die verschiedenen Algorithmen von der Struktur der betrachteten partiellen Sprache abhängen. Insbesondere wird bei den empirischen Tests deutlich, dass die entwickelten Verfahren auch für größere praktische Beispiele tauglich sind.
Experimentelle Tests der Synthesealgorithmen erfordern natürlich entsprechende Implementierungen. Wir stellen in diesem Rahmen auch ein Synthese-Werkzeug mit einer benutzerfreundlichen graphischen Oberfläche zur komfortablen Anwendung der implementierten Verfahren zur Verfügung.
Das nächste entscheidende Ziel der Arbeit ist dann die Aufhebung der bisher betrachteten Einschränkungen der Syntheseproblemstellung. Auch hierbei gehen wir systematisch vor. Wir teilen die Synthesepro-

blemstellung in vier einzelne Bestandteile auf. Wir setzen also das Baukastenprinzip der Lösungsdimension der Synthese im Rahmen der Problemstellung fort. Die entsprechende Unterteilung der Problemstellung in Bausteine bezeichnen wir als Problemdimension. Sie ist in Abbildung 7 rechts dargestellt. Die Idee ist nun, dass wir die im Rahmen der Lösungsdimension entwickelten Syntheseverfahren derart anpassen können, dass wir sie auch auf Syntheseprobleme, welche entsprechend der einzelnen Bausteine der Problemdimension verändert bzw. verallgemeinert sind, anwenden können. Hierbei fließen wiederum Ideen aus verschiedenen Arbeiten zur Synthese in der Literatur ein.
Der erste Baustein der Problemdimension ist die betrachtete Sprachklasse. Bisher haben wir endliche partielle Sprachen vorausgesetzt. Die entwickelten Synthesealgorithmen sollen aber auch auf unendliche partielle Sprachen erweitert werden. Im Falle unendlicher Sprachen besteht natürlich immer das Problem, dass sie für algorithmische Zwecke endlich repräsentiert werden müssen. Wir betrachten in diesem Kontext Synthesealgorithmen für verschiedene Arten von termbasierten endlichen Repräsentationen unendlicher Sprachen. Natürlich deckt jede Art einer endlichen Repräsentation dabei nur eine bestimmte Klasse von unendlichen partiellen Sprachen ab.
Der zweite Problem-Baustein ist der Sprachtyp der partiellen Sprache. Diese Arbeit soll sich ja generell mit der Synthese von Petrinetzen unter Berücksichtigung von halbgeordnetem Verhalten befassen. Allerdings gibt es neben BPOs auch allgemeinere Darstellungsmöglichkeiten für halbgeordnete Abläufe. Wir zeigen in diesem Kontext, wie sich die Synthesealgorithmen auf sog. geschichtete Ordnungen erweitern lassen. Daneben werden kurz auch noch verschiedene Semantiken von BPOs im Rahmen der Synthese diskutiert.
Weiterhin sollen nicht mehr nur S/T-Netze betrachtet werden, sondern die Synthesealgorithmen sollen auf verschiedenste Petrinetzklassen verallgemeinert werden. Die Netzklasse ist also der dritte Baustein der Problemdimension. Wir betrachten in diesem Rahmen eine parametrische Petrinetzdefinition, welche alle typischen Netzklassen, wie elementare Netze, S/T-Netze und Erweiterungen dieser Netzklassen, beispielsweise um Inhibitorkanten, Lesekanten oder Kapazitäten, umfasst. Die Syntheseideen werden auf diese parametrische Petrinetzdefinition erweitert. In dem Bereich der Petrinetztheorie gibt es etliche Beispiele, in denen zu einem bereits gut erforschten Thema viele weitere Artikel geschrieben werden, um weitere Petrinetzklassen zu berücksichtigen. Dieses Problem tritt bei dem hier gewählten parametrischen Ansatz nicht auf, da es zur Betrachtung einer weiteren Petrinetzklasse nur nötig ist, die Parameter geeignet zu wählen.
In der umfassenden Diskussion dieser Arbeit muss schließlich auch berücksichtigt werden, dass sich anstelle des bisher betrachteten klassischen Syntheseproblems, welches die exakte Reproduktion einer Sprache durch ein beliebiges Netz fordert, auch verschiedenste weitere Synthesefragestellungen formulieren lassen. Beispiele solcher Varianten der klassischen Fragestellung sind die Frage nach einem Petrinetz dessen Verhalten um einen bestimmten Spielraum von dem spezifizierten Verhalten abweichen darf, die Frage nach einem Petrinetz, dessen Verhalten in einem gewissen Sinne eine beste Approximation an das spezifizierte Verhalten darstellt oder die Frage nach Syntheselösungen mit möglichst wenig Stellen. Derartige Varianten bzw. Abwandlungen

der Synthesefragestellung stellen also den vierten Baustein der Problemdimension dar. In dieser Arbeit werden verschiedenste solcher Fragestellungen untersucht. Es wird gezeigt, dass die entwickelte Synthesesystematik auch in diesem Rahmen wieder anwendbar ist. Jede betrachtete Fragestellung lässt sich durch Modifikationen der Synthesealgorithmen für das klassische Syntheseproblem lösen.
Insgesamt zeigt sich, dass zur Lösung der verschiedenen, durch die Bausteine der Problemdimension gegebenen Syntheseprobleme jeweils wieder die Betrachtung der drei Bausteine der Lösungsdimension notwendig ist. Umfassend wird die Lösungsdimension in dieser Arbeit, wie dargestellt, für den Fall der exakten Synthese eines S/T-Netzes aus einer endlichen partiellen Sprache diskutiert. Die in diesem Fall entwickelten grundsätzlichen Synthesestrategien und -ideen lassen sich für die meisten abgewandelten Problemstellungen wiederverwenden. Das bedeutet, dass sich Synthesealgorithmen für entsprechend komplexere Syntheseprobleme normalerweise mit geeigneten Erweiterungen und Zusatzüberlegungen direkt aus den Lösungsalgorithmen für unsere Standard-Problemstellung ergeben. Somit lassen sich die Ausführungen dieser Arbeit zu den einzelnen Bausteinen sowohl der Problemdimension als auch der Lösungsdimension mehr oder weniger beliebig miteinander kombinieren. Folglich ergibt sich, wie in Abbildung 7 dargestellt, tatsächlich ein zusammenhängender Baukasten für die Synthese von Petrinetzen aus halbgeordneten Abläufen, welcher die Problem- und die Lösungsdimension umfasst. Die Ausführungen dieser Arbeit zeigen insbesondere, dass die entwickelte Systematik in einem sehr allgemeinen Rahmen anwendbar ist.
Neben den gerade erläuterten umfassenden theoretischen Ausführungen zum Problem der Synthese von Petrinetzen aus halbgeordneten Abläufen im Kontext des dargestellten Baukastenprinzips wird in dieser Arbeit auch die praktische Anwendbarkeit entsprechender Synthesealgorithmen diskutiert und typische Anwendungsszenarien skizziert. Wir erläutern detailliert die Anwendungsperspektive der Synthesealgorithmen im Rahmen des in den letzten Unterabschnitten schon kurz dargestellten und motivierten streng ablauforientierten Modellierungsansatzes. Bei diesem Ansatz werden zunächst Szenarien eines zu modellierenden Systems gesammelt und formalisiert. Aus den formalen Szenario-Modellen wird dann mithilfe von Syntheseverfahren ein Petrinetzmodell des Systems erzeugt. Wir stellen in dieser Arbeit insbesondere mehrere konkrete Anwendungsbereiche für einen solchen Ansatz dar und illustrieren diese mit etlichen Modellierungs-Beispielen.
An dieser Stelle nehmen wir nun abschließend noch einmal Bezug auf die wesentliche Vorarbeit [153]. In dieser Arbeit wird das Problem der exakten Synthese eines Stellen/Transition-Netzes aus einer endlichen partiellen Sprache betrachtet. Es wird, wie schon erläutert, eine Regionendefinition sowie ein zugehöriges Berechnungsverfahren für dieses Problem entwickelt und ein Übereinstimmungstest-Algorithmus skizziert. Im Gegensatz dazu werden in der vorliegenden Arbeit eine Vielzahl von Synthesealgorithmen im Rahmen einer systematischen Vorgehensweise dargestellt. Die in [153] entwickelten Lösungsansätze sind dabei mögliche Ausprägungen der entsprechenden Bausteine unseres Baukastens. Die Ansätze aus [153] werden im Kontext der Ausführungen zu den entsprechenden Bausteinen aufgegriffen und in unsere Systematik eingebettet. Im Gegensatz zu [153] präsentieren wir auch Implementierungen der Synthesealgorithmen. Außerdem ver-

gleichen und analysieren wir die Verfahren hier im Detail. Dabei zeigt sich auch, dass die Berechnungsmethode zur Erzeugung von Netzen in [153] praktisch kaum tauglich ist. Andere der hier entwickelten Algorithmen weisen dahingegen eine wesentlich bessere, auch für praktische Zwecke ausreichende Performance auf. Den Aspekt der praktischen Anwendbarkeit betreffend sei auch darauf hingewiesen, dass in dieser Arbeit im Gegensatz zu [153] mögliche Anwendungsfelder für die entwickelten Syntheseverfahren ausführlich dargelegt werden. Schließlich ist ein ganz entscheidender Punkt, dass wir uns in der vorliegenden Arbeit nicht auf die sehr spezielle Synthesefragestellung aus [153] beschränken, sondern im Rahmen des Synthesebaukastens systematisch auch viele andere Problemstellungen aus dem Bereich der Synthese von Petrinetzen aus halbgeordneten Abläufen untersuchen und lösen. So heben wir insbesondere die wesentliche Einschränkung auf Spezifikationen für endliches Verhalten auf (Ideen hierzu sind auch schon in [153] dargelegt), wir betrachten auch Verallgemeinerungen von BPOs, wir beschränken uns nicht mehr nur auf die Synthese von S/T-Netzen und wir diskutieren Abwandlungen der klassischen Synthesefragestellung.

1.5 GLIEDERUNG

Um die umfassenden theoretischen Untersuchungen des Problems der Synthese von Petrinetzen aus halbgeordneten Abläufen zu motivieren und die Bedeutung dieses Themas deutlich zu machen, beginnen wir die Arbeit in Kapitel 2 mit der Darstellung von Anwendungsfeldern für entsprechende Synthesealgorithmen. Wir erläutern den Ansatz der streng ablauforientierten Modellierung und stellen konkrete Anwendungen vor. In Kapitel 3 klären wir dann die für die folgenden theoretischen Überlegungen notwendigen formalen Grundlagen. In den nächsten zwei Kapiteln entwickeln wir im Rahmen des Baukasten-Prinzips aus Abbildung 7 verschiedenste Syntheseverfahren. Zuerst betrachten wir in Kapitel 4 detailliert die einzelnen Bausteine der Lösungsdimension des Baukastens. In Kapitel 5 diskutieren wir anschließend die Problemdimension des Baukastens. In Kapitel 6 runden wir die Arbeit schließlich mit einer kurzen Zusammenfassung und einem Ausblick auf interessante weiterführende Forschungsthemen ab.
Wir geben nun noch einen kurzen Überblick über die Inhalte der einzelnen Kapitel der Arbeit. Es sei hier angemerkt, dass Teile der in dieser Dissertation vorgelegten Ergebnisse bereits in verschiedenen Publikationen in Journalen und Tagungsbänden veröffentlicht wurden. Im Rahmen der folgenden Kapitelzusammenfassungen weisen wir jeweils auf entsprechende Publikationen hin.

- **Kapitel 1 – Einleitung**: Der Inhalt der Einleitung soll an dieser Stelle nicht noch einmal zusammengefasst werden. Das im Rahmen der Einleitung vorgestellte Baukasten-Prinzip basiert auf Überlegungen der Vorarbeit [159].
- **Kapitel 2 – Streng ablauforientierter Modellierungsansatz**: In diesem Kapitel wird ein Modellierungsansatz, welcher auf Syntheseverfahren basiert, vorgestellt. Wesentliche Ideen dieses Ansatzes finden sich in der Publikation [31]. In Abschnitt 2.1 wird dargestellt, wie sich dieser Ansatz im Rahmen der Entwicklung von Software und Algorithmen einsetzen lässt. Abschnitt 2.2 diskutiert den Modellierungsansatz dann im Kontext der Modellierung

von Geschäftsprozessen, wobei insbesondere auch auf das Thema des Process-Mining eingegangen wird. Relevante Publikationen zu diesem Abschnitt sind [31, 26]. Schließlich wird in Abschnitt 2.3 die Anwendung des streng ablauforientierter Modellierungsansatz zur Erstellung von Lernprozessmodellen angesprochen. In die Ausführungen dieses Abschnittes fließt die Publikation [23] ein.

- **Kapitel 3 – Formale Grundlagen**: Zunächst werden in Abschnitt 3.1 einige Schreibweisen geklärt. Anschließend führen wir in Abschnitt 3.2 die für diese Arbeit notwendigen Grundlagen zu Petrinetzen ein. In Abschnitt 3.3 werden dann die notwendigen Konzepte im Bereich halbgeordneter Abläufe formal vorgestellt. Abschließend werden in Abschnitt 3.4 einige Begriffe und Verfahren im Bereich von linearen Ungleichungssystemen, welche im Kontext der Synthesealgorithmen eine entscheidende Rolle spielen, diskutiert.

- **Kapitel 4 – Verfahren zur Synthese von S/T-Netzen aus endlichen partiellen Sprachen**: In diesem Kapitel wird die Lösungsdimension des Baukastens aus Abbildung 7 für die spezielle Problemstellung der Synthese eines S/T-Netzes aus einer endlichen partiellen Sprachen behandelt. In Abschnitt 4.1 wird diese Problemstellung zunächst analysiert und diskutiert. Anschließend wird in Abschnitt 4.2 das prinzipielle Vorgehen zur Lösung des Problems kurz skizziert. In Abschnitt 4.3 werden dann konkret vier verschiedene Regionendefinitionen eingeführt. Abschnitt 4.4 stellt drei zugehörige Berechnungsverfahren für Regionen vor. Schließlich werden in Abschnitt 4.5 zwei Übereinstimmungstest-Verfahren entwickelt. Aus den Ausführungen der letzten drei Abschnitte ergeben sich dann entsprechende Synthesealgorithmen. Relevante Publikationen zu diesen Abschnitten des Kapitels sind [27, 155, 30, 159]. In Abschnitt 4.6 präsentieren wir Implementierungen der in den vorherigen Abschnitten entwickelten Synthesealgorithmen. Diesem Abschnitt liegt die Publikation [29] zugrunde. Als Abschluss des Kapitels vergleichen wir in Abschnitt 4.7 die Synthesealgorithmen sowohl im Rahmen von Komplexitätsbetrachtungen als auch von experimentellen Ergebnissen. Diese Ausführungen basieren auf der Publikation [30].

- **Kapitel 5 – Verallgemeinerung der Synthesefragestellung**: In diesem Kapitel wird nun die Problemdimension des Baukastens aus Abbildung 7 diskutiert. Zunächst verallgemeinern wir die bisher betrachtete Syntheseproblemstellung hinsichtlich der Sprachklasse. Wir lösen das Synthese-Problem in Abschnitt 5.1 für verschiedene Klassen von unendlichen partiellen Sprachen. Diesem Abschnitt liegen die Publikationen [32, 28] zugrunde. In Abschnitt 5.2 diskutieren wir dann alternative Sprachtypen. Zunächst wird wie in der Publikation [158] die Synthese aus sog. geschichteten Sprachen betrachtet. Anschließend werden noch alternative Semantiken für partielle Sprachen untersucht. In Abschnitt 5.3 wird daraufhin das Syntheseproblem im Rahmen einer parametrischen Petrinetzdefinition, durch welche sich eine Vielzahl von Netzklassen abbilden lässt, diskutiert. Eine zugehörige Publikation ist [157]. Zuletzt werden in Abschnitt 5.4 Varianten

der klassischen Synthesefragestellung besprochen. Zu diesem Abschnitt ist die Publikation [164] zu nennen.

- **Kapitel 6 – Abschlussbetrachtungen**: In Abschnitt 6.1 wird zuerst noch einmal auf einige wichtige Aspekte dieser Arbeit zurück geblickt, ehe die Dissertation in Abschnitt 6.2 mit einem Ausblick auf interessante weiterführende Themen abgeschlossen wird.

2

STRENG ABLAUFORIENTIERTER MODELLIERUNGSANSATZ

In diesem Kapitel werden Anwendungen zum Thema der Synthese von Petrinetzen aus halbgeordneten Abläufen diskutiert. Zumeist werden mögliche Anwendungen erst nach den theoretischen Ausführungen zu einem Thema detailliert erläutert. In dieser Arbeit ist das Anwendungskapitel allerdings vorgezogen, um das zu untersuchende Thema noch einmal umfassend zu motivieren und dessen Bedeutung und Vielseitigkeit hervorzuheben. Dennoch darf dieses Kapitel keinesfalls als reines Motivationskapitel betrachtet werden, sondern es werden ausführliche wissenschaftliche Überlegungen zu Anwendungsfeldern der Synthese von Petrinetzen aus halbgeordneten Abläufen präsentiert. Der Anwendungskontext von Petrinetzsynthese wurde in der Einleitung schon kurz dargestellt. Die Grundidee besteht darin Syntheseverfahren im Rahmen einer systematischen Vorgehensweisen zur Modellierung von Systemen, dem sog. streng ablauforientierten Modellierungsansatz, zu verwenden. Die Ausgangslage ist also, dass ein System beispielsweise zum Zwecke der Analyse, der Optimierung oder der Implementierung modelliert werden soll.

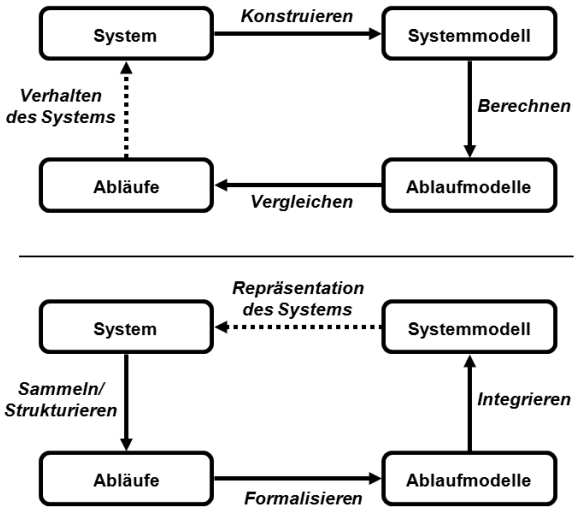

Abbildung 8: Klassischer (oben) und streng ablauforientierter (unten) Modellierungsansatz (in Anlehnung an [72]).

Den Startpunkt für die folgenden Überlegungen bildet somit ein zu modellierendes reales System. Nun gibt es verschiedene Herangehensweisen, um zu einem geeigneten formalen Modell des Systems zu gelangen. Der klassische Ansatz zur direkten Modellierung von Systemen ist in Abbildung 8 oben prototypisch illustriert. Die Modellierung findet ent-

lang der dargestellten Pfeile statt. Die einzelnen Modellierungsschritte entsprechen mehr oder minder den durchgezogenen Pfeilen des Diagramms. Es wird also direkt mit der Konstruktion eines Modells des Systems begonnen. Zur Überprüfung, ob das konstruierte Modell valide ist, werden anschließend zugehörige Ablaufmodelle berechnet. Für diese wird untersucht, ob sie den Abläufen des tatsächlichen Systems entsprechen.
Der klassische Modellierungsansatz sollte von einem Modellierungsexperten durchgeführt werden. In dem ersten Schritt, dem zentralen Konstruktionsschritt, erstellt der Modellierer aus seinen expliziten und impliziten Informationen und Vorstellungen über das System unmittelbar ein Modell des Systems. Da die wesentlichen Informationen über ein System typischerweise Informationen über dessen Verhalten oder genauer gesagt dessen Abläufe sind, bedeutet dies, dass der Modellierer versucht ein Systemmodell zu konstruieren, welches die relevanten Abläufe unterstützt. Bei genauerer Betrachtung muss der Modellierer bei diesem Konstruktionsschritt drei schwierige Aufgaben auf einmal durchführen.

- Zuerst einmal muss er sich einen Überblick über seine Informationen und Vorstellungen zu den Abläufen des Systems verschaffen, d.h. er muss alle entsprechenden Informationen gedanklich sammeln und strukturieren.
- Weiter genügt es nicht, die einzelnen Abläufe zu verstehen, sondern er muss das System als Ganzes überblicken. Hierzu muss er aus den verschiedenen Einzelinformationen über das System gedanklich ein Gesamtsystem zusammenfügen, d.h. er muss die einzelnen Ablaufinformationen geeignet integrieren.
- Schließlich muss der Modellierer einen Abstraktionsschritt durchführen, der es ihm ermöglicht, aus seiner Vorstellung des realen Systems eine geeignete formale Modell-Repräsentation des Systems zu entwickeln.

Somit wird bei diesem Vorgehen ein Modell eines Systems erstellt, indem die vorhandenen Informationen zugleich gesammelt/strukturiert, integriert und formalisiert werden. Da selbst ein Modellierungsexperte mit dieser Dreifachbelastung an schwierigen Aufgaben häufig überfordert ist, weist ein auf diese Weise entwickeltes Modell in vielen Fällen nicht exakt das gewünschte Systemverhalten auf. Daher ist bei einem solchen Vorgehen meist anschließend ein Validierungsschritt vorgesehen, in dem ex-post überprüft wird, ob das Modell das reale System getreu widerspiegelt. Hierzu wird normalerweise das Verhalten des Modells mit dem gewünschten Verhalten des betrachteten Systems verglichen. Zu diesem Zwecke bietet sich die Betrachtung der Abläufe des Modells an. Mit geeigneten Simulations-Werkzeugen ist es möglich, aus dem formalen Systemmodell formale Modelle von zugehörigen Abläufen zu berechnen. Der Modellierungsexperte kann dann prüfen, inwiefern die Abläufe des Modells tatsächlich den gewünschten Abläufen des betrachteten Systems entsprechen. Ergeben sich hier Unterschiede, so versucht er, das Modell des Systems entsprechend anzupassen. Iterativ werden anschließend wiederum die Abläufe des veränderten Modells des Systems untersucht bis sich ein zufriedenstellendes Modell ergibt.
Die zentrale Idee des streng ablauforientierten Modellierungsansatz ist eine Aufgliederung und Explizierung der drei identifizierten Aufgaben

des gerade beschriebenen Konstruktionsschrittes des klassischen Modellierungsansatzes. Wie in Abbildung 8 unten dargestellt, wird bei diesem Vorgehen mit dem Sammeln/Strukturieren der Ablaufinformationen des Systems begonnen. Dann werden die Ablaufinformationen formal repräsentiert. Schließlich werden sie zu einem Modell des ursprünglichen Systems integriert, wobei die an dieser Stelle erreichte formale Ebene die Anwendung automatischer Integrationsverfahren erlaubt. Wie in Abbildung 8 illustriert, kehrt dieses Vorgehen den klassischen Modellierungsansatz gerade um. Der wesentliche Vorteil des streng ablauforientierten Modellierungsansatzes besteht darin, dass der komplexe Konstruktionsschritt des klassischen Ansatzes in drei klar abgegrenzte Schritte, welche gerade den drei Haupt-Modellierungsaufgaben entsprechen, zerlegt wird. Dadurch werden die drei ohnehin schwierigen Aufgabenstellungen einerseits nicht durch die Probleme der jeweils anderen Aufgaben noch zusätzlich belastet und andererseits erfahren sie eine systematische Strukturierung. Inbesondere ergibt sich die Möglichkeit, den Integrationsschritt automatisiert durchzuführen.
Bei dem streng ablauforientierten Modellierungsansatz steht von Beginn an das Verhalten des Systems im Vordergrund. Zuallererst sollen die Abläufe des zu modellierenden Systems genau untersucht werden. Es wird explizit eine valide, typischerweise anfangs noch informale oder semi-formale Szenario-Spezifikation für das Verhalten des Systems entwickelt. Das Ziel dieses ersten Modellierungsschrittes ist also die Gewinnung und Strukturierung exakter Beschreibungen der Abläufe des Systems. Solche Beschreibungen können insbesondere von fachkundigen Experten, welche gerade den betrachteten Ablauf sehr genau kennen, erstellt werden. Exakte Ablaufbeschreibungen können aber auch auf andere Weise entwickelt werden, z.B. durch Beobachtung des Systems oder durch automatische Aufzeichnung des Systemverhaltens in Log-Dateien. Entscheidend ist nun, dass eine anfänglich informale Szenario-Spezifikation in dem nächsten Modellierungsschritt soweit formalisiert wird, dass die Abläufe des Systems präzise und unmissverständlich beschrieben sind. Die einzelnen Ablaufbeschreibungen müssen also in entsprechende formale Ablaufmodelle übersetzt werden. Formale Ablaufmodelle werden somit im Gegensatz zu dem klassischen Ansatz nicht durch Simulation aus einem Systemmodell gewonnen, sondern sie werden mithilfe geeigneter Informationen über das System noch vor der Erstellung des eigentlichen Systemmodells hergeleitet. Ausgehend von einer formalen Ablauf-Spezifikation folgt nun der zentrale Schritt des Modellierungsansatzes. Die formalen Modelle der einzelnen Abläufe sollen zu einem integrierten Modell zusammengefügt werden, dessen Verhalten gerade durch die betrachteten Abläufe gegeben ist. Da wir an dieser Stelle formale Ablaufmodelle betrachten, liegt es nahe, algorithmische Verfahren zur Integration der Abläufe zu verwenden. Es soll dementsprechend automatisch ein Systemmodell generiert werden, welches per Konstruktion der Ablauf-Spezifikation des Systems genügt. Ein solches Modell stellt im Rahmen der vorhandenen Ablaufinformationen dann eine bestmögliche Repräsentation das betrachteten Systems dar (eine weitere Validierung des Modells ist daher im Idealfall nicht mehr notwendig). Eine solche automatisierte Konstruktion eines Modells erfordert allerdings nichttriviale algorithmische Überlegungen. An dieser Stelle kommen nun entsprechende Syntheseverfahren ins Spiel. Es stellt sich das Problem, ausgehend von einer formalen Spezifikation der Abläufe eines Systems, automatisch ein

integriertes Modell des Systems zu erzeugen. Somit wirft der zentrale Modellierungsschritt des streng ablauforientierten Modellierungsansatzes gerade das Syntheseproblem auf, welches wir in den folgenden Kapiteln dieser Arbeit im Kontext von Petrinetzen behandeln.
Im Folgenden erläutern wir detailliert den Hintergrund und die Vorteile des streng ablauforientierten Modellierungsansatzes. Generell liegt Wissen über ein System typischerweise in einer informalen und unstrukturierten Form vor. Häufig verteilen sich die Informationen über das System zudem auf verschiedenste Personen oder Dokumente. In einem solch komplexen Umfeld stellt sich nun der klassische Schritt der verzahnten Sammlung/Strukturierung, Integration und Formalisierung der Informationen hin zu einem validen Systemmodell als sehr schwierig und fehleranfällig dar. Der Schlüsselschritt des streng ablauforientierten Modellierungsansatzes, um eine Brücke von einer anfänglich informalen, unstrukturierten Sicht auf ein System hin zu einem formalen, integrierten Modell des Systems zu schlagen, ist nun eine umfassende Untersuchung der Abläufe des Systems. Diese Vorgehensweise leitet sich insbesondere aus der Annahme ab, dass Fach-Experten einzelne Abläufe eines Systems besser kennen und verstehen als das u.U. sehr komplexe Gesamtsystem. Die Spezifikation von einzelnen Abläufen stellt sich als benutzernahe und intuitive Aufgabe dar, welche unabhängig voneinander von verschiedenen Experten für verschiedene Teilbereiche eines Systems durchgeführt werden kann. Eine integrierte Betrachtung des Systems ist normalerweise schwieriger. Dies liegt zum einen daran, dass ein Fachexperte nur partielles Wissen über ein System besitzt und es häufig nicht ohne Weiteres möglich ist, dieses in integrierter Form einem zuständigen Modellierer zu vermitteln. Zum anderen kann ein System auch im Rahmen einer integrierten Betrachtung nur über dessen Abläufe vollständig verstanden werden, so dass zusätzliche Abstraktionsschritte notwendig sind. Manchmal stellt sich sicherlich auch die integrierte Betrachtung eines Teiles eines Systems einschließlich dessen Verzweigungsstruktur als einfachere Aufgabe dar als einzelne Abläufe zu untersuchen, allerdings können in einem solchen Fall entsprechende Abläufe direkt aus dem betrachteten Systemausschnitt abgeleitet werden. Insgesamt ergibt sich als logische Konsequenz dieser Überlegungen, dass als erster Schritt bei dem betrachteten Modellierungsansatz eine Sammlung/Strukturierung der Abläufe des Systems durchgeführt und eine entsprechende explizite Ablauf-Spezifikation erstellt wird.
Hierzu ist anzumerken, dass, wie oben beschrieben, auch bei dem klassischen direkten Modellierungsansatz die Abläufe des Systems eine wichtige Rolle spielen und in irgendeiner Form bei der Konstruktion eines Modells eingehen müssen. Somit stellt die Erstellung einer Ablauf-Spezifikation eine Explizierung und Systematisierung eines Schrittes dar, welcher auch bei dem klassischen Ansatz zumindest implizit vorkommt. Neben dem genannten Vorteil, dass eine Ablauf-Spezifikation einerseits auf intuitive Weise und andererseits in einem verteilten Umfeld erstellt werden kann, ergeben sich folgende weitere Vorteile eines derartigen Fokus auf die Abläufe eines Systems: die Betrachtung und Dokumentation des Systems aus der Sicht von Nutzern, die intuitive Verständlichkeit einer Ablauf-Spezifikation aufgrund des einfachen Abstraktionsniveaus von Abläufen, die Möglichkeit auch partielle Spezifikationen zu schreiben und inkrementell zu erweitern, kurze Rückkoplungs- und Feedback-Zyklen, die Möglichkeit direkt

Testfälle abzuleiten und schließlich spezielle Techniken zur Gewinnung von Ablaufbeschreibungen wie die Aufzeichnung von Log-Dateien.
Aus den genannten Gründen hat sich das Vorgehen, eine Ablauf-Spezifikation zu erstellen, mit verschiedensten Schwerpunkten und Variationen auch in etlichen Bereichen des Softwareengineering durchgesetzt. Der nächste kreative Schritt ist in diesem Kontext dann aber üblicherweise die Integration der meist informal spezifizierten Abläufe in ein formales Systemmodell, dessen Abläufe den spezifizierten Abläufen in einem vagen Sinn entsprechen. Somit ist bei diesem Vorgehen im Gegensatz zu dem ganz klassischen Modellierungsansatz zwar der Schritt des Sammelns/Strukturierens der Ablaufinformationen explizit vorangestellt, dennoch stellt sich nach wie vor die schwierige Aufgabe, die Ablaufinformationen in einem gemeinsamen Schritt gleichzeitig gedanklich zu integrieren und formalisiert festzuhalten. Nach der Erstellung des Modells erfolgt typischerweise analog zum klassischen Ansatz eine Validierung des Systemmodells, wobei hierzu nun die zuvor spezifizierten Abläufe zu Rate gezogen werden können. Dieser im Softwareengineering weit verbreitete sog. ablauforientierte, jedoch nicht streng ablauforientierte, Modellierungsansatz ist in Abbildung 9 dargestellt. Er stellt eine bedeutsame und logische Fortentwicklung des klassischen Modellierungsansatzes dar.

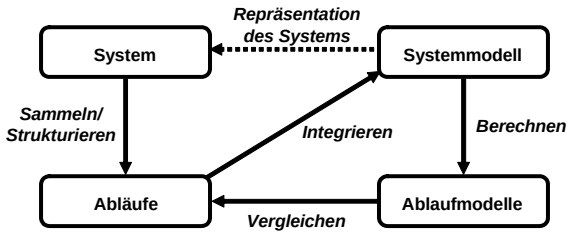

Abbildung 9: Verbreiteter ablauforientierter Modellierungsansatz.

Der hier vorgestellte streng ablauforientierten Ansatz ist eine konsequente Weiterentwicklung des ablauforientierten Modellierungsansatzes. Bei letzterem ist es nicht notwendig, die spezifizierten Abläufe in einer formalisierten Form zu betrachten. Auch wenn bei diesem Ansatz manchmal eine formale Syntax verwendet wird, so erlauben entsprechende Modellierungssprachen normalerweise einen gewissen Interpretationsspielraum. Meist wird aber ohnehin natürliche Sprache oder eine semi-formale Notation verwendet. Während dieses informale Vorgehen Flexibilität mit sich bringt, so kann es auch sehr leicht zu Missverständnissen führen. Der für die Systemmodellierung zuständige Modellierer weiß in vielen Fällen nicht genau, wie bestimmte Aspekte einer informalen Ablauf-Spezifikation zu verstehen sind. Somit besteht im Rahmen der Konstruktion eines integrierten Systemmodells die zusätzliche Herausforderung, dass den informalen Informationen schrittweise Semantik zugewiesen werden muss. Der Übergang auf eine formale Ebene findet also im Zuge der Übersetzung der Spezifikation in ein Systemmodell fließend statt. Bei einem solchen Vorgehen kann kaum gewährleistet werden, dass das erstellte Modell der ursprünglichen Intuition wirklich entspricht.
Daher wird bei dem streng ablauforientierten Ansatz eine direkte Übersetzung der Abläufe in eine formale Form durchgeführt. Wir glauben, dass ein Formalisierungsschritt auf der intuitiven Ebene der einzelnen

Abläufe sehr einfach und klar ist und auch von den jeweiligen Fachexperten vollzogen werden kann. Die betreffenden formalen Objekte sind so einfach gehalten wie möglich. Es geht hier im Prinzip ausschließlich um eine Übersetzung in eine formale Form, ohne dass etwaige kreative Überlegungen notwendig sind. Somit bleibt bei dem ansonsten häufig problematischen Übergang von der informalen auf die formale Ebene nun wenig Raum für Missverständnisse. Voraussetzung hierzu ist allerdings eine ausreichende Präzision der zu übersetzenden Ablauf-Spezifikation. Diese zu gewährleisten, ist in diesem Kontext eine sehr wichtige aber auch schwierige Aufgabe. Da eine entsprechende Präzision nicht immer realisiert werden kann, ist häufig eine Validierung der einzelnen formalen Ablaufmodelle sinnvoll und notwendig. Auf der formalen und damit präzisen, Nutzern aber dennoch leicht vermittelbaren Ebene der Ablaufmodelle, können Missverständnisse sehr gut ausgeräumt werden. Dies liegt daran, dass Nutzer mit entsprechender Unterstützung Ablaufmodelle sehr gut verstehen und überblicken können. Insbesondere weisen Ablaufmodelle typischerweise viele Parallelen zu der Denkweise der Nutzer auf.
Mit einer formalen Ablauf-Spezifikation ist dann unmittelbar die streng formale Ebene erreicht. Der Integrationsschritt folgt nun ausgehend von der formalen Ablauf-Spezifikation. Aus den formalen Abläufen soll ein formales Systemmodell, welches ein entsprechendes Ablaufverhalten aufweist, konstruiert werden. Dieser Schlüsselschritt soll und kann weitgehend automatisch durchgeführt werden. Hierzu werden wir in dieser Arbeit entsprechende Syntheseverfahren vorstellen. Solche Verfahren garantieren, dass das erzeugte Systemmodell bestimmte Anforderungen in Bezug auf die formalen Ablaufmodelle erfüllt. Modellierungsfehler bei der Übersetzung der Ablauf-Spezifikation in ein Systemmodell können auf diese Weise vermieden werden.
Allerdings erfordert die Entwicklung von Syntheseverfahren sowohl für die Ablaufebene als auch für die Systemebene die Beschränkung auf bestimmte Formalismen. In den bisherigen Ausführungen dieses Kapitels haben wir keinen Bezug zu speziellen Modellierungssprachen hergestellt. Es wurden die wesentlichen Konzepte des streng ablauforientierten Modellierungsansatz erläutert, jedoch wurde von technischen Feinheiten abgesehen. Wir schlagen an dieser Stelle nun vor, BPOs zur Modellierung von Abläufen und Petrinetze zur Modellierung des Kontrollflusses von Systemen zu betrachten. Diese beiden Formalismen sind sehr allgemeine und streng formale Modellierungskonzepte. Wir haben in der Einleitung schon diskutiert, dass sich BPOs und Petrinetze zu entsprechenden Modellierungszwecken, sobald Nebenläufigkeit eine gewisse Rolle spielt, sehr gut eignen. Nebenläufigkeit ist aber in der Realität bei den meisten komplexeren Systemen von Bedeutung. Dementsprechend werden wir im Weiteren den streng ablauforientierten Modellierungsansatz im Zusammenhang mit BPOs und Petrinetzen verfolgen. Als theoretische Fragestellung im Rahmen des zentralen Konstruktionsschrittes des streng ablauforientierten Modellierungsansatzes stellt sich dann gerade das Problem der Synthese eines Petrinetzes aus halbgeordneten Abläufen. Dieses Problem werden wir in den auf dieses Kapitel folgenden, theorielastigen Kapiteln ausführlich behandeln.
In den weiteren Ausführungen dieses Kapitels stellen wir konkrete Anwendungsmöglichkeiten für den beschriebenen streng ablauforientierten Modellierungsansatz vor. Zu beachten ist hierbei, dass Anwendungen typischerweise zwei zentrale Anforderungen an entsprechen-

de Syntheseverfahren stellen. Zum einen spielt bei der Modellierung großer Systeme die Performance der Verfahren oft eine wichtige Rolle. So werden in etlichen Fällen effiziente Syntheseverfahren benötigt, um die Berechnung eines Netzmodells aus entsprechenden Ablaufmodellen überhaupt zu ermöglichen. Zum anderen ist es für Anwendungszwecke häufig wichtig, ein möglichst übersichtliches und kompaktes Lösungsnetz und nicht ein beliebiges Lösungsnetz zu synthetisieren. Kompakte Netze erleichtern das Verständnis und händische Analysen des modellierten Systems. Anwender, insbesondere Manager, legen meist einen großen Wert auf leicht lesbare, interpretierbare und kontrollierbare Referenzmodelle eines Systems. Dies gilt in besonderem Maße für automatisch von Syntheseverfahren erzeugte Modelle, denn für diese ist eine händische Feinabstimmung, Weiterverarbeitung und Verwaltung besonders schwierig. Aber auch eine automatische Verarbeitung durch Analyse- Simulations- und Optimierungsverfahren ist für kompaktere Netze effizienter. Wenn das synthetisierte Lösungsnetz sogar als Grundlage für Implementierungen des modellierten Systems dienen soll, hat die Kompaktheit des Netzes im Sinne einer effizienten Implementierung eine besonders große Bedeutung. Die Kompaktheit eines Netzes ist intuitiv natürlich insbesondere durch die Anzahl seiner Stellen und Transitionen bestimmt. Daneben spielen aber auch die Anzahl der Kanten sowie die Kantengewichte und die Anfangsmarkierung eine Rolle.
In diesem Kontext ist häufig auch eine Ergänzung von Syntheseverfahren um interaktive Aspekte sinnvoll. Dies bedeutet, dass die Konstruktion des Systemmodells ausgehend von einer formalen Ablauf-Spezifikation mithilfe von Syntheseverfahren zwar automatisch gesteuert wird, der Modellierer jedoch in den Erstellungsvorgang einbezogen wird. Als Beispiel für ein interaktives Vorgehen wäre es denkbar, dass besonders kritische Systemstellen nicht gänzlich automatisch synthetisiert werden, sondern noch einmal ein Feedback des Modellierers erfordern. Während eine interaktive Vorgehensweise das Modellierungsvorgehen verkompliziert und zusätzlichen Modellierungsaufwand erfordert, so resultieren auch einige Vorteile. Es ergibt sich ein höheres Maß an Flexibilität bei der Modellierung, der Modellierer hat einen größeren Einfluss auf das synthetisierte Systemmodell, problematische Aspekte des zu modellierenden Systems werden noch einmal genau durchdacht und der Modellierer versteht das erzeugte Modell u.U. besser, da er direkt in den Erstellungsprozess einbezogen wird. Letzterer Aspekt ist insbesondere auch im Rahmen der Wartbarkeit des Modells wichtig. Auch wenn dementsprechend das interaktive Einbeziehen des Nutzers bei Syntheseverfahren in vielen Fällen ein interessantes Vorgehen darstellt, beschränken wir uns in dieser Arbeit auf vollautomatische Synthesealgorithmen. Entsprechende interaktive Erweiterungen von Syntheseverfahren sind ein Thema für zukünftige Forschungsarbeiten.
Im weiteren Verlauf dieses Kapitels sollen nun einige interessante Anwendungsbereiche für den streng ablauforientierten Modellierungsansatz mit BPOs und Petrinetzen skizziert werden. Wir arbeiten den Ansatz spezifisch für die verschiedenen Bereiche genauer aus. Wir beginnen mit einer Erläuterung des Modellierungsansatzes im Rahmen der Software-Modellierung. Anschließend diskutieren wir den Ansatz im Bereich der Geschäftsprozessmodellierung und gehen schließlich noch auf die Anwendbarkeit des Ansatzes zur Modellierung von Lernprozessen ein.

2.1 SOFTWAREENGINEERING

Wie schon erwähnt, haben sich gerade im Bereich des Softwareengineering einige Varianten des vorgestellten ablauforientierten Modellierungsansatzes, welcher viele Parallelen zu dem streng ablauforientierten Modellierungsansatz aufweist, durchgesetzt. Dies gilt aber nur im Rahmen der Modellierung komplexer Softwaresysteme, nicht jedoch im Rahmen der Modellierung und Implementierung einfacher Software und Algorithmen. Dies lässt sich sicherlich dadurch rechtfertigen, dass für erfahrene Modellierer bei Systemen bzw. Modellen geringer Komplexität die Denkweise des klassischen Modellierungsansatzes zum Systemverständnis meist ausreicht. Für unerfahrene Modellierer ist dies aber häufig nicht der Fall und sie stoßen schon bei einfacheren Beispielen auf Probleme. Diese Zielgruppe kann durch eine ablauforientierte Herangehensweise schon bei der Konstruktion von einfachen Systemen oder Systemmodellen in einem hohen Maße unterstützt werden. Nun stellen gerade Lernende im Bereich der Informatik entsprechend unerfahrene Modellierer dar. Daher greifen wir in Unterabschnitt 2.1.1 den in [73] vorgeschlagenen Anwendungskontext der Lehre der Informatik für einen streng ablauforientierten Modellierungsansatz mit BPOs und Petrinetzen auf. In Unterabschnitt 2.1.2 gehen wir dann auf existierende ablauforientierte Modellierungsansätze im Bereich großer Softwaresysteme ein und stellen unseren streng ablauforientierten Modellierungsansatz mit BPOs und Petrinetzen in diesem Kontext dar.

2.1.1 *Generierung einfacher Algorithmen*

In diesem Unterabschnitt wird wie in [73, 77] Ablauforientierung bei der Modellbildung als didaktisches Konzept eingeführt. Hintergrund hierzu bildet die von entsprechenden Erfahrungen getragene Annahme, dass viele Lernende der Informatik zwar die Fähigkeit besitzen, die Elemente von Modellierungssprachen und in manchen Fällen auch komplexere Beispiel-Modelle zu verstehen, sie aber nicht ausreichend Kompetenz aufweisen, selbst Modelle zu konstruieren (siehe z.B. [208]). Unsere These ist, dass dies vielfach darauf zurückzuführen ist, wie Modelle von dynamischen Systemen in der Lehre sowohl an der Schule als auch im Hochschulbereich eingeführt und vermittelt werden.
Meist werden zuerst die syntaktischen Konstrukte von Modellierungssprachen eingeführt und erläutert. Das Verhalten entsprechender Modelle wird anschließend mithilfe konkreter Ausführungen von Modellinstanzen diskutiert. Dies führt zur Betrachtung von Abläufen. Abläufe stellen in diesem Rahmen damit die Semantik von Systemmodellen dar. Sie haben überspitzt ausgedrückt dann keine eigenständige Existenzberechtigung, sondern sie sind stets von einem Systemmodell abgeleitet. Dieses Prinzip, welches das Systemmodell in den Vordergrund stellt und eine zugehörige Ablaufsemantik als abgeleitetes Konstrukt vermittelt, wird den Lernenden in den meisten Fällen unterrichtet. Es spiegelt in gewisser Weise den zuvor diskutierten klassischen Modellierungsansatz wieder. Es legt den Lernenden nahe, zuerst ein Systemmodell zu entwickeln und dann ausgehend von diesem Modell Abläufe zu berechnen, um das Modell zu verstehen und zu untersuchen. Der Validierungsschritt des klassischen Modellierungsansatzes, bei dem die konstruierten Modellabläufe mit den gemeinten Abläufen des betrachteten Systems verglichen werden, ergibt sich als logische Konsequenz

dieser Denkweise. Wie ein zu untersuchendes Systemmodell aber erstmals konstruiert werden soll, ist dabei unklar. Hierzu werden wie bei dem klassischen Modellierungsansatz kaum systematische Hilfsmittel vermittelt. Stattdessen wird an die Kreativität der Lernenden appelliert. Insbesondere erfordert der Konstruktionsschritt, wie im Rahmen des klassischen Modellierungsansatzes erläutert, ein gleichzeitiges sammeln/strukturieren, formalisieren und integrieren der unstrukturierten und informalen Informationen über das System. Gerade an dieser Hürde scheitern viele unerfahrene Lernende dann auch bei einfacheren Modellierungsaufgaben.
Bei dem beschriebenen Lehransatz steht eindeutig die Kompetenz des Modellverstehens im Vordergrund. In [43] wird darauf hingewiesen, dass es daneben auch die Kompetenz des Modellerstellens gibt. Diese wird entsprechend obigen Überlegungen in der Lehre häufig zu wenig beachtet, womit sich die anfänglich diskutierten Probleme von Lernenden bei der Modellerstellung erklären lassen. In [43] wird nun vorgeschlagen, die Kompetenz des Modellerstellens getrennt von der Kompetenz des Modellverstehens explizit zu vermitteln. Auch wenn eine strenge Trennung der Lehre der beiden Kompetenzen aus unserer Sicht nicht notwendig ist, stellt die Grundannahme, dass die Fähigkeit des Modellerstellens im Rahmen der Lehre unbedingt mehr Bedeutung erhalten sollte, die Basis für die Überlegungen dieses Unterabschnittes dar. Hierzu empfehlen wir, den Lernenden einen ablauforientierten Modellierungsansatz als Hilfsmittel zu erklären. Hierdurch wird nicht nur die Kompetenz des Modellerstellens in den Vordergrund gerückt, sondern es wird den Lernenden eine sinnvolle Systematik zur Erstellung konkreter Modelle an die Hand gegeben. Ablauforientierung hat sich für große Softwareengineering-Projekte schon als sehr nützlich erwiesen und sollte daher auch in kleinerem Rahmen insbesondere für schwächere Lernende hilfreich sein. Aber auch stärkere Lernende sollten davon profitieren, da sich ein für diese Lerngruppe typisches Problem derart darstellt, dass sie die einfachen Lehr-Beispiele zwar problemlos ad hoc kreativ lösen können, sie aber dann an größeren Modellierungsaufgaben scheitern. Die Vermittlung eines ablauforientierten Modellierungsansatzes wirkt dem entgegen, da ein solcher Modellierungsansatz den Lernenden eine skalierbare, systematische Vorgehensweise an die Hand gibt. Zudem fördert eine Ablauforientierung in der Lehre sicherlich auch das Ablaufverständnis, da Abläufe in diesem Rahmen als eigenständige Konstrukte betrachtet werden müssen. Dies kann zu einem besseren Verständnis von Modellverhalten und somit auch zu einer Verbesserung der Kompetenz des Modellverstehens führen.
Besonders deutlich werden die diskutierten Ideen in dem Kontext von Programmierkursen. Programmiersprachen können als formale Modellierungssprachen für Algorithmen angesehen werden. Typischerweise werden auch Programmiersprachen analog zu den allgemeinen Überlegungen über Modellierungssprachen zunächst als syntaktische Konstrukte eingeführt. Konkrete Programme sollen dann konkrete Problemstellungen für verschiedene erlaubte Eingaben lösen. Ein Programm beinhaltet Kontrollstrukturen wie „Wiederholung“, „bedingte Anweisung“ und Verschachtelungen dieser Kontrollstrukturen, so dass unterschiedliche Eingaben normalerweise zu unterschiedlichen Lösungsabläufen eines Programms führen. Diese Abläufe eines Programms werden aber meist nur zum Verständnis des Verhaltens eines existierenden Programmes und dabei normalerweise auch nur informal

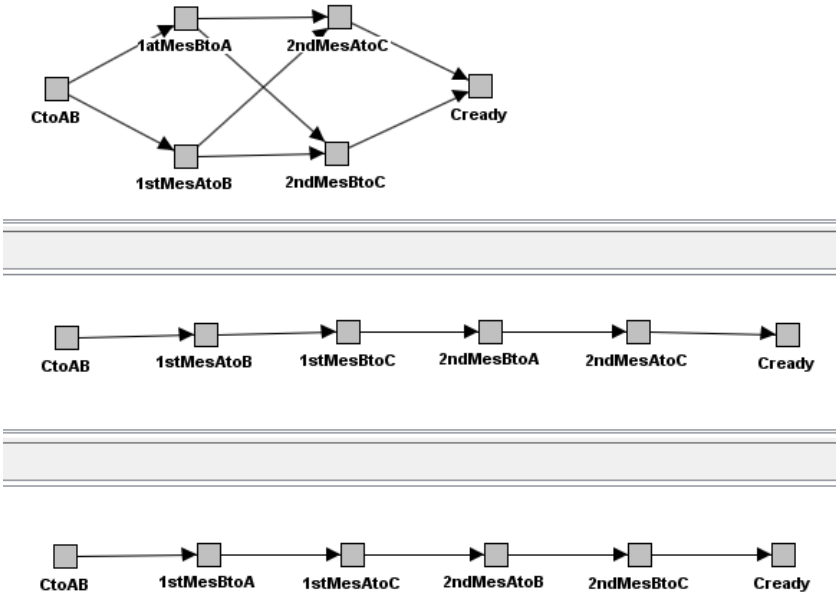

Abbildung 10: Abläufe des Echo-Algorithmus (erste Version).

betrachtet. Vielfach fehlt die Vermittlung systematischer Vorgehensweisen zum Erstellen von Programmen. Dementsprechend fällt es Schülern und Studenten anfänglich oft schwer, eigenständig Programme mit komplexen Kontrollstrukturen zu schreiben. Wir schlagen daher gerade für Programmierkurse eine stärkere Ablauforientierung als Hilfsmittel zur Programm-Erstellung vor. Hierbei soll vermittelt werden, dass ein Programm als Zusammenfassung seiner möglichen Abläufe angesehen werden kann. Sind erst einmal die meist intuitiv verständlichen Abläufe eines Programms bekannt, so kann die betrachtete Menge der Abläufe anschließend durch den geeigneten Einsatz von Kontrollstrukturen in ein Programm übersetzt werden.
Die genannten Argumente für die Vermittlung ablauforientierter Konzepte in der Lehre gelten in besonderem Maße für einen streng ablauforientierten Modellierungsansatz. Spielt Nebenläufigkeit eine Rolle, beispielsweise im Kontext der Lehre von verteilten Algorithmen oder Systemen, so eignet sich unser streng ablauforientierter Modellierungsansatz mit BPOs und Petrinetzen sehr gut. Die folgenden Beispiele zeigen, wie dieser Ansatz die Erstellung von Programmen bzw. Modellen von Algorithmen entsprechend unterstützen kann.
Wir betrachten zwei verteilte Algorithmen aus dem Buch [219]. Das erste Beispiel ist der sog. Echo-Algorithmus. Hierbei sind ein Chef C und zwei Agenten A und B durch drei bidirektionale Nachrichtenkanäle miteinander verbunden. Der Echo-Algorithmus dient dem Zweck, die Funktionstüchtigkeit der Nachrichtenkanäle zu überprüfen. Der Chef sendet zunächst beiden Agenten eine Nachricht. Jeder Agent leitet diese Nachricht an den anderen weiter und der andere Agent leitet sie dann wieder an den Chef zurück. Erhält der Chef also eine Nachricht von beiden Agenten, so wird er fertig und die Funktionstüchtigkeit der Kanäle ist sichergestellt. Die Agenten verhalten sich dabei so, dass sie, sobald sie eine Nachricht erhalten haben, diese an den Partner,

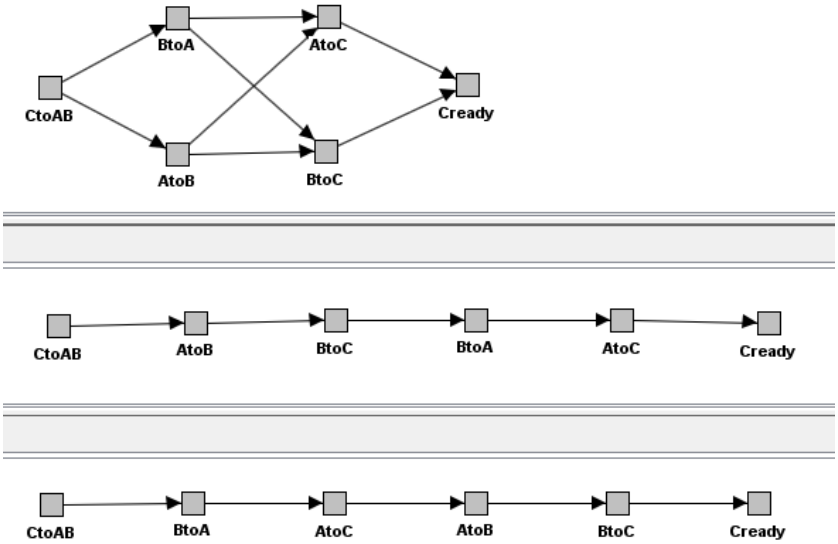

Abbildung 11: Abläufe des Echo-Algorithmus (zweite Version).

welcher nicht der Absender der Nachricht ist, weiterleiten können. Sie müssen die Nachricht aber nicht unbedingt sofort weiterleiten, d.h. sie können, beispielsweise wenn sie beschäftigt sind, auch eine gewisse Zeit abwarten. Damit ergeben sich drei mögliche Abläufe. Zu Anfang sendet C an A und B (CtoAB). Entweder leiten dann beide Agenten die vom Chef erhaltene Nachricht unabhängig voneinander an den anderen Agenten weiter (1stMesAtoB bzw. 1stMesBtoA) oder ein Agent (z.B. B) wartet zuerst noch solange bis er auch die Nachricht des anderen Agenten erhalten hat (1stMesAtoB) und leitet diese Nachricht dann an C weiter (1stMesBtoC), ehe er erst danach die Nachricht von C an den anderen Agenten weiterleitet (2ndMesBtoA). Diese Abläufe sind in Abbildung 10 dargestellt. Bei diesem aus [219] übernommenen Beispiel werden die Ereignisse danach unterschieden, ob sie die erste Nachricht (1stMes) oder die zweite Nachricht (2ndMes) eines Agenten darstellen. Diese Unterscheidung ist aber eher künstlich, da sich die Reihenfolge der Sendevorgänge auch über die kausalen Abhängigkeiten ergibt. Wenn wir diese Unterscheidung aufheben, ergeben sich ganz analog die Abläufe in Abbildung 11.
Ein Netz, welches ersteres Verhalten wiedergibt ist in Abbildung 12 dargestellt, ein Netz für das zweite Verhalten findet sich in Abbildung 13. In beiden Fällen gilt, dass die Modellierung der Abläufe wesentlich einfacher ist als die Modellierung des gesamten Algorithmus. Das Netz in Abbildung 12 ist kompliziert und lässt sich von typischen Lernern, welche unerfahrene Modellierer sind, nicht ohne weiteres erstellen. Einfacher lässt sich der beschrieben Algorithmus aber in der Form eines Petrinetzes nicht darstellen. Die drei zugehörigen Abläufe sind allerdings leicht verständlich und intuitiv modellierbar. Ein entsprechendes Syntheseverfahren kann hieraus dann automatisch das betrachtete Netz erzeugen. Mit diesem Vorgehen lässt sich also auch von Modellierungsanfängern sehr einfach ein Petrinetzmodell des beschriebenen Problems erstellen. Insbesondere kann die anfängliche Betrachtung der Abläufe des verteilten Algorithmus und eine darauf aufbauende durch Syn-

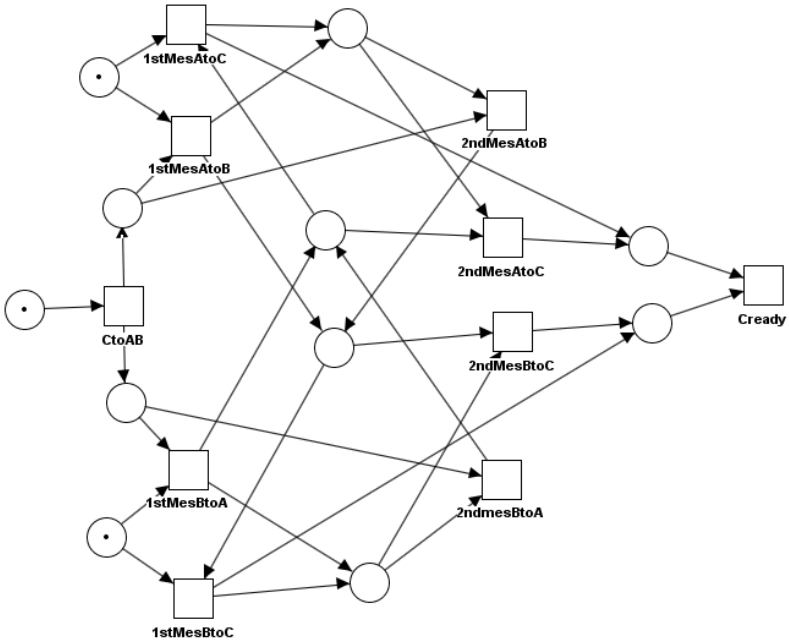

Abbildung 12: Petrinetzmodell des Echo-Algorithmus (erste Version).

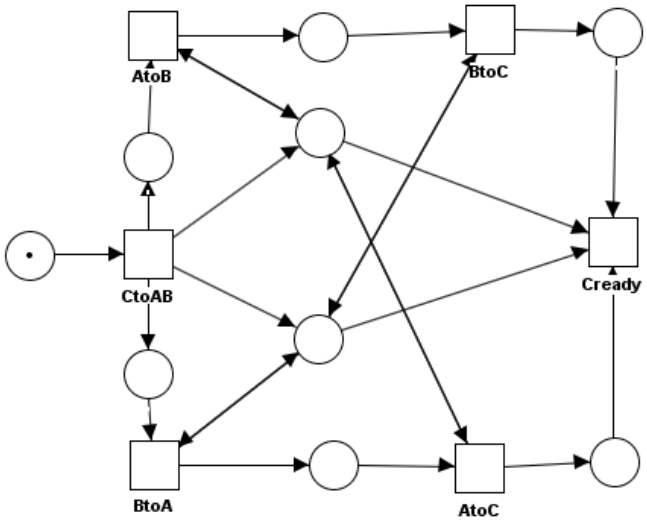

Abbildung 13: Petrinetzmodell des Echo-Algorithmus (zweite Version).

these unterstützte Erzeugung eines integrierten Petrinetzmodells des Algorithmus stark zum Verständnis der Lernenden beitragen. Für das zweite Netz gilt Ähnliches. Es wirkt zwar auf den ersten Blick nicht so kompliziert, enthält aber zwei komplexe „Selbst-Schleifen"-Stellen in der Mitte. Diese stellen das feine, gerade für Lernende anfangs nur schwierig zu verstehende Zusammenspiel von Nichtdeterminismus und Nebenläufigkeit des betrachteten verteilten Algorithmus dar.

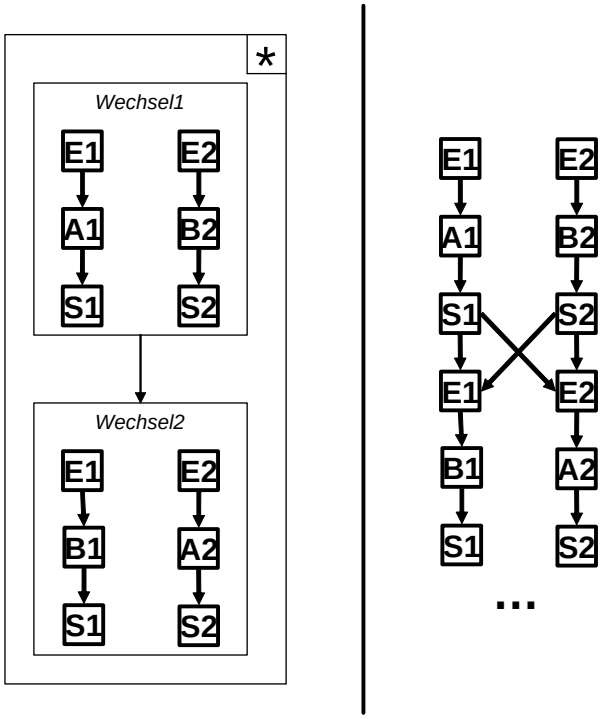

Abbildung 14: Ablaufverhalten der Agenten.

Ein zweites Beispiel eines verteilten Algorithmus stellt die Verabredung von zwei Agenten dar. Jeder Agent hält sich entweder in dem Sektor A oder in dem Sektor B auf. Die Agenten sollen jeweils ihre Orte tauschen. Zu Beginn befindet sich Agent 1 in B und Agent 2 in A. Jeder Agent hat im Anfangszustand dem anderen Agenten eine entsprechende Nachricht über seinen Aufenthaltsort gesendet. Agent 1 und Agent 2 können unabhängig voneinander diese Nachrichten empfangen (E1 bzw. E2), danach ihren Sektor wechseln (A1 für Wechsel von Agent 1 nach A bzw. B2 für Wechsel von Agent 2 nach B) und schließlich eine Nachricht senden, dass sie gewechselt haben (S1 bzw. S2). Dies ist in der BPO „Wechsel1" in Abbildung 14 links dargestellt. Nachdem ein Agent seine Nachricht gesendet hat, muss er warten bis der andere Agent eine entsprechende Nachricht sendet. Es ergibt sich also wieder die Anfangssituation, nun aber mit vertauschten Sektoren. In dieser Situation kann der beschrieben Ablauf also dann mit vertauschten Rollen durchgeführt werden, wie in der BPO „Wechsel2" illustriert. Danach ist schließlich wieder der Ursprungszustand erreicht. Dieses Verhalten soll sich nun beliebig oft wiederholen lassen. Auf der Ablaufebene ist dies, wie in Abbildung 14 links dargestellt, wiederum leicht modellierbar. Es wird mit dem Ablauf „Wechsel1" begonnen, erst dann folgt entsprechend der Pfeilrichtung „Wechsel2". Das *-Kästchen drückt aus, dass diese Abfolge von „Wechsel1" und „Wechsel2" beliebig wiederholt

werden kann. Diese Notation ist nur eine Kurzschreibweise dafür, dass die in Abbildung 14 rechts abgebildete BPO, welche die Hintereinanderausführung der Abläufe „Wechsel1“ und „Wechsel2“ repräsentiert, beliebig oft aneinander gehängt werden kann. Ein Petrinetz mit diesem Verhalten ist in Abbildung 15 illustriert. Mit einem entsprechenden Syntheseverfahren lässt sich dieses automatisch aus dem in Abbildung 14 links gegebenen, intuitiv modellierbaren Ablaufverhalten gewinnen. Das Petrinetz per Hand zu erstellen, ist dahingegen nicht einfach. Daher sind auch hier ähnlich wie im letzten Beispiel Syntheseverfahren bei der Modellierung sehr hilfreich.

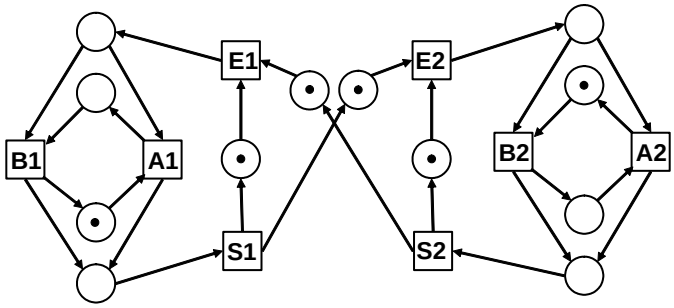

Abbildung 15: Petrinetzmodell der Agenten.

Natürlich erfordern die hier vorgestellten Ideen, wie in [73] ausführlich skizziert, etliche weitere Untersuchungen. Insbesondere ist eine umfassende Evaluation in der Lehre nötig. Erste Ergebnisse in dieser Richtung sind in [77] beschrieben. Tatsächlich vergleichbare Vorarbeiten zu dem vorgestellten Konzept gibt es kaum [73]. Allerdings ist ein Vergleich zu alternativen theoretischen Arbeiten, welche die Entwicklung von Algorithmen aus didaktischer Perspektive beleuchten, wie [225], wichtig. In der genannten Publikation geht es auch um die Betrachtung von Abläufen, allerdings aus einer etwas anderen Sichtweise und mehr aus einem Anwendungsbezug heraus.

2.1.2 *Modellierung komplexer Softwaresysteme*

Im Bereich des Softwareengineering, welcher mit der Erstellung großer, komplexer Softwaresysteme befasst ist, hat konzeptuelle Modellierung viel Beachtung gefunden. Das Hauptziel der Modellierung in diesem Bereich ist es, eine Brücke von den informalen, verteilten Informationen über ein zu implementierendes Informationssystem hin zu einer tatsächlichen Implementierung des Systems zu schlagen. Die wichtigsten Ansätze im Rahmen der Softwaremodellierung waren die in den 1970ern entwickelten Methoden der strukturierten Analyse und des strukturierte Designs, sowie die sich in den späten 1980ern durchsetzenden Konzepte der objektorientierten Analyse und des objektorientierten Designs (siehe z.B. [110]). In den 1990ern wurde schließlich nicht zuletzt aufgrund der gestiegenen Komplexität von Informationssystemen auf breiter Front anerkannt, dass eine systematische Anforderungserhebung – also die Aufnahme, die Dokumentation und die Analyse und Validierung der Anforderungen an ein System – einen fundamentalen Aspekt bei der Entwicklung von Software darstellt. Dementsprechend entwickelte sich in der Folge der Bereich der Anforderungserhebung

zu einem eigenständigen und facettenreichen Forschungsbereich des Softwareengineering [187].
Analysten für Informationssysteme haben in diesem Zuge erkannt, dass eine fehlerhafte Anforderungsanalyse ein Hauptgrund für eine Vielzahl unzufriedenstellender Softwaresysteme und gar für das Scheitern vieler Softwareprojekte ist [95, 187]. Besonders problematisch ist, dass sich Fehler bei der anfänglichen Anforderungsanalyse später kaum mehr beheben lassen, da Kosten zur Fehlerbehebung mit zunehmender Zeit im Entwicklungsprozess der Software exponentiell steigen. Es gibt etliche häufig zitierte Studien [95, 187], welche zuerst einmal zeigen, dass die meisten IT-Projekte entweder abgebrochen werden, dass die Kosten und der Zeitaufwand des Projektes höher sind als erwartet, oder dass die ursprünglichen Projektziele eingeschränkt werden. Weiter zeigen die Studien, dass ein Großteil dieser Probleme auf eine unzureichende Anforderungserhebung zurückgeführt werden kann. Nach diesen Erkenntnissen ist es wahrscheinlich, dass eine Verbesserung der Qualität der Anforderungen durch den Einsatz strukturierter Vorgehensweisen und formaler Modelle bei der Anforderungserhebung sowohl zu einer Verbesserung der Qualität der entstehenden Informationssysteme als auch zur Reduktion der Kosten der Systementwicklung führt. Aktuelle Studien zeigen allerdings, dass sich diese Denkweise bis heute in der industriellen Praxis noch nicht ausreichend durchgesetzt hat [181, 187]. Daher ist die Forschung im Bereich der Anforderungserhebung, insbesondere praxisnahe Aspekte betreffend, auch gegenwärtig noch wichtig und zeitgemäß.
Eine Hürde für den praktischen Einsatz von systematischen Methoden der Anforderungserhebung stellt vor allem der Einstieg in die entsprechenden Konzepte dar. Dementsprechend spielen aus praktischer Sicht die ersten Phasen der Anforderungserhebung eine besondere Rolle. Diese wurden erst in den letzten Jahren umfassend untersucht. Die Grundproblematik ist, dass anfangs Spezifikationsmodelle erwünscht sind, welche das Verhalten eines komplexen Systems insbesondere aus Anwendersicht beschreiben. Für eine Implementierung eines Systems sind dahingegen integrierte zustandsbasierte Modelle erforderlich [110]. Es hat sich gezeigt, dass Szenarien bzw. Abläufe, welche ursprünglich durch Jacobsons Anwendungsfälle eingeführt wurden [123], das Schlüsselkonzept zur intuitiven Erstellung nutzerorientierter Verhaltens-Spezifikationen darstellen [187, 104, 105, 209, 70]. Hier liegt die schon zu Beginn des Kapitels diskutierte Idee zugrunde, dass Fachexperten einzelne Szenarien eines Systems besser kennen und beschreiben können als das System im Ganzen. Insbesondere erlaubt die Instanzebene von Szenarien die Anwendung sehr einfacher und intuitiver Modellierungskonzepte. Folglich helfen Szenarien, um von Anfang an Anforderungsmodelle zu erstellen, welche valide sind in dem Sinne, dass sie die wahren Systemanforderungen getreu widerspiegeln. Diese Sichtweise hat sich in den letzten Jahren im Bereich der Entwicklung komplexer Softwaresystem weitgehend durchgesetzt. Im Bereich von Datenmodellen hat sich eine Herangehensweise, welche frühe Anforderungsmodelle betont, sogar schon etliche Jahre früher etabliert [10, 165].
In dieser Arbeit wurden die wesentlichen Vorteile der Betrachtung von Szenarien am Anfang des Prozesses der Anforderungserhebung schon zu Beginn des Kapitels genauer erläutert (siehe auch [9, 105]). Allerdings können Szenarien das gesamte Verhalten eines Software-

systems nicht in einer angemessenen strukturierten Weise darstellen [110]. Daher erfordern die späteren Phasen der Anforderungserhebung, welche sich mit Aufgaben der Implementierung, des abschließenden System-Designs, der Dokumentation, der Analyse, der Simulation oder der Optimierung beschäftigen, ein integriertes zustandsbasiertes Model, das die vollständige Reaktivität (jeder Komponente) des Systems beschreibt. Die Erstellung einer Szenario-Spezifikation reicht normalerweise also nicht aus, sondern sie bildet die Grundlage zur Konstruktion entsprechender integrierter Modelle.
Auf diesen Ideen basierende Ansätze zur Modellierung von Softwaresystemen mithilfe von Szenarien wurden in den letzten Jahren in der Literatur vielfach untersucht [187, 110, 105]. Dies zeigt sich nicht zuletzt auch an der Vielzahl der in letzter Zeit entwickelten Szenarionotationen, wie dem ITU-Standard der Message-Sequence-Charts, der populären Erweiterung in Form von Live-Sequence-Charts, den Scenario-Trees, den Chisel-Digrammen und besonders den verschiedenen UML-Diagrammen zur Modellierung von Szenarien bzw. von Zusammenfassungen von Szenarien, nämlich Sequenzdiagrammen, Kommunikationsdiagrammen, Aktivitätsdiagrammen, Interaktions-Überblicksdiagrammen und Anwendungsfalldiagrammen [9]. Konkrete Beispiele für Vorgehensweisen zur Modellierung von Softwaresystemen, welche weitestgehend den Prinzipien des beschriebenen ablauforientierten Modellierungsansatzes folgen, finden sich in [150, 9, 187, 96, 104, 105, 209], wobei die ersteren zwei Arbeiten einen sehr guten Überblick über das Thema bieten. Es gibt analytische, automatisiert synthetische und interaktiv synthetische Ansätze zur Konstruktion von Design-Modellen aus Szenarien. In einem radikaleren Ansatz wird sogar vorgeschlagen, eine Szenario-Spezifikation nicht nur als Systemanforderung, sondern als endgültige Implementierung zu betrachten [110]. Einige der existierenden Ansätze weisen auch viele Parallelen zu dem beschriebenen streng ablauforientierten Modellierungsvorgehen auf. Meistens finden sich jedoch einige Aspekte, die sich von dem von uns vorgeschlagenen streng ablauforientierten Modellierungsansatz in seiner reinen Form unterscheiden. Insbesondere steht häufig das Schnittstellenverhalten des Softwaresystems im Vordergrund. Typischerweise besteht das Hauptziel der Modellierung von Szenarien darin, das Schnittstellenverhalten aus der Nutzersicht darzustellen. Dementsprechend wird das System durch entsprechende Szenarien meist hauptsächlich aus einer Sicht von außen beschrieben. Ausgehend von derartigen Szenarien wird dann versucht, ein entsprechendes Schnittstellenverhalten durch geeignete interne Systemkomponenten und deren Reaktivität zu erzeugen. Bei unserem Ansatz liegt dagegen das Hauptaugenmerk auf dem funktionalen bzw. algorithmischen Kern des Systems, aus dem sich erst in zweiter Hinsicht das Schnittstellenverhalten ergibt. Szenarien beschreiben also hier vor allem systeminternes Verhalten. Ein weiterer Unterschied besteht darin, dass die meisten verwendeten Szenarionotation wie Anwendungsfalldiagramme Alternativen und sogar Schleifen erlauben. Derartige Formalismen befinden sich damit auf der logischen Ebene von Systemen und besitzen selbst wiederum Abläufe. Unser Ansatz sieht dahingegen ein konsequentes Herunterbrechen derartiger Konstrukte auf die Ebene einzelner Abläufe vor, so dass das intuitive Abstraktionsniveau von Einzelinstanzen durchgängig eingehalten wird. Schließlich ist es bei unserem streng ablauforientierten Modellierungsansatz wichtig, dass Abläufe zumindest in einem zweiten Schritt in

einer streng formalen Modellierungssprache mit einer klaren Semantik formuliert werden. Während sich ein derartiges Vorgehen auch bei einigen der automatisierten Modellierungsansätze aus [150, 9] findet, so ist es für die üblicherweise verwendeten Beschreibungssprachen für Szenarien wie Anwendungsfalldiagramme doch eher typisch, dass eine Darstellung der Szenarien auf informeller oder semi-formaler Ebene stattfindet.
In diesem Rahmen stellt die von uns vorgeschlagene Realisierung des streng ablauforientierten Modellierungsansatzes mithilfe von Petrinetz-Syntheseverfahren eine interessante Vorgehensweise dar. Zuerst einmal bildet er dadurch eine Ergänzung existierender Ansätze, dass er sowohl durchgängig streng formale Modellierungssprachen verwendet als auch streng formale Beziehungen zwischen dem Systemmodell und den Szenario-Modellen verwirklicht. Außerdem wird zur integrierten Systemmodellierung der interessante, ohnehin vielen entsprechenden Modellierungssprachen aus verschiedensten Anwendungsbereichen zugrundeliegende Formalismus der Petrinetze verwendet. Zur Modellierung von Szenarien verwenden wir BPOs, welche, wie in der Einleitung beschrieben, eine sehr allgemeine Darstellung von Abhängigkeitsbeziehungen von Ereignissen eines Ablaufes erlauben. Daher lässt sich der Ansatz auch in einer Vielzahl von Situationen einsetzen.
Der vorgeschlagene streng ablauforientierte Modellierungsansatz für Softwaresysteme wird im Folgenden durch ein Beispiel illustriert. In dem Beispiel werden Szenarien in der Form von UML-Sequenzdiagrammen in BPOs übersetzt und aus diesen wird dann durch Synthese ein entsprechendes Systemmodell in der Form eines S/T-Netzes erzeugt. Sequenzdiagramme stellen den erfolgreichsten und verbreitetsten UML-Formalismus zur Modellierung von Verhalten dar.
Wir betrachten ein System, welches Forderungen von Kunden in einem Versicherungsunternehmen bearbeitet. Die Abbildungen 16, 17, 18 und 19 zeigen vier Sequenzdiagramme, welche das Verhalten des Systems spezifizieren. Es gibt also vier mögliche Szenarien. Diese fangen alle identisch an. Sie beginnen mit der Meldung einer Forderung einschließlich eines entsprechenden Schadensberichtes von einem Versicherungskunden (Fm). Das System muss dann unabhängig voneinander (dargestellt durch die eckigen Klammern, welche eine Co-Region festlegen) die Forderung in der Unternehmensdatenbank registrieren (Fr) und einen Angestellten zur Prüfung der Versicherung des Kunden sowie einen Angestellten zur Prüfung des Schadenshergangs zu der Forderung zuweisen (Vz und Sz). Anschließend sorgt das System durch einen entsprechenden Eintrag in der Unternehmensdatenbank dafür, dass Rückstellungen für die Forderung gebildet werden (Rz). Diese Aufgabe modelliert die Pflicht aller Versicherungsunternehmen, Reserven zum Ausgleich von Risiken wie Schwankungen im Schadensverlauf oder Prozessrisiken zurückzustellen. Während dessen führen die zwei zugewiesenen Prüfer (sobald sie jeweils die Nachricht über ihre Zuweisung erhalten haben) entsprechende Evaluationen der Forderung durch. Der Versicherungsprüfer untersucht, ob der Kunde eine der Forderung entsprechende gültige Versicherung bei dem Versicherungsunternehmen hat. Der Schadensprüfer führt eine genaue Untersuchung der Schadensmeldung durch, um einen etwaigen Versicherungsbetrug auszuschließen. Wenn diese Prüfungen jeweils durchgeführt sind, übermitteln die Prüfer dem System Ihre Bewertung der Situation (Vb und Sb). Sobald die Reserven zurückgestellt sind, kann das System diese

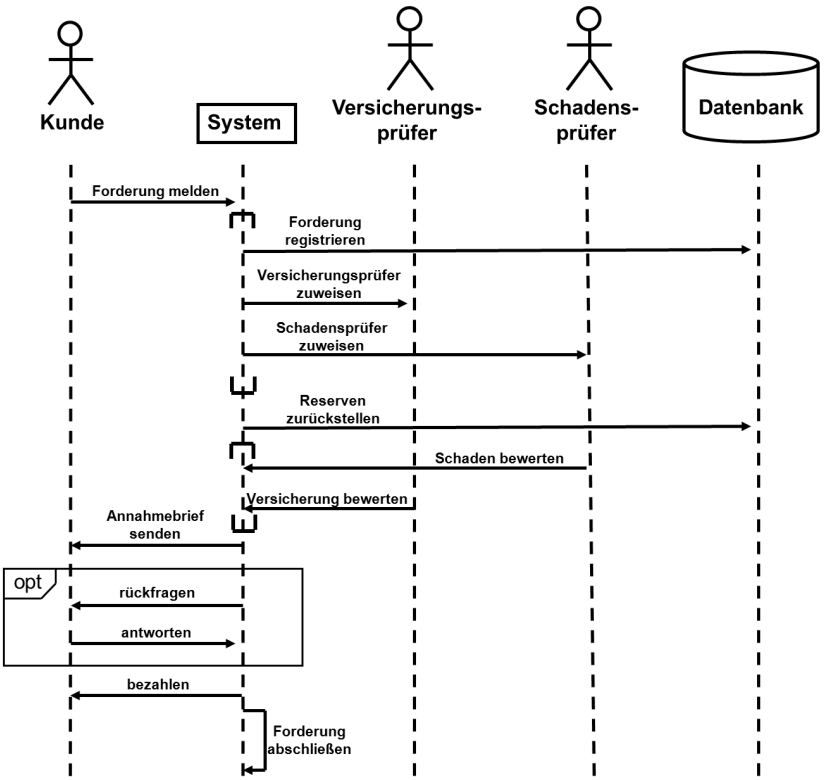

Abbildung 16: Erstes Sequenzdiagramm.

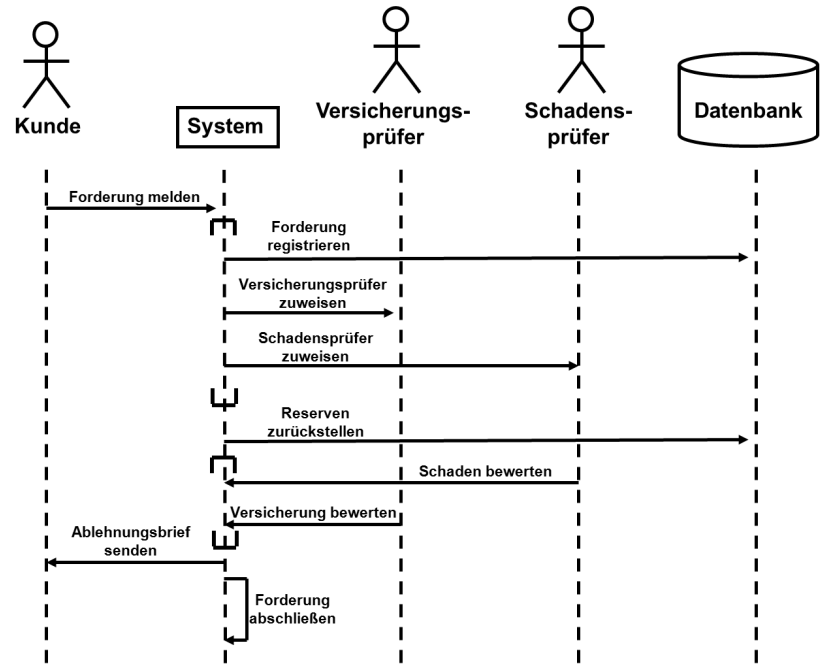

Abbildung 17: Zweites Sequenzdiagramm.

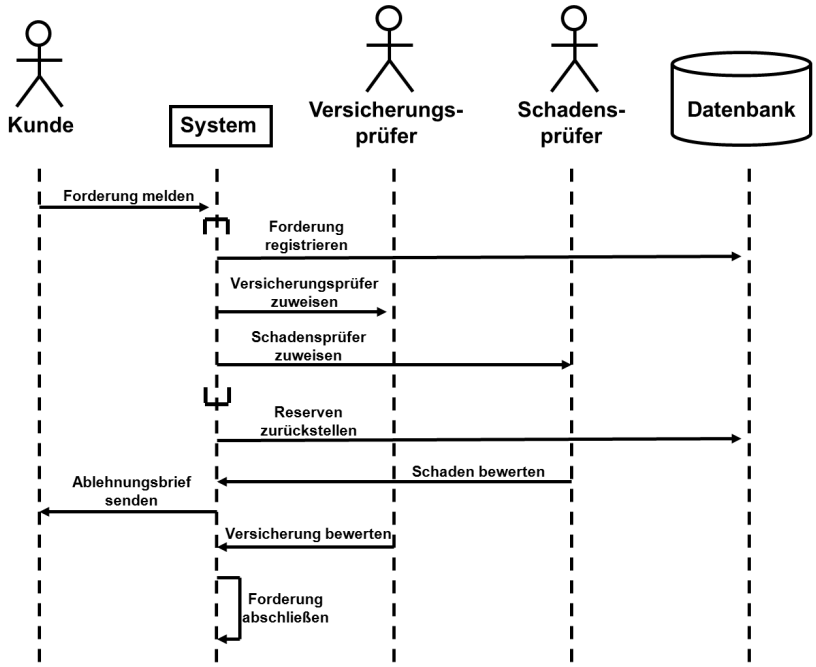

Abbildung 18: Drittes Sequenzdiagramm.

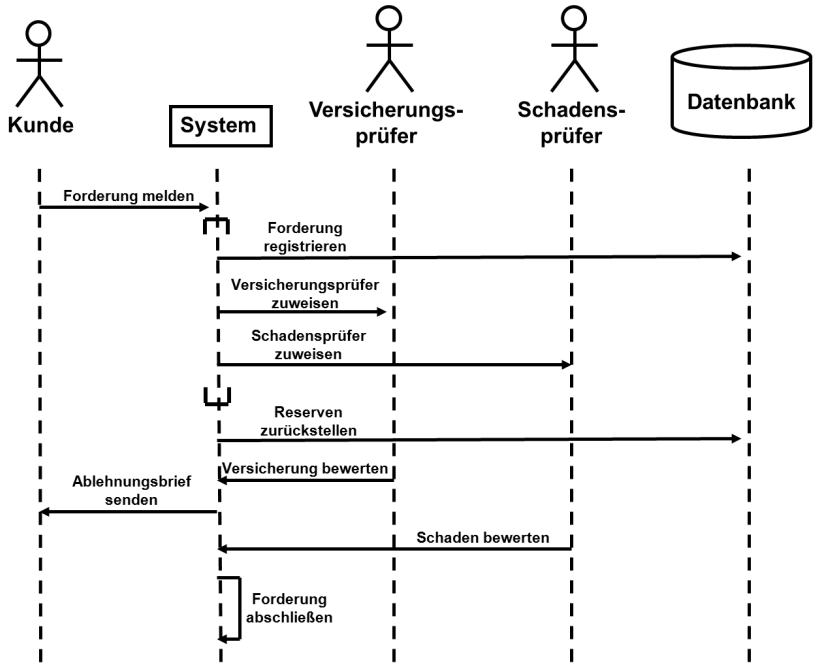

Abbildung 19: Viertes Sequenzdiagramm.

Evaluationen entgegennehmen. An dieser Stelle unterscheiden sich die Szenarien nun.
Das erste Sequenzdiagramm modelliert die Situation, dass beide Evaluationen positiv sind. In diesem Fall werden die Evaluationen unabhängig voneinander vom System empfangen und anschließend eine Meldung an den Kunden gesendet, dass die Versicherung den Schaden übernimmt (Ans). Daraufhin kann das System optional (dargestellt durch das durch den Interaktionsoperanden „opt“ gegebene optionale Fragment) noch Rückfragen an den Kunden zur genaueren Einschätzung der Schadenssumme stellen (r) und dann auf eine entsprechende Antwort des Kunden (a) warten. Dieser Schritt kann aber auch übersprungen werden. Danach zahlt das System den durch die Versicherung abgedeckten Schaden an den Kunden (b) und schließt schließlich die Forderung ab (Fa). Das zweite Szenario stellt die Situation dar, dass nach dem Empfang beider Evaluationen, von einer Bezahlung des Schadens abgesehen wird. In dieser Situation wird also eine entsprechende Meldung an den Kunden verfasst (As) und daraufhin die Forderung abgeschlossen (Fa). Das dritte und vierte Sequenzdiagramm stellen Varianten dieses Szenarios dar. Falls die zuerst empfangene Evaluation, also entweder die Bewertung des Schadensprüfers (im dritten Szenario) oder die Bewertung des Versicherungsprüfers (im vierten Szenario), negativ ist, kann dies unmittelbar zu einer Ablehnung der Forderung führen. Eine Ablehnungsmeldung kann dann sofort versendet werden (As), ohne die andere Evaluation abzuwarten. Diese wird dann erst danach empfangen. Anschließend wird wiederum die Forderung abgeschlossen (Fa).
Wir wollen hier das durch die Sequenzdiagramme beschriebene Verhalten in der Form eines integrierten Petrinetzmodells repräsentieren. Dabei berücksichtigen wir nur die Sendeereignisse, da uns dies in dem Beispiel ausreichend erscheint. Die Sequenzdiagramme lassen sich auf natürliche Weise in die fünf BPOs in Abbildung 20 übersetzen. Dabei ist zu beachten, dass das erste Sequenzdiagramm aufgrund des optionalen Fragmentes zwei mögliche Abläufe definiert. Die Synthese eines Petrinetzes aus den fünf BPOs führt zu dem Netz in Abbildung 21. Dieses gibt die Szenario-Spezifikation getreu wieder.
Wir betrachten nun eine kleine Variante des Beispiels. Das System soll bei dieser Variante so gestaltet sein, dass im Falle des dritten und vierten Sequenzdiagramms die Forderung unmittelbar nach dem Versenden der Ablehnungsnachricht abgeschlossen wird. Es wird in diesem Fall also nicht mehr auf die zweite ohnehin nicht mehr bedeutsame Evaluation gewartet, d.h. die Nachricht Versicherung bewerten bzw. Schaden bewerten wird einfach aus dem dritten bzw. vierten Sequenzdiagramm entfernt. Dementsprechend entfällt das jeweilige Ereignis auch in den zugehörigen BPOs. In diesem Fall ergibt sich nun das Netz aus Abbildung 22, welches die Szenarien wiederum getreu widerspiegelt. Obwohl wir nur eine kleine und leicht einsichtige Änderung der Szenarien vorgenommen haben, ergibt sich hier ein deutlich anderes Netz.
Es ist noch anzumerken, dass die beiden erstellten Netze jeweils die einfachste und kompakteste Möglichkeit sind, das betrachtete Verhalten als Petrinetz darzustellen. Dennoch sind beide Netze relativ komplex. Diese per Hand zu modellieren, ist selbst für erfahrene Modellierer sehr herausfordernd und führt leicht zu Modellierungsfehlern. Dahingegen sind die betrachteten Szenarien leicht einsehbar und verständlich.

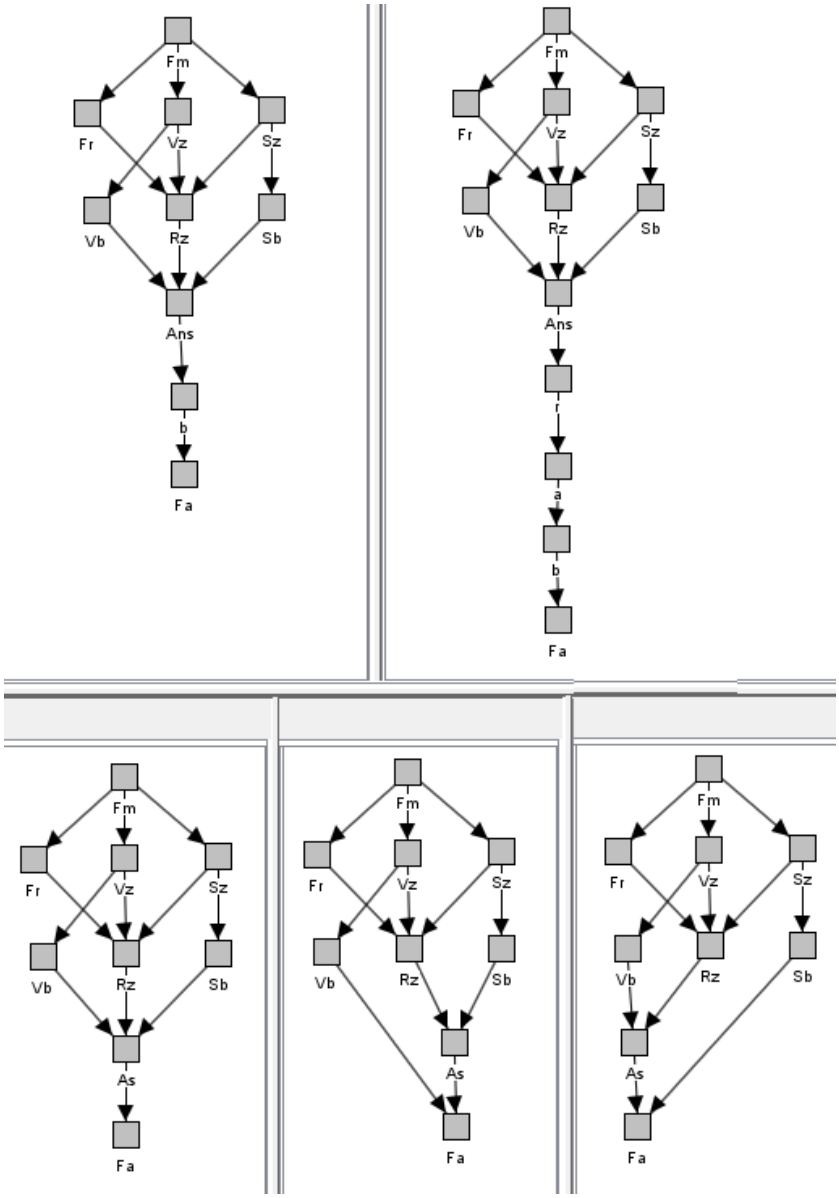

Abbildung 20: Aus Sequenzdiagrammen resultierende BPOs.

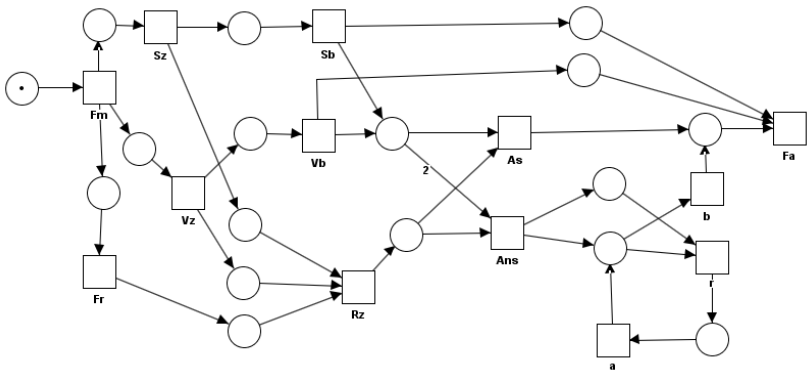

Abbildung 21: Den Sequenzdiagrammen entsprechendes Petrinetz.

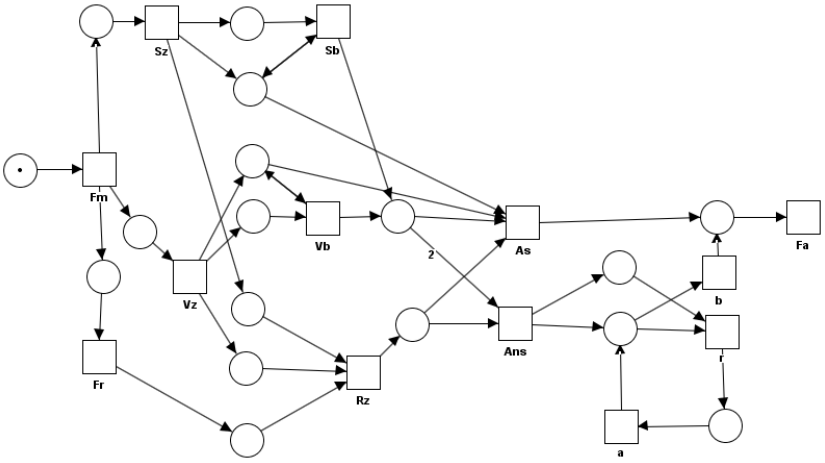

Abbildung 22: Petrinetz im Falle veränderter Szenarien.

Insbesondere sind viele Anwender gut geübt darin, entsprechende Sequenzdiagramme zu erstellen. Daher sollte die Modellierung der Beispiel-Sequenzdiagramme eine intuitive nicht all zu schwere Aufgabe darstellen. Diese lassen sich dann mit entsprechenden Syntheseverfahren automatisch in ein Petrinetzmodell übersetzen. Falls die Szenarien dabei getreu modelliert sind, so ist das Modell so weit dies möglich ist dann auch valide.
Die Beispiele illustrieren daher sehr schön unsere Annahme, dass das unmittelbare Design eines Petrinetzmodelles oft sehr schwierig ist, während die Modellierung von Szenarien und eine darauf aufbauende Verwendung automatischer Syntheseverfahren normalerweise ein wesentlich einfacheres und intuitiveres Modellierungsvorgehen ist. Um dies noch besser zu veranschaulichen, wollen wir nun noch einmal auf den Unterschied in den zwei Beispielen eingehen. Konzeptuell haben wir nur eine kleine Veränderung vorgenommen. Diese lässt sich auch entsprechend einfach in den Szenarien berücksichtigen. Die Änderung bewirkt allerdings auf der kausalen Ebene, also die Abhängigkeiten der Ereignisse betreffend, einen erheblichen Unterschied. Die korrekten Zusammenhänge auf der Ebene des Petrinetzes zu verstehen und zu modellieren, ist hier sehr schwierig. In dem zweiten Beispiel ergibt sich ein wesentlich komplexeres Petrinetzmodell, welches komplizierte Stellen mit „Selbst-Schleifen“ erfordert. Obwohl also der im zweiten Beispiel modellierte Zusammenhang auf den ersten Blick nicht wesentlich komplizierter wirkt als im ersten Fall, erfordert er einen wesentlich komplexeren Kontrollfluss im Petrinetz. Auf der Szenarioebene lassen sich die zwei verschiedenen Zusammenhänge dahingegen, wie die Beispiele zeigen, sehr einfach darstellen und unterscheiden. Natürlich ist das Beispiel klein gewählt. Für größere Systeme wirkt sich die bessere Verständlichkeit von einzelnen Szenarien gegenüber einem Modell des Gesamtsystems noch stärker zugunsten eines Modellierungsvorgehens basierend auf Szenarien und Syntheseverfahren aus.
Ein Aspekt, auf den an dieser Stelle abschließend noch kurz eingegangen werden soll, sind die Schwierigkeiten im Rahmen der Erstellung einer präzisen Szenario-Spezifikation. Die Problematik hierbei liegt darin, dass die Informationen über das zu modellierende System meist verteilt auf verschiedenste Anwender in unstrukturierter, informaler Form vor-

liegen. Während es in einer solchen Situation, wie schon erwähnt, kaum möglich ist, in sinnvoller Weise eine direkte formale Modellierung des Gesamtsystems anzustreben, so ist selbst die Erstellung einer anfangs noch informalen Szenario-Spezifikation, welche in dem Sinne ausreichend präzise ist, dass sie später auch formalisiert werden kann, eine sehr schwierige Aufgabe. Sicherlich stellt die Verteiltheit der Informationen bei der Modellierung von Einzelabläufen ein geringeres Problem dar als bei der Modellierung eines Gesamtsystems, da Informationen über ein Szenario meist von einem Informationslieferanten erlangt werden können. Außerdem sind geeignete Szenario-Informationen im Gegensatz zu strukturierten oder gar formalen gesamtheitlichen Informationen über das System typischerweise im Prinzip verfügbar, da die meisten Anwender ein System über dessen Abläufe verstehen und das System auch mit anschaulichen Beispielabläufen erklären können. Dennoch bleiben zwei wesentliche Probleme. Zum einen stellt sich die Frage, wie die verteilt vorhandenen Informationen systematisch erlangt werden können. Zum anderen besteht meist eine Lücke zwischen dem informalen Denken von Anwendern und der von Systementwicklern erwünschten Präzision von Informationen. Während Anwender oftmals die von Entwicklern verwendeten Ausdrücke und Diagramme kaum verstehen [172], fällt es den Entwicklern umgekehrt häufig schwer, mit der Komplexität der natürlichsprachlichen Informationen von Anwendern umzugehen [98].
Derartige Probleme sind in den meisten ablauforientierten Modellierungsansätzen außer Acht gelassen. Auch wir gehen hier im Rahmen der Modellierung von Softwaresystemen nicht näher auf diese Thematik ein, werden dies aber im Bereich der Geschäftsprozessmodellierung im nächsten Abschnitt in einem etwas anderen Kontext nachholen. In der Literatur der Anforderungserhebung des Softwareengineering finden sich in letzter Zeit einige Arbeiten, die sich gerade mit den besprochenen Problemen befassen. Dabei sind vor allem die Arbeiten [187, 96] zu nennen, welche entsprechende Vorschläge im Rahmen der Erhebung von Szenario-Informationen präsentieren. Wichtige Ansätze, welche sich generell mit dem Weg von unstrukturierten, möglicherweise mehrdeutigen und unklaren, natürlichsprachlichen Informationen über komprimierte, klar strukturierte Informationen hin zu konzeptuellen Modellierungssprachen beschäftigen sind beispielsweise der KCPM-Ansatz (Klagenfurt conceptual pre-design model) [166, 96] oder das Vorgehen aus [98].

2.2 GESCHÄFTSPROZESSMODELLIERUNG

„Allgemein ist ein Geschäftsprozess eine zusammengehörende Abfolge von Unternehmensverrichtungen zum Zweck einer Leistungserstellung.“ [205]. Geschäftsprozesse stellen somit eine verbundene Abfolge von Aufgaben dar. Im Zuge der Ausführung dieser Aufgaben wird eine Leistung, beispielsweise ein Produkt oder eine Dienstleistung, erstellt. Bei der Ausführung der Aufgaben sind in der Regel zahlreiche Akteure und organisatorische Einheiten des Unternehmens oder sogar externe Partner wie Lieferanten beteiligt.
Da Software meist im Kontext von Geschäftsprozessen angewandt wird, ist die Modellierung von Geschäftsprozessen traditionell ein wichtiger Teil vieler Softwareentwicklungs-Projekte [167, 179]. Gerade in den letzten Jahren sind die Anwendungsfelder der Geschäftspro-

zessmodellierung aber stark gewachsen und der Stellenwert von Geschäftsprozessmodellierung ist deutlich gestiegen [6, 223, 87, 205, 5]. Geschäftsprozessmodelle werden mehr und mehr zu reinen Organisationszwecken wie Dokumentation, Prozess-Reorganisation und Prozess-Optimierung, Zertifizierung, aktivitätsbasierter Kostenberechnung und Ressourcen-Planung verwendet. Derartige Aufgaben werden unter dem Begriff des Geschäftsprozess-Management zusammengefasst. Geschäftsprozess-Management hat sich zu einer eigenen bedeutsamen Disziplin, welche an der Schnittstelle zwischen Informatik und Betriebswirtschaftslehre liegt, entwickelt. In vielen Unternehmen hat sich eine neue Sichtweise, bei der nicht mehr die einzelnen Teile bzw. Funktionen sondern die Geschäftsprozesse eines Unternehmens im Vordergrund stehen, durchgesetzt. Die klassische Funktionsorientierung verliert im Gegensatz zur Prozessorientierung immer mehr an Relevanz [205]. Belege hierfür sind beispielsweise die jährliche weltweite Befragung von CIOs (Chief Information Officer) des Gartner Executive Programs (`www.gartner.com/exp`), bei der die Verbesserung der Geschäftsprozesse zum vierten Mal hintereinander als oberste Geschäftspriorität genannt wurde, der Anstieg von akademischen Studiengängen, welche mit Namen wie „Business Process Engineering“ betitelt sind, oder der Stellenwert von Geschäftsprozessen in neueren gesetzlichen Regelungen und Standards wie der Normenreihe EN ISO 9000 ff. und dem Sabanes-Oxley-Act in den Vereinigten Staaten.
Auch der Bereich des Workflow-Management, welcher sich mit der informationstechnischen Unterstützung und Automatisierung von Geschäftsprozessen beschäftigt, gewinnt zunehmend an Bedeutung in der Praxis. Die Nachfrage und das Angebot von sog. Workflow-Systemen wächst sehr stark. Workflow-Systeme stellen eine konsequente technologische Umsetzung der Prinzipien des Geschäftsprozess-Managements dar. Bei traditionellen betrieblichen Informationssystemen sind die zugehörigen Geschäftsprozesse in der Software integriert. Sie sind damit nicht deutlich sichtbar und können insbesondere nur mit größerem Aufwand verändert werden. Modernere ERP-Systeme wie SAP R3 legen die unterstützten Prozesse offen, allerdings erfordert die Änderung von Prozessen (das sog. Customizing) immer noch einen größeren Programmieraufwand. Flexibler sind dahingegen Workflow-Systeme bzw. Workflow-Management-Systeme. Diese stellen Standardsoftware dar, bei der die jeweilige Prozesslogik beliebig und flexibel definiert werden kann und in der Form von Prozessmodellen eine zusätzliche Eingabe des Systems darstellt. Prozesslogik und Anwendungsfunktionalität sind hier also konsequent getrennt. Die zentrale Komponente eines Workflow-Systems ist eine Workflow-Engine. Eine Workflow-Engine steuert und überwacht mithilfe eines Modells eines Geschäftsprozesses die korrekte Ausführung der Arbeitsschritte des Geschäftsprozesses durch die relevanten Akteure und Systeme. Neben der eigentlichen Workflow-Engine gehören zum Workflow-Management Techniken und Werkzeuge zur Prozessentwicklung, -definition und -analyse, zur Interaktion mit Anwendern und Applikationen, zur Administration, und zur Kooperation mehrerer Workflow-Systeme.
Einen guten Überblick über die Charakteristika und Komponenten eines Workflow-Systems bietet die von der Workflow Management Coalition (WfMC) vorgeschlagene Workflow-Referenzarchitektur [87, 5, 223] (Abbildung 23). Ihre einzelnen Bestandteile haben folgende Bedeutung:
Process Definition Tools: Hierbei handelt es sich um Werkzeuge zum

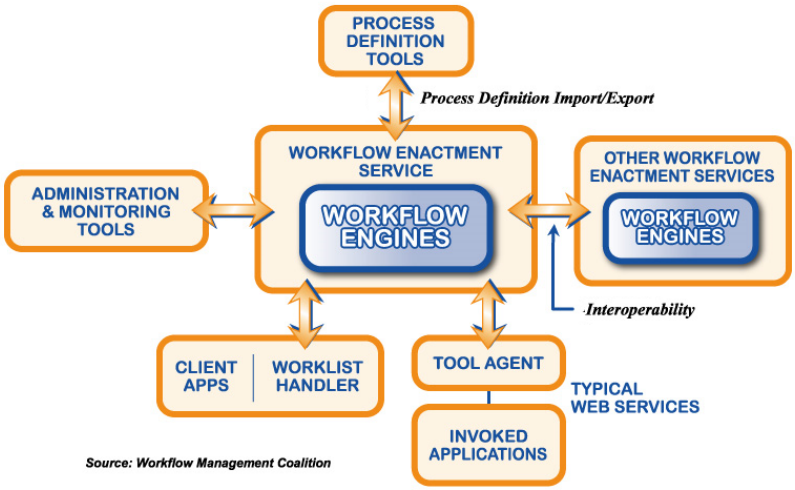

Abbildung 23: Workflow-Referenzarchitektur.

Erstellen und Verändern von Prozessmodellen bzw. allgemeiner Prozessdefinitionen, aber auch für ihre Analyse. Die Prozesse werden in einer geeigneten Form über eine Schnittstelle von den Workflow-Engines gelesen. Eine Prozessdefinition legt unter anderem Aufgaben, ihre Reihenfolge und notwendige Ressourcen (Benutzer, externe Programme, ...) fest. Aufgaben können nebenläufig oder auch alternativ ausführbar sein.
Workflow Engine: Der Workflow-Enactment-Service erzeugt für die eingehenden Fälle Prozessinstanzen, die von einer oder mehreren Workflow-Engines ausgeführt bzw. gesteuert werden. Für jeden Fall wird jede auszuführende Aufgabe Work-Item genannt. Ihr werden geeignete Ressourcen gemäß der Prozessdefinition zugeordnet.
Workflow Client Applications: Hiermit sind Interaktions-Schnittstelle zu den Anwendern, insbesondere über Worklists, die die für einen Anwender bearbeitbaren Work-Items zeigen, gemeint.
Invoked Applications: Dies sind Interaktions-Schnittstelle zu externen Anwendungen, die interaktiv oder vollautomatisch Work-Items bearbeiten können.
Workflow Interoperability: Für die Kooperation mehrerer Worflowsysteme werden in dieser Schnittstelle Standardfunktionen und Protokolle festgelegt.
Administration and Monitoring: Administratoren und Verantwortliche einzelner Prozessinstanzen können über diese Schnittstelle Prozessparameter setzen, Statusinformationen bekommen und allgemein Daten zur Analyse und Reorgansisation von Prozessen erheben.
Die genannten Entwicklungen zeigen, dass Geschäftsprozessmodellen in Unternehmen eine immer wichtigere Rolle zukommt. In diesem Kontext hat die Validität von Geschäftsprozessmodellen eine große Bedeutung. Trotz der gestiegenen Anforderungen an Geschäftsprozessmodelle, gibt es in der Praxis aber nur geringe Anstrengungen zur Sicherstellung der Validität von Geschäftsprozessmodellen. Meist

werden Prozessmodelle ad hoc – normalerweise in Workshops oder von einem einzelnen Experten – konzipiert, ohne dass die genauen Anforderungen der verschiedenen Nutzer detailliert untersucht oder gar dokumentiert werden. Dies ist sehr problematisch, da reale Geschäftsprozesse häufig sehr komplex sind. Einzelne beteiligte Personen überblicken und verstehen solche Prozesse normalerweise nicht in ihrer Gänze. Häufig ist ihnen nicht einmal die genaue Ablaufstruktur ihres eigener Verantwortungsbereiches bewusst. Aber auch in der Theorie gehen die meisten Ansätze im Bereich der Geschäftsprozessmodellierung von der Validität der Modelle aus und konzentrieren sich auf Analyse- und Optimierungsfragestellungen. Für die eigentliche Modellierung eines Prozesses wird zumeist der zu Beginn des Kapitels beschriebene klassische Modellierungsansatz implizit unterstellt, welcher allerdings, wie schon diskutiert, die Validität eines erstellten Modells normalerweise nicht garantieren kann. Die Analyse und die Optimierung von invaliden Modellen ist aber nutzlos und auf invaliden Modellen basierende Entscheidungen oder gar Implementierungen beispielsweise in einem Workflow-System führen normalerweise zu Fehlern. Dies kann insbesondere bei der Modellierung von sicherheitskritischen Systemen fatale Folgen haben. Es besteht hier also ein Mangel an Ansätzen, welche sich systematisch mit der Modellierung von Geschäftsprozessen im Hinblick auf die Sicherstellung von Validität befassen.
Wir machen in dieser Arbeit den neuartigen Vorschlag, den vorgestellten streng ablauforientierten Modellierungsansatz ähnlich wie im Softwareengineering auch zur Modellierung valider Geschäftsprozesse zu verwenden. In diesem Kontext sind dann Syntheseverfahren im Bereich der Geschäftsprozessmodellierung relevant. Ein kleines Beispiel für ein solches Vorgehen haben wir schon in der Einleitung im Rahmen der Modellierung eines Begutachtungsprozesses für Dissertationen behandelt. Das dort diskutierte Beispiel hat deutlich gezeigt, inwiefern der streng ablauforientierte Modellierungsansatz und insbesondere die Synthese von Petrinetzen aus BPOs zur Modellierung von Geschäftsprozessen eingesetzt werden kann. Das Beispiel hat auch schon einige positive Aspekte und Vorteile eines solchen Vorgehens deutlich gemacht. In Unterabschnitt 2.2.1 wird ein entsprechendes Vorgehen einschließlich der zugrundeliegenden Ideen im Detail beschrieben.
Interessant ist in diesem Kontext, dass das vor einigen Jahren sehr erfolgreich eingeführte Process-Mining [3, 1] als spezielle Ausprägung des streng ablauforientierten Modellierungsansatzes betrachtet werden kann. Den Ausgangspunkt des Process-Mining bildet die Erstellung einer formalen Ablauf-Spezifikation aus Log-Dateien eines Informationssystems. Eine solche Spezifikation soll dann automatisch in ein konsistentes Systemmodell übersetzt werden. Hierzu sind entsprechende Konstruktionsverfahren notwendig. Insbesondere lassen sich an dieser Stelle wiederum Syntheseverfahren sinnvoll einsetzen. Diese Zusammenhänge beleuchten wir in Unterabschnitt 2.2.2 genauer.

2.2.1 *Ablauforientierte Geschäftsprozessmodellierung*

Wir sind im Rahmen eines Industrieprojektes mit dem Beschaffungsbereich der AUDI AG auf das Problem der Erzeugung valider Prozessmodelle gestoßen. Die korrekte Modellierung der existierenden Prozesse ist für AUDI besonders auch aus dem Grund wichtig, dass die Dokumentation der Geschäftsprozesse in letzter Zeit ein immer wichtigerer

Bestandteil der TÜV-Zertifizierung (Technischer Überwachungs-Verein) für deutsche Automobilhersteller wurde. Die relevanten Prozesse sind aber sehr komplex. Sie sind häufig bereichsübergreifend und ihnen liegt eine sehr heterogene Systemlandschaft mit etlichen Medienbrüchen zugrunde. In diesem schwierigen Umfeld fragte AUDI nach unserer akademischen Unterstützung bei der Erstellung valider Prozessmodelle.
Die uns gestellte Aufgabe wies viele Ähnlichkeiten zu den Problemstellungen des Softwareengineering auf. Dementsprechend war unsere Idee, in diesem Umfeld die gleichen Mittel zu verwenden, welche im IT-Bereich schon erfolgreich waren. Wir behaupten also, dass ein ablauforientierter Modellierungsansatz auch zur Erstellung valider Geschäftsprozessmodelle geeignet ist. Entsprechend den Argumentationen zu Beginn des Kapitels sehen wir gar einen streng ablauforientierten Modellierungsansatz als besonders vielversprechend an. Die generellen Vorteile eines solchen Ansatzes wurden am Anfang des Kapitels schon erläutert. Wir glauben, dass sie im Bereich der Geschäftsprozessmodellierung analog wie im Software-Bereich Gültigkeit haben. Einen entsprechenden Modellierungsansatz haben wir im Rahmen des angesprochenen Industrieprojektes im Detail entwickelt. Wir haben hierbei alle Modellierungsphasen von der Problemstellung der Modellierung eines Geschäftsprozesses bis hin zum endgültigen Modell ausgearbeitet. Wir stellen diesen Ansatz im Folgenden vor.
Bei dem Ansatz ist zu beachten, dass zwar eine Fokussierung auf die Abläufe eines Prozesses im Vordergrund steht, aber gerade in einem realen Kontext auch etliche weitere Aspekte berücksichtigt werden müssen. Daher genügt es nicht eine reine Szenario-Spezifikation eines Prozesses zu erstellen, sondern es soll eine weiter gefasste Spezifikation betrachtet werden, welche sich aber hauptsächlich auf Szenarien abstützt. Außerdem ist noch zu erwähnen, dass der präsentierte Ansatz, wie in dem Projekt verlangt, die Aufgabe der Modellierung eines IST-Prozesses beschreibt, analog aber auch zur Erstellung eines SOLL-Prozesses verwendet werden kann. Schließlich lassen sich auch problemlos verschiedenste hierarchische Modellierungskonzepte in den Ansatz einbinden.
Der Modellierungsansatz ist großteils an die im letzten Abschnitt kurz beschriebenen Vorgehensweisen aus dem Bereich der Anforderungserhebung für Software-Systeme angelehnt. Somit ergeben sich viele Parallelen zu entsprechenden Softwareengineering-Ansätzen. Allerdings erfordern bereichsspezifische Probleme die Anwendung etlicher neuer Techniken. Beispielsweise liegt der Fokus bei der Modellierung eines Software-Systems vor allem auf diesem einen System selbst. Dahingegen hat ein Geschäftsprozessmodell häufig einen größeren Geltungsbereich, welcher viele Systeme beinhaltet und sogar Organisationsgrenzen überschreitet. Weiter müssen bei der Software-Modellierung teilweise schon Implementierungsaspekte berücksichtigt werden, während Geschäftsprozessmodelle weitgehend implementierungsunabhängig sind. Insbesondere stellen Geschäftsprozesse auch vollständig von Systemen losgelöste Aspekte, wie Interaktionen zwischen relevanten Menschen dar [223]. Schließlich sind die zentralen Aspekte bei der Software-Modellierung die Komponenten oder Objekte eines Systems, die Kommunikation und die Abhängigkeiten zwischen Komponenten und eine Unterscheidung von Inter- und Intraobjektverhalten. Bei der Geschäftsprozessmodellierung spielen die eher als global zu betrach-

tenden Aktivitäten die entscheidende Rolle [223, 6]. Dabei sind die Abhängigkeiten der Aktivitäten durch Vor- und Nachbedingungen und die mit Aktivitäten verknüpften Ressourcen gegeben. Modularität kommt hier durch geeignete Verfeinerungs- und Kompositionskonzepte ins Spiel.
Detaillierte Vorschläge, um Konzepte aus dem Bereich der Anforderungserhebung im Softwareengineering auf die Geschäftsprozessmodellierung zu übertragen, wurden erstmals in den Artikeln [167, 200] vorgestellt. Im diesen Arbeiten werden erste wichtige Beiträge zu diesem Thema im Rahmen einer geeigneten Adaption des KCPM-Ansatzes [166] geleistet. Allerdings wird bei den genannten Ansätzen nicht auf die bei der Anforderungserhebung so zentrale Betrachtung von Szenarien eingegangen. Es gibt auch einige weitere Arbeiten, welche die Nützlichkeit von Methoden der Anforderungserhebung für das Design von Geschäftsprozessmodellen erwähnen, siehe beispielsweise [224, 118, 55]. Diese berücksichtigen aber weder alle wichtigen Aspekte der Anforderungserhebung – insbesondere die Anfangsphase der Anforderungserhebung wird häufig vernachlässigt – noch gehen sie bei den einzelnen Aspekten ausreichend ins Detail.
Ansonsten gibt es zwar etliche Ansätze zur Modellierung von Geschäftsprozessen, aber so weit wir das überblicken können, werden kaum systematische Vorgehensweisen zur Gewinnung geeigneter Informationen über einen Prozess und zur Integration dieser Informationen in ein Prozessmodell vorgeschlagen. Interessante Beispiele für Ansätze, welche zumindest am Rande auf derartige Probleme eingehen, sind [179, 224, 55] und einige der Ansätze aus [118]. Von diesen ist insbesondere der ARIS-Ansatz [206, 204] in der Praxis sehr erfolgreich. Aber auch bei diesen Ansätzen liegt der Fokus eher auf Fragestellungen späterer Modellierungsphasen, während die Erhebung der Anforderung an ein Prozessmodell meist nicht näher betrachtet wird. Daher sind alleine schon die explizite Erstellung einer Spezifikation für ein Prozessmodell und die damit verbundenen Vorgehensweisen der frühen Modellierungsphasen wesentliche Neuerungen des hier vorgestellten Ansatzes. Völlig neuartig ist auf jeden Fall die Betrachtung von Szenarien und erst recht die Betrachtung formaler Abläufe in diesem Rahmen.
Neben den im letzten Abschnitt angesprochenen Arbeiten aus dem Bereich des Softwareengineering und den in den letzten zwei Absätzen genannten Arbeiten zu Problemen der Geschäftsprozessmodellierung sollen hier noch einige weitere Literatur-Beispiele genannt werden, welche Parallelen zu gewissen Aspekten des in diesem Unterabschnitt diskutierten Modellierungsansatzes aufweisen. Zunächst betrachten wir Arbeiten aus dem Bereich der Geschäftsprozessmodellierung. Die Arbeiten [72, 76, 83, 128] verwenden die Nutzersicht von Szenarien zur Konstruktion von Geschäftsprozessmodellen. Im Rahmen von Instanz-EPKs [203] und ähnlichen Konzepten werden ebenfalls detailliert Szenarien untersucht, allerdings hauptsächlich zu Analysezwecken. Etliche Arbeiten wie [114, 171, 128] beschäftigen sich mit der formalen Integration verschiedener Sichten auf einen Prozess. Die Bücher [6, 87, 223] schneiden einige Strategien für frühe Modellierungsphasen an. Schließlich werden aktuell an unterschiedlichsten Stellen etliche nutzerorientierte Modellierungstechniken für Geschäftsprozesse wie Design-Prinzipien (Top-Down, Bottom-Up und Inside-Out-Ansätze), Ansätze zum Management von Modellierungsaktivitäten (Terminologie, Konventionen, Prozess-Governance und Prozess-

Ownership), Werkzeugunterstützung für etliche Modellierungsaktivitäten (siehe http://bpmn.org), Referenzmodelle (Best Practices), Design-Schablonen (`www.workflowpatterns.com`) und Modellierungsrichtlinien (Qualitätskriterien) diskutiert [191]. Außerhalb des Bereichs des Softwareengineering und des Geschäftsprozess-Management sind letztendlich hauptsächlich Arbeiten, welche sich generell mit Vorgehensweisen zur Beschaffung von Informationen beschäftigen, relevant. Hierzu gibt es in verschiedensten Fachrichtungen interessante Ansätze wie beispielsweise [38, 119].

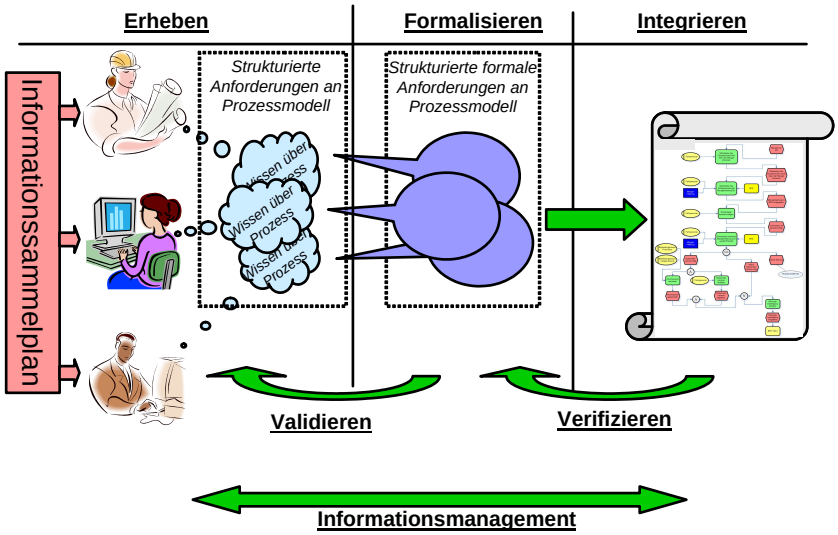

Abbildung 24: Streng ablauforientierter Ansatz zur Modellierung von Geschäftsprozessen.

Im weiteren wird nun unser innovativer streng ablauforientierter Ansatz zur Erstellung valider Geschäftsprozessmodelle beschrieben. Er ist in Abbildung 24 grob skizziert. Er teilt sich in die fünf Phasen Aufnahme, Formalisierung, Validierung, Integration und eine sechste orthogonale Phase des Informationsmanagements auf. Den Ausgangspunkt des Ansatzes stellt ein verteilt vorhandenes Wissen über einen Geschäftsprozess in einer realen informalen Umgebung dar. Der Fokus der Aufnahme-Phase ist die Sammlung der notwendigen Informationen über den Prozess. Hierbei sollen im Rahmen unseres Ansatzes hauptsächlich Szenario-Informationen gewonnen werden, natürlich spielen aber auch etliche andere Arten von Informationen eine Rolle. Das Hauptproblem bei der Aufnahme-Phase besteht darin, die teilweise vagen Informationen aus verschiedenen Informationsquellen zu verstehen und in einer geeigneten Wissens-Datenbank zusammenzufügen. Hierfür wird zuerst ein Informationssammelplan erstellt, welcher entsprechende Erhebungs-Strategien festlegt. Basierend auf diesem werden Informationen eingeholt und in Informationssammelblättern festgehalten. Jedes Informationssammelblatt muss dann zuerst gefiltert und anschließend strukturiert dokumentiert werden. Letzteres bedeutet, dass die Informationen zunächst in Informationsstücke aufgeteilt und nach bestimmten Kriterien klassifiziert werden und die Informationsstücke anschließend systematisch unter Berücksichtigung aller relevanten Zusammenhänge festgehalten werden. Somit entsteht eine strukturierte Sammlung relevanter Informationsstücke. Diese Infor-

mationsstücke werden in der nächsten Modellierungsphase in eine formale Form übertragen. Insbesondere sollen formale Ablaufmodelle erstellt werden. Daneben entstehen aber auch andere ergänzende formale Informationsartefakte. Die formalen und damit präzisen Informationsartefakte ermöglichen eine zuverlässige Validierung, ob sie die wahren Anforderungen an den Prozess korrekt darstellen, in einer folgenden Feedback-Phase. Die validierten Informationsartefakte dokumentieren dann die endgültigen Anforderungen an ein Prozessmodell. Die Artefakte werden in der folgenden Phase in ein mit einer geeigneten Modellierungssprache dargestelltes Geschäftsprozessmodell integriert. Die formale Spezifikation und die Fokussierung auf Abläufe ermöglicht hierbei die Anwendung automatischer Syntheseverfahren zur Erzeugung des Modells. In der abschließenden Phase wird das Prozessmodell dann noch einmal bzgl. der dokumentierten Anforderungen verifiziert. Auch diese Phase profitiert von einer formalen Spezifikation des Prozesses, da eine solche eine formale Verifikation, ob das Prozessmodell der Spezifikation genügt, ermöglicht. Während all dieser fünf Phasen ist es in der orthogonalen Phase des Informationsmanagements notwendig, den Fortschritt der Informationsgewinnung zu verwalten und die erhaltenen Informationen zu organisieren. Insgesamt stellt sich unser Modellierungsansatz, wie beschrieben, als eine Folge von fünf Phasen mit einer parallelen Phase dar. Allerdings lässt sich bei einer tatsächlichen Durchführung des Modellierungsvorgehens eine solch strenge Anordnung der Phasen nicht einhalten, sondern die Phasen werden sich typischerweise überlappen. Im Folgenden werden nun die einzelnen Phasen detailliert beschrieben.

Aufnahme: Wir unterteilen die Aufnahme-Phase in die folgenden neun Einzelschritte: Ziele und Rahmen festlegen, Modellgrenzen bestimmen, grobe Modellstruktur definieren, Erhebungsplan erstellen, Erhebungsmethode festlegen, Dokumentationsmethode festlegen, Informationen sammeln, Informationen filtern und Informationen integrieren/klassifizieren. Die ersten drei Schritte haben hierbei eher vorbereitenden Charakter. Die folgenden sechs Schritte sind die eigentlichen Kern-Schritte der Aufnahme-Phase.

Den Ausgangspunkt bei der Modellierung eines Geschäftsprozesses stellt eine Festlegung des Rahmens und der Ziele des Modellierungsprojektes dar. Vor allem die organisatorischen Rahmenbedingungen des Projektes müssen geklärt werden. Der nächste Schritt ist dann eine klare Festlegung der Grenzen des zu modellierenden Prozesses. Der Prozess muss in seine Umgebung eingeordnet werden, so dass ein Überblick gewonnen werden kann, was genau modelliert werden soll. Insbesondere die Schnittstellen zu anderen Prozessen sollten deutlich gemacht werden. Nachdem der Kontext des Geschäftsprozesses auf diese Weise geklärt ist, soll die grobe Struktur des Prozesses einschließlich von Prozesszielen, zugehörigen Organisationsstrukturen und relevanten Dokumenten und Systemen identifiziert werden. Es soll also ein High-Level-Modell des Prozesses erstellt werden. Hierzu sollte wenn möglich ein Experte mit einer entsprechenden High-Level-Sicht auf den Prozess zu Rate gezogen werden. Auf diese Weise ergibt sich ein guter Übersicht über den Prozess. Dies hilft bei der Überlegung, wie geeignete Informationen für die detaillierte Modellierung des Prozess gewonnen werden können. Hiebei stellt sich vor allem die Frage nach dem Informationsbedarf und nach zugehörigen Informationsquellen. Mit diesen Fragestellungen beschäftigt sich der nächste Schritt der Erstellung eines

Informationssammelplans. Ein solcher legt detailliert fest, welche Informationen benötigt werden und von welchen Informationslieferanten und Dokumenten diese gewonnen werden sollen. Der Plan soll im nächsten Schritt dadurch ergänzt werden, dass für die verschiedenen Informationsquellen Strategien zur Gewinnung der Informationen festgelegt werden. Es müssen also geeignete Erhebungsmethoden gewählt werden. Aber auch andere Aspekte wie beispielsweise die Reihenfolge der Erhebung der verschiedenen Informationen spielen hier eine Rolle. Zur Wahl der Erhebungsmethoden stellen wir hier einen konkreten in den meisten Situationen sinnvoll anwendbaren Ansatz vor, welcher gerade die Aufnahme von Szenario-Informationen unterstützt. Es gibt etliche Möglichkeiten zur Erhebung der Anforderungen an den zu modellierenden Prozess von entsprechenden Informationslieferanten. Zu nennen sind hier Interviews, Beobachtungen, automatische Aufzeichnungen, Rollenspiele, Fragebögen, Workshops, etc. [187, 119]. Oft gibt es neben den Informationslieferanten auch etliche Typen von Dokumenten wie Arbeitsanweisungen, schon bestehende Prozessmodelle, Intranet-Informationen oder akademische Abschlussarbeiten, welche hilfreiche Informationen über den zu modellierenden Prozess bereitstellen. Praktische Erfahrungen legen nahe, dass zuerst all diese Dokumente bearbeitet werden sollten bevor die wichtigen Informationslieferanten konsultiert werden. Dies ist sinnvoll, um schon vor dem Kontakt mit den Informationslieferanten einen guten Überblick über den Prozess zu erhalten. Die Aufnahme der Anforderungen direkt von Informationslieferanten ist aufwändig, insbesondere für die Informationslieferanten, so dass ein gutes Vorwissen über den Prozess wünschenswert ist. Zur Erhebung der Anforderungen von den Informationslieferanten schlagen wir dann Interviews mit einem speziellen Fokus auf die Diskussion von Szenarien vor. Die Interviews sollten sich an dem folgenden Leitfaden orientieren. Es wird mit einer Vorstellungsrunde begonnen. Dann wird das Ziel des Interviews dem Informationslieferanten erklärt. Dabei sollten insbesondere das gewünschte Abstraktionsniveau, eine Abgrenzung des relevanten Prozessteils, dessen Umfeld und, falls schon verfügbar, auch die relevanten Schnittstellen deutlich gemacht werden. Eine Vermittlung des Konzepts von Szenarien vor dem eigentlichen Interview unterstützt die Gewinnung sinnvoll strukturierter Informationen. Ähnlich positive Effekte ergeben sich auch durch eine Illustration der Prozessmodellierungssprache anhand existierender Prozessmodelle. Anschließend wird mit der Aufnahme einzelner Szenarioinstanzen typischerweise in der Form wirklich stattgefundener Beispiele begonnen. Aus unserer Erfahrung vermitteln Informationslieferanten ihr Wissen sehr gut und auch sehr gerne auf diesem Weg. Somit ist dies eine effiziente Möglichkeit, um die relevanten Informationen in einer klar verständlichen Form zu erhalten. Diese werden dann in strukturierter Textform dokumentiert. Gemeinsam mit den Informationslieferanten wird dann versucht, schematische Szenarien aus den diskutierten Szenarioinstanzen abzuleiten. Dieser Schritt wird durch vorgefertigte Szenarioschablonen bzw. Fragebögen, welche geeignet auszufüllende Felder für alle wichtigen Aspekte eines Szenarios enthalten, unterstützt. Ist eine solche Schablone vollständig ausgefüllt, so wird nach Details bzgl. einzelner wichtiger Aktivitäten, Ereignisse, vorkommender Systeme, Geschäftsobjekte und Akteure des Szenarios gefragt. Für jede der genannten Detailinformationen gibt es auch wieder eine eigene Schablone, um entsprechende Anforderungen zu

dokumentieren. Anstelle von Szenarien können analog auch einfache Prozessfragmente betrachtet werden. Prozessfragmente erlauben auch Alternativen und Schleifen sowie die Berücksichtigung von Bedingungen bzw. Ereignissen. Modellierungserfahrene Informationslieferanten können komplexe Prozessstrukturen manchmal leichter in der Form solcher Fragmente beschreiben. Aus diesen lassen sich dann aber unmittelbar wieder einzelne entsprechende Szenarien ableiten, so dass sich hier kein Bruch in dem Vorgehen ergibt. Generell stellen Szenarien aber ohnehin das einfachere und verständlichere Modellierungskonzept dar. Zuletzt bleibt noch zu erwähnen, dass es bei den Interviews allgemein wichtig ist, auf die Vollständigkeit und die Klarheit jeder aufgenommenen Information zu achten. Auch die Einhaltung einer angemessenen Gesprächs-Intensität bei den Interviews ist entscheidend [119].
Wenn die Erhebungsmethoden festgelegt sind, muss eine zu diesen konsistente Dokumentationsmethode zur strukturierten Speicherung der Informationen ausgewählt werden. Auch hier machen wir einen konkreten Vorschlag, welcher obigen Erhebungsansatz fortsetzt. Die verschiedenen ausgefüllten Schablonen, am wichtigsten natürlich die zentralen Szenario-Schablonen, bilden hier schon einen Teil der Dokumentation. Wir verwenden die Schablonen jedoch nicht nur in den Interviews, sondern sie stellen generell eine Maske zur Speicherung entsprechender Informationen dar. Soweit wie möglich werden alle gewonnenen Informationen in die verschiedenen Schablonen eingefügt. Die Schablonen werden dann in eine Datenbank übertragen, wobei eine Tabelle für jeden Schablonentyp angelegt wird. Informationen, welche keiner Maske entsprechen, werden in einer weiteren Tabelle der Datenbank als allgemeine Informationen gespeichert. Diese Informationen werden mit einigen allgemeinen Angaben wie der zugehörigen Informationsquelle versehen. Insbesondere sehen wir für solche Informationen im Hinblick auf eine Integration der Informationen in ein Prozessmodell und eine automatische Verarbeitung der Informationen eine zu dem ARIS-Ansatz ähnliche [206] grobe Klassifizierung in eine Prozess-, eine Daten- und eine Organisationssicht als sinnvoll an. Die Prozesssicht umfasst die für den Kontrollfluss des Prozesses relevanten Informationen, also hauptsächlich Informationen über die Ablaufstruktur von Szenarien, aber auch Detailinformationen zu einzelnen Aktivitäten und Ereignissen. Die Datensicht umfasst Informationen zu physischen Objekten, Datenobjekten und Systemen. Die Organisationssicht beinhaltet Informationen zu Akteuren, Verantwortlichkeiten, und Zugriffsrechten. Im Gegensatz zu dem ARIS-Ansatz betrachten wir keine Kontrollsicht, da diese in der bei uns dominanten Prozesssicht in natürlicher Weise integriert ist. Es ist zu beachten, dass die verschiedenen Schablonen von Haus aus einer Sicht zugeordnet sind, so gehören beispielsweise die Szenario-Schablonen zur Prozesssicht. Eine derartige datenbankorientierte Dokumentation der Informationen erlaubt es, später durch geeignete Datenbankabfragen und Suchfunktionalitäten verschiedene interessante Perspektiven auf die gespeicherten Anforderungen an ein Prozessmodell zu generieren. Es ist auch möglich, automatisch Anforderungsdokumente, welche gewissen vorgegebenen Standards entsprechen, zu erzeugen.
Wir schlagen zusätzlich zu der Anforderungsdatenbank den Aufbau eines Wörterbuches (ähnlich zu [166]) vor, um eine konsistente Sprache für das Prozessmodell festzulegen und ähnliche Begriffe bei der Dokumentation zu harmonisieren. Für einen Begriffsabgleich wird jeder Ein-

trag in dem Wörterbuch mit einer kurzen Beschreibung ähnlich wie in einem Glossar und einer Liste möglicher oder verwendeter Synonyme ähnlich wie in einem Thesaurus versehen. Um eine effiziente Navigation durch das Wörterbuch zu ermöglichen, sollen die Einträge durch Beziehungen verbunden werden, so dass eine Thesaurus-artige Repräsentation des bereichsspezifischen Vokabulars des Geschäftsprozesses entsteht. Hierbei ist vor allem eine hierarchische Ordnung zwischen den Einträgen sinnvoll, um bestimmte Begriffe möglichst schnell auffinden zu können. Ähnlich wie im Bereich der objektorientierten Modellierung ist eine Verfeinerung der Hierarchie in eine „ist-ein"-, eine „Teil-von"- und eine „Eigenschaft-von"-Beziehung sinnvoll. Darüberhinaus ist aber auch eine Assoziations-Beziehung zur generellen Verknüpfung von Einträgen wichtig. Durch eine graphische Repräsentation ist es möglich, das Wörterbuch sinnvoll zu den Informationslieferanten rückzukoppeln und somit die Konsistenz der Begriffsbildung zu unterstützen. Hierzu ist ein Werkzeug zur Visualisierung der Beziehungen der Einträge, welches insbesondere intuitive Ausblendungs- und Such-Funktionalitäten aufweisen sollte, notwendig. Da die wesentlichen Komponenten eines Prozessmodells Aktivitäten, Ereignisse und zugeordnete Objekte einschließlich Akteuren sind, schlagen wir schließlich zur besseren Lesbarkeit eine graphische Unterscheidung der Begriffe für Aktivitäten, Ereignisse und Objekte im Wörterbuch vor. Das gesamte Wörterbuch sollte letztendlich auch noch mit den Informationen in der zugrundeliegenden Anforderungs-Datenbank verknüpft sein, indem beispielsweise Hyperlinks zwischen dem Wörterbuch und entsprechenden Begriffen in der Datenbank eingefügt werden.

Sind Erhebungsmethoden und Dokumentationsmethoden gewählt, so ist der nächste Schritt die tatsächliche Durchführung der Informationsaufnahme entsprechend des Informationssammelplans. Mithilfe der gewählten Erhebungsmethoden müssen die Informationen gewonnen und in lose arrangierte Informationssammelblätter eingetragen werden. Informationssammelblätter können gänzlich unstrukturiert sein oder aber wie die erwähnten Schablonen eine gewisse Struktur aufweisen. Selbst wenn bei der Erhebung der Informationen soweit möglich ein Wörterbuch, wie oben beschrieben, zu Rate gezogen wird, so lassen sich dennoch bei jeglicher Art von Informationsaufnahme gewisse Probleme nicht in Gänze vermeiden. Typischerweise enthalten die verschiedenen Informationssammelblätter Redundanzen, Wiederholungen, Synonyme und Homonyme, spezielle Ausnahmefälle, implizite Informationen und Verwechslungen zwischen verschiedenen Abstraktionsebenen wie Schema- und Instanzebene. Dementsprechend ist der nächste Schritt eine Filterung aller Informationssammelblätter. Die Informationen sollen in einer möglichst klaren und deutlichen Form aufgeschrieben werden. Dabei sollen insbesondere die oben aufgezählten Probleme korrigiert werden. Ein Wörterbuch im Sinne eines Thesaurus oder Glossars ist hier natürlich hilfreich. Der letzte Schritt der Aufnahme-Phase besteht schließlich in einer Klassifizierung und Integration des gesammelten Wissens. Die Informationen der Informationssammelblätter werden in geeignete Informationsstücke aufgeteilt, klassifiziert und in strukturierter Form entsprechend der verwendeten Dokumentationsmethode gespeichert. Es ist hierbei sinnvoll, bei der Dokumentation der Informationsstücke Referenzen auf die zugehörigen Informationssammelblätter abzuspeichern.

Formalisierung: Die Formalisierungs-Phase teilen wir in zwei Schritte auf: Formalisierungsmethode wählen und Informationen formalisieren/modellieren. In dieser Phase werden die Anforderungen an das Prozessmodell formal präzisiert. Während bei informalen und semiformalen Anforderungen in der folgenden Phase der Validierung u.U. nicht alle Missverständnisse zwischen dem Informationslieferant und dem Rezipient ausgeräumt werden können, so ist dies bei formalen Modellen normalerweise möglich.
Der erste Schritt in dieser Phase ist die Auswahl geeigneter Modellierungsformalismen. Es gibt eine Vielzahl interessanter Formalismen. Die Formalisierungskonzepte sollten mit der Dokumentationsmethode harmonisiert sein. Daneben hängt die Wahl typischerweise auch von der Ziel-Prozessmodellierungssprache, vorhandener Werkzeugunterstützung und Projektvorgaben ab. Generell sollten graphische Modellierungssprachen bevorzugt werden. Unser Vorgehen für den zentralen Anforderungsteil der Szenarien ist eine Formalisierung als BPOs. Die generellen Vorteile der Modellierung von Szenarien mithilfe von BPOs haben wir schon erläutert. Ein Argument, welches gerade für die Verwendung von BPOs zur Modellierung von Szenarien von Geschäftsprozessen spricht, ist, dass BPOs genau dem Konzept der Instanz-EPKs entsprechen. Nach unserem Wissen stellen Instanz-EPKs [203] die einzige, wenn auch noch wenig verbreitete, Geschäftsprozessspezifische Szenario-Notation dar. Einige Anwendungen und konzeptuelle Ideen für Instanz-EPKs sind in [83] beschrieben. Schließlich ist die Verwendung von BPOs im Rahmen des hier vorgestellten Ansatzes aus dem Grund besonders sinnvoll, dass das Konzept der BPOs einerseits sehr gut mit den von uns verwendeten Szenario-Schablonen der Dokumentationsmethode abgestimmt ist und andererseits auch in den folgenden Modellierungsphasen wieder aufgegriffen wird. Neben einer Formalisierung der Szenarien kann ergänzend auch die Verwendung einiger weiterer Formalisierungskonzepte sinnvoll sein. Im Bereich der Prozesssicht bietet sich beispielsweise eine formale Darstellung von Prozessfragmenten, Vor- und Nachbedingungen, Hierarchiebeziehungen zwischen Aktivitäten und Ereignissen und Verhaltenseinschränkungen wie Invarianten an. Im Rahmen der Datensicht sind ER-Diagramme und verwandte UML-Konzepte interessant. Für die Organisationssicht sind Organigramme und formale Darstellungen von Gruppen- und Rollenbeziehungen relevant.
Nach der Festlegung der Formalisierungsmethoden werden die in der letzten Phase gebildeten strukturierten Informationsstücke entsprechend der gewählten Methoden in formale Repräsentationen übersetzt. Die entstehenden formalen Anforderungsmodelle werden Informationsartefakte genannt. Die formalen Modelle werden jeweils mit den zugehörigen dokumentierten Informationsstücken verknüpft, so dass sich die ursprünglichen Informationen in jedem Fall wieder rückverfolgen lassen.
Validierung: Generell sieht unser Ansatz eine Validierung der einzelnen Anforderungen an das zu erstellende Prozessmodell vor. Dies liegt daran, dass einzelne Anforderungen und dabei insbesondere Szenarien besser verstanden und damit auch validiert werden können als ein gesamtes Prozessmodell. Außerdem lassen sich durch eine frühe Validierungs-Phase Fehler in den Anforderungen frühzeitig erkennen und mit geringem Aufwand beheben. Die Validierungs-Phase wird bei unserem Ansatz mithilfe der in der Phase Formalisierung erstellten for-

malen Informationsartefakte durchgeführt. Für diese findet, wie in den nächsten Absätzen beschrieben, eine sorgfältige, umfassende Validierung statt, bei der insbesondere Validierungs-Qualitätsziele, welche von jedem einzelnen Artefakt erreicht werden müssen, angewandt werden. Somit findet entsprechend obigen Überlegungen eine Validierung der einzelnen Anforderungen statt, allerdings erst auf der formalen Ebene. Der Vorteil dabei ist, dass auf dieser Ebene eine unmissverständliche Validierung durchgeführt werden kann. Zudem ist es für das weitere Vorgehen essentiell, dass gerade die formalen Anforderungen korrekt und vollständig sind. Dementsprechend ist an dieser Stelle eine umfassende Validierung notwendig, da alle Schritte sowohl der Aufnahme als auch der Formalisierungs-Phase potentielle Fehlerquellen darstellen. Dennoch ist es neben dieser systematischen Haupt-Validierungs-Phase wichtig, Validierung auch als eine Aufgabe anzusehen, die in allen Modellierungs-Phasen, insbesondere auch den ersten beiden Phasen der Aufnahme und der Formalisierung, eine Rolle spielen sollte. Beispielsweise sollten wenn möglich das vorbereitende High-Level-Modell des Prozesses, der Erhebungsplan und die dokumentierten noch informalen Informationsstücke, insbesondere die ausgefüllten Schablonen, kurz validiert werden.
Die eigentliche Haupt-Validierungs-Phase unseres Ansatzes unterteilen wir in drei Einzelschritte: Analysieren, Validieren bzgl. Korrektheit und Validieren bzgl. Vollständigkeit. Den ersten Validierungsschritt bezeichnen wir also als Analyse der Informationsartefakte. Er setzt sich ausschließlich mit den in der Form formaler Informationsartefakte dargestellten Informationen und den zugrundeliegenden Informationsstücken auseinander, ohne Informationslieferanten einzubeziehen. Die unmissverständlichen Informationsartefakte ermöglichen eine Überprüfung auf Inkonsistenzen, Konflikte und ähnliche Probleme der Anforderungen. Konflikte können eigentlich nur auftreten, wenn mehr als eine Informationsquelle für eine bestimmte Information zu Rate gezogen wurde. Es ist wichtig, Konflikte in diesem frühen Modellierungsstadium zu identifizieren, zu analysieren und aufzulösen. Sie sollten auch dokumentiert werden, da es möglich ist, dass dasselbe kontroverse Thema im Weiteren noch einmal auftritt. Ein Beispiel für eine Inkonsistenz innerhalb der Informationsartefakte könnte eine Vorbedingung sein, welche niemals als Nachbedingung auftritt. Derartige Probleme können sogar mit automatischen oder semi-automatischen Hilfsmitteln und Analyseverfahren aufgedeckt werden. Beispiele hierfür sind geeignete Matching-Verfahren zur Suche nach Inkonsistenzen zwischen Vor- und Nachbedingungen, Syntax-Checker zur Überprüfung der syntaktischen Korrektheit der Artefakte oder Muster-Analyseverfahren und statistische Methoden zum Auffinden von möglicherweise widersprüchlichen Mustern in den Anforderungsartefakten. Auch mögliche Inkonsistenzen zwischen den Informationsartefakten und den zugrundeliegenden Informationsstücken sollten näher untersucht werden. So ist es beispielsweise sinnvoll, zu prüfen, ob die Aktivitäten eines Ablaufmodells mit den Aktivitäten, welche in einer entsprechenden Szenario-Schablone vorkommen, übereinstimmen. In diesem Kontext ist es auch wichtig, dass die dokumentierten Informationsstücke noch einmal genauer bzgl. formaler Kriterien überprüft werden. So sollten die ausgefüllten Schablonen keine leeren Felder beinhalten und der Inhalt einiger Felder muss eine korrekte Form aufweisen. Insgesamt werden in dem ersten Validierungsschritt Probleme und Unklarheiten der Informationsarte-

fakte aufgedeckt. Allerdings können diese nur mithilfe zusätzlicher Informationen von den Informationslieferanten aufgelöst werden. Daher findet der Analyse-Schritt vor den folgenden Validierungsschritten statt, bei denen dann die Informationslieferanten wieder konsultiert werden. Die in der Analysephase aufgedeckten Probleme stellen hierbei schon einige wichtige, in jedem Falle zu diskutierende Punkte dar.
Den Validierungsteil, bei dem die Informationslieferanten einbezogen werden, teilen wir in die zwei Schritte der Validierung bzgl. Korrektheit und der Validierung bzgl. Vollständigkeit auf. Im Rahmen der Validierung bzgl. Korrektheit wird versucht, Konflikte, Inkonsistenzen und ähnliche Probleme, welche sich insbesondere aus dem Analyseschritt ergeben, aufzulösen. Die wichtigste Aufgabe dieses Schrittes ist aber eine detaillierte Überprüfung, ob die modellierten Informationsartefakte tatsächlich die von den entsprechenden Informationslieferanten beabsichtigten Anforderungen an das Prozessmodell repräsentieren. Das Hauptziel hierbei ist die Ausräumung von Fehlern, welche aus Missverständnissen der Aufnahme-Phase resultieren, und von Fehlern, welche in fehlerhaften Übertragungen der informalen Informationsstücke zu formalen Anforderungsartefakten in der Formalisierungs-Phase begründet sind. Dementsprechend werden die Informationsartefakte mit den zugehörigen Informationslieferanten diskutiert. Es wird sichergestellt, dass die Artefakte genau die Vorstellungen der Informationslieferanten widerspiegeln. In diesem Rahmen kann auf Standard-Validierungs-Techniken aus dem Bereich des Softwareengineering wie Inspektion, Review und Walkthrough zurückgegriffen werden [187]. Die in unserem Ansatz zentralen Ablaufmodelle sind für ein derartiges Vorgehen besonders gut geeignet, da sie die Anforderungen der Informationslieferanten sehr deutlich und klar darstellen. Eine Rückkopplung von Informationsartefakten nicht nur zu den zugehörigen Informationslieferanten sondern zusätzlich auch zu anderen Informationslieferanten mit einer anderen Perspektive auf das Artefakt oder gar zu externen Spezialisten, kann bei wichtigen Informationen sinnvoll sein. Generell kann die Diskussion der Informationsartefakte in kollaborativen Sitzungen wie auch in Einzelinterviews stattfinden. Abschließend stellen wir noch einige interessante Techniken zur Unterstützung der Validierung bzgl. Korrektheit vor. Bei dem Perspektiven-basierten Lesen muss sich der Informationslieferant auf eine spezielle Sichtweise oder Rolle konzentrieren, um Probleme aufzudecken [187]. Auch automatische Validierungsverfahren lassen sich in dem formalen Rahmen nutzen, beispielsweise können Ablaufmodelle simuliert oder durchgespielt und bzgl. Performance getestet werden. Ebenso ist eine Erstellung von prototypischen Prozessmodellen zumindest für Teile des Prozesses sinnvoll. Dies stellt schon eine erste Einsatzmöglichkeit für Syntheseverfahren dar. Die Analyse von Prototypen soll neue Einsichten über die dem Prototyp zugrundeliegenden Ablaufmodelle erbringen.
Der letzte Schritt der Validierungs-Phase ist schließlich die Validierung der Artefakte bzgl. Vollständigkeit. Mithilfe der Informationslieferanten wird versucht, noch fehlende Informationen zu ergänzen. Auch wenn wir hier eine konzeptuelle Trennung vorgenommen haben, so sollte dieser Schritt dennoch nicht unabhängig von dem vorherigen Validierungsschritt betrachtet werden. Normalerweise finden die Schritte der Validierung bzgl. Korrektheit und der Validierung bzgl. Vollständigkeit gemeinsam statt. Die Schritte werden meist innerhalb derselben Diskussionsrunde durchgeführt und größtenteils werden dieselben

Validierungs-Techniken verwandt. Dennoch haben wir eine Unterscheidung der zwei Schritte vorgenommen, da die Validierung bzgl. Vollständigkeit einen neuen Fokus aufweist. Es soll geprüft werden, ob die vorhandenen Informationsartefakte in dem Sinne vollständig sind, dass sie den Prozess umfassend beschrieben und keine Lücken bestehen. Wir schlagen für diesen Schritt die folgenden speziell zum Auffinden fehlender Informationen geeigneten Validierungs-Techniken vor. Zuerst einmal kann eine entsprechende Untersuchung der Ablaufmodelle Hinweise auf kontextuell noch fehlende Abläufe einbringen. Hierbei ist insbesondere auch eine Rückkopplung von Informationsartefakten zu relevanten Informationslieferanten, welche ursprünglich nicht die zugehörigen Informationen geliefert haben, vielversprechend. Interessant ist auch die Identifikation übereinstimmender Zustände in verschiedenen Abläufen und das Auffinden struktureller Abhängigkeiten unterschiedlicher Abläufe. Hiermit kann auf Abläufe geschlossen werden, welche sich durch neue Kombinationen von Teilen der betrachteten Abläufe ergeben. Der wichtigste Aspekt bei der Validierung bzgl. Vollständigkeit ist aber sicherlich eine Überprüfung, ob alle Kontextaspekte des Prozesses berücksichtigt sind. Beispielsweise sollte geprüft werden, ob alle zuvor identifizierten Schnittstellen, Stakeholder und Umgebungsobjekte des Prozesses durch die Ablaufmodelle geeignet abgedeckt sind.

Integration: Die nächste Phase des Modellierungsansatzes ist die Integration der formalen Informationsartefakte in ein formales Prozessmodell. Unser Vorgehen ist an dieser Stelle die Erstellung einer entsprechenden Petrinetz-basierten Darstellung des Prozesses. Die allgemeinen Vorteile der Modellierung mit Petrinetzen wurden schon in der Einleitung erläutert. Insbesondere im Bereich der Geschäftsprozessmodellierung ist die Betrachtung von Petrinetzen äußerst sinnvoll, da Petrinetze in diesem Bereich sehr beliebt sind. Dies liegt wohl daran, dass der verteilten Ausführung von Aktivitäten und dem feinen Zusammenspiel von Nebenläufigkeit und Nichtdeterminismus im Rahmen des Geschäftsprozess-Managements eine besondere Bedeutung zukommt. Wichtige und erfolgreiche anwendungsorientierte Modellierungssprachen für Geschäftsprozesse wie EPKs, BPMN oder BPEL basieren wesentlich auf den Konzepten von Petrinetzen. Die Hauptaufgabe der Integrations-Phase ist somit eine Integration der spezifizierten BPOs in einem Petrinetz, welches ein entsprechendes Verhalten aufweist. Hierzu gibt es nicht mehr all zuviel zu sagen, denn dies ist gerade wieder das Syntheseproblem, welches wir in dieser Arbeit untersuchen. Die zentrale Integrations-Phase des Modellierungsansatzes erfordert also wiederum entsprechende Petrinetz-Syntheseverfahren.

Allerdings sind hier häufig einige Besonderheiten zu beachten. Zum einen repräsentieren die spezifizierten BPOs in einem hochgradig verteilten Umfeld nicht unbedingt vollständige Abläufe des betrachteten Geschäftsprozesses. Gesamtabläufe können sehr groß sein. Außerdem kann auch ihre Anzahl im Falle vieler aufeinander folgender Alternativen sehr groß sein. Aus diesen beiden Gründen werden von den Informationslieferanten manchmal nur Teilabläufe beschrieben. Dann kann es nötig sein, einzelne spezifizierte BPOs vor der Anwendung von Syntheseverfahren noch zu Gesamtabläufen zusammenzusetzen, beispielsweise können zwei spezifizierte BPOs den ersten und den zweiten Teil oder zwei parallele Teile eines Gesamtablaufs darstellen. Die Schnittstellen zwischen den Abläufen können dabei z.B. durch Vor- und

Nachbedingungen identifiziert werden. Zum anderen müssen in der Integrations-Phase auch die über die Ablaufmodelle hinausgehenden Informationsartefakte berücksichtigt werden. Derartige Informationen können aber zumeist einfach zu einem synthetisierten Petrinetzmodell ergänzt werden. Beispielsweise können Transitionen mit zugehörigen Verantwortlichkeiten oder Systemen annotiert werden. Ansonsten ergeben sich u.U. auch Varianten des Syntheseproblems, bei denen über BPOs hinaus entsprechende zusätzliche Informationsartefakte eine Rolle spielen, beispielsweise durch die Berücksichtigung entsprechender Informationen über Zustände eines Prozesses, oder es sind händische Adaptionen eines synthetisierten Modells von Nöten.

Verifkation: In der Verifikations-Phase wird abschließend das erstellte Prozessmodell bzgl. der Informationsartefakte verifiziert. Es wird geprüft, ob das Modell die Anforderungen korrekt widerspiegelt. Dies ist normalerweise trotz der Anwendung formaler Syntheseverfahren sinnvoll, da ja, wie erläutert, meist auch über Ablaufmodelle hinausgehende Anforderungsartefakte berücksichtigt werden müssen. Die formalen Informationsartefakte ermöglichen hierbei eine umfassende Anwendung automatischer Verifikationsmethoden. Es lassen sich beispielsweise formale Petrinetz-Verifikationsmethoden und Model-Checking-Verfahren verwenden. Neben dem Abgleich des Prozessmodells mit den Anforderungsmodellen ist auch eine Überprüfung des Prozessmodells bzgl. allgemeiner Korrektheitskriterien im Rahmen der Verifikations-Phase sinnvoll. Es gibt etliche Eigenschaften für Prozessmodelle, welche unabhängig von einer konkreten Spezifikation bedeutsam sind. Beispiele hierfür sind die Abwesenheit von Verklemmungssituationen, bestimmte Terminierungseigenschaften und gewisse strukturelle Eigenschaften. Viele solcher Eigenschaften können für Petrinetze automatisch mit entsprechenden formalen Methoden überprüft werden.

Informationsmanagement: Parallel zu den fünf zuvor diskutierten Phasen findet die Phase des Informationsmanagement statt. Diese stellt einerseits die Infrastruktur zum Speichern, Verknüpfen und Aktualisieren der Dokumente und Daten zur Verfügung. Hierzu muss ein geeignetes Management von Werkzeugen, Daten, Dateisystemen und Zugriffsrechten durchgeführt werden. Diese Aufgaben sind insbesondere eng mit der Wahl einer Aufnahme- und Dokumentationsmethodik in der Aufnahme-Phase und einer Formalisierungsmethodik in der Formalisierungs-Phase verknüpft. Allerdings geht es beim Informationsmanagement um die organisatorischen Aspekte im Hintergrund, beispielsweise das Bereitstellen der Werkzeuge, das Anlegen und Verwalten der Datenbanken, das Verknüpfen und physische Strukturieren der verschiedenen Daten und das Bereitstellen von geeigneten Zugriffsmöglichkeiten auf Datenbanken, Daten und Werkzeuge typischerweise im Rahmen einer Einbettung in das Intranet eines Unternehmens.
Andererseits dient die Phase des Informationsmanagement auch der Überwachung des Fortschrittes der fünf anderen Phasen. Hierzu sei angemerkt, dass diese Aufgabe an anderer Stelle auch als Teil der Validierung betrachtet wird [187]. Wir schlagen hierzu die Verwendung einer umfassenden To-Do-Liste vor. Jeder identifizierte Informationsbedarf stellt eine Zeile einer Tabelle dar. Die Modellierungsaktivitäten, welche für eine Information entsprechend des diskutierten Modellierungsansatzes durchgeführt werden sollen, werden in den Spalten der Tabelle repräsentiert. Die Zeileneinträge ergeben sich direkt aus dem Erhebungsplan. Die Spalten sollten nicht nur die Haupt-Aktivitäten

Aufnahme, Formalisieren, Validieren, Integration und Verifikation enthalten, sondern hier ist eine genauere Aufteilung in möglichst feingranulare Aufgaben wichtig. Für bestimmte Aufgaben ist es darüber hinaus oft hilfreich, die Haupt-To-Do-Liste durch ergänzende detaillierte Check-Listen zu ergänzen [187]. Wichtig ist, dass bei der Verwendung einer To-Do-Liste nicht nur quantitative Aspekte im Vordergrund stehen dürfen. Dies führt typischerweise zu einer oberflächlichen Bearbeitung der Aktivitäten, so dass sie abgehakt werden können. Es ist also wichtig, dass bei einer solchen Liste auch die Qualität der Ausführung und der Ergebnisse einer Modellierungsaktivität durch entsprechende Qualitätskriterien zum Tragen kommt.
Damit ist die Darstellung unseres neuartigen Modellierungsansatzes für Geschäftsprozesse abgeschlossen. Die wesentliche Neuerung ist die Verwendung des streng ablauforientierten Modellierungsvorgehens zusammen mit einer systematischen Erhebung, Formalisierung und Dokumentation der Anforderungen an das Prozessmodell. Der wichtigste Vorteil dieses Ansatzes ist eine bestmögliche Sicherstellung der Validität der erzeugten Geschäftsprozessmodelle. Die in dieser Hinsicht positiven Effekte des streng ablauforientierten Modellierungsansatzes wurden im wesentlichen schon zu Beginn des Kapitels erläutert. Entscheidend ist an dieser Stelle insbesondere, dass eine formale Modellierung der Abläufe nicht nur eine automatische Integration der Abläufe erlaubt, sondern auch eine zuverlässige Validierung der gesammelten Informationen und eine formale Verifikation des erstellten Modells ermöglicht. Die beschriebene systematische Vorgehensweise zur Gewinnung der Anforderungen trägt dabei dem Problem, dass gerade in großen Unternehmen die Informationen typischerweise in einer hochgradig verteilten, unstrukturierten und informalen Form vorliegen, Rechnung. Die konsequente Dokumentation der Anforderungen hat noch einen weiteren positiven Nebeneffekt. Die Anforderungen können jeder Zeit wieder eingesehen und zurückverfolgt werden. Damit können sie beispielsweise wiederverwendet werden, wenn der Geschäftsprozess später einmal geändert oder erweitert wird.
Der wesentliche Nachteil des beschriebenen Ansatzes ist natürlich der relativ hohe Zeit- und Ressourcenaufwand insbesondere im Rahmen der frühen Modellierungsphasen und die damit verbundenen Kosten. An dieser Stelle muss projektspezifisch sicherlich ein Trade-Off zwischen dem notwendigen Aufwand und den zugehörigen Kosten gefunden werden. Allerdings haben wir vom Softwareengineering gelernt, dass sich entsprechende Anstrengungen in frühen Modellierungsphasen zumindest bei wichtigen Projekten normalerweise auszahlen.
Zum Abschluss dieses Unterabschnittes kommen wir nun noch einmal auf unser Industrieprojekt mit AUDI zurück. Wir haben den beschriebenen Modellierungsansatz im Rahmen der Erstellung des sog. Bedarfs-Kapazitäts-Management-Prozesses des Beschaffungsbereiches von AUDI prototypisch durchgeführt. Der Ansatz ließ sich gut umsetzen und die Ergebnisse waren sehr vielversprechend. Wir haben umfassende, detaillierte Informationen über den Prozess gewonnen und diese bei der Erstellung eines Prozessmodells genutzt. Das resultierende Modell hat den wahren Geschäftsprozess sehr präzise widergespiegelt. Allerdings sind wir auch auf einige praktische Schwierigkeiten gestoßen. So gab es rechtliche Einschränkungen, beispielsweise hinsichtlich der Speicherung der Namen von Informationslieferanten, und organisatorische Probleme, beispielsweise mangelnde Verfügbarkeit von Informationslie-

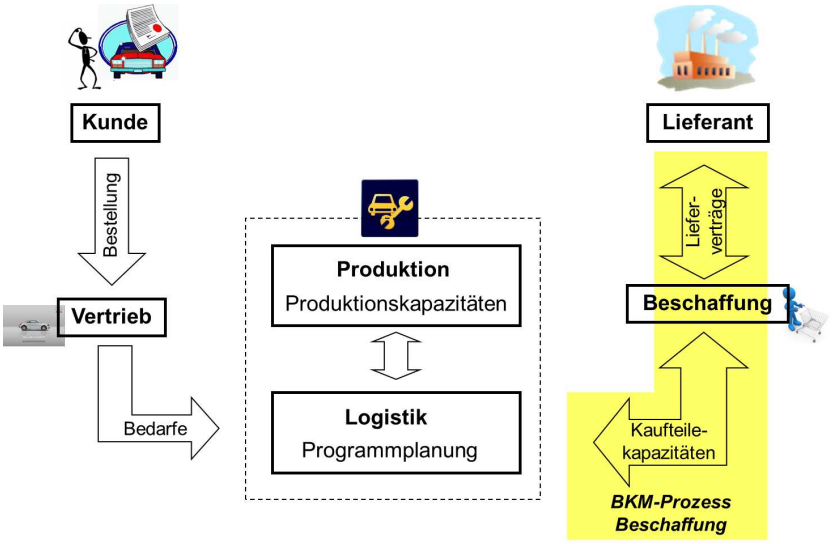

Abbildung 25: Einordnung des BKM-Prozesses.

feranten. Eine über unser Industrieprojekt hinausgehende umfassende Evaluation des Modellierungsansatzes steht allerdings noch aus.
Zur besseren Illustration des Modellierungsansatzes demonstrieren wir schließlich noch einige der wesentlichen Schritte des Ansatzes mit Beispielen aus dem Industrieprojekt. Der dort betrachtete Bedarfs-Kapazitäts-Management-Prozess (BKM-Prozess) befasst sich grob gesagt damit, die Kapazitäten der von Lieferanten eingekauften Teile an den Bedarf in der Produktion anzupassen. Dies ist sehr wichtig, da ja ein Großteil der in die Produktion eingehenden Teile nicht von AUDI selbst, sondern von Zulieferern produziert wird. Der Einkauf muss hier vor allem dafür sorgen, dass die Lieferanten zu jeder Zeit genug Teile liefern (können), um den Bedarf für die Produktion zu decken.
Die Ausgangssituation im Rahmen der Modellierung von Prozessen im Einkauf von AUDI war für uns tatsächlich eine große „grüne Wiese“, d.h. es war den Verantwortlichen relativ unklar, wie die Prozesse, auch der BKM-Prozess, im Detail eigentlich ablaufen. Wir haben daher den vorgestellten Modellierungsansatz im Rahmen dieses Industrie-Projektes entwickelt und dann, mit dem Ziel ein valides Modell des BKM-Prozesses zu erstellen, angewendet.
Der erste Schritt zur Modellierung des BKM-Prozesses war eine Festlegung des Rahmens der Modellierungsaufgabe. Wichtig sind hier die Ziele der Modellierung. Wir haben in diesem Abschnitt schon etliche typische Ziele im Rahmen der Modellierung von Geschäftsprozessen angesprochen. Im Falle des BKM-Prozesses von AUDI gab es aber noch einige interessante spezielle Ziele. Wichtig ist bei AUDI generell eine gute Dokumentation der Prozesse, da eine solche, wie schon erwähnt, eine Bedeutung für die TÜV-Zertifizierung von AUDI hat. Aber auch eine zunehmende Prozessorientierung bei AUDI und eine hohe Fluktuation der Mitarbeiter im Einkaufsbereich machten die reine Dokumentation des Prozesses zum wichtigsten Modellierungsziel. Daneben war es aber als zweites zentrales Ziel generell auch bedeutsam, für eine gute Beherrschung und Transparenz der Einkaufsprozesse zu sorgen. So soll eine hohe Flexibilität trotz der Komplexität der meist bereichsübergreifenden Prozesse und der heterogenen Systemlandschaft des

Einkaufsbereichs erzielt werden. Den BKM-Prozess betreffend sollte insbesondere das Kapazitätsmanagement für den gesamten VW-Konzern vereinheitlicht werden und eine neue Kapazitätsdatenbank eingeführt werden. Analysen und Optimierungen der Prozesse spielten dabei natürlich auch eine, allerdings eher untergeordnete, Rolle.

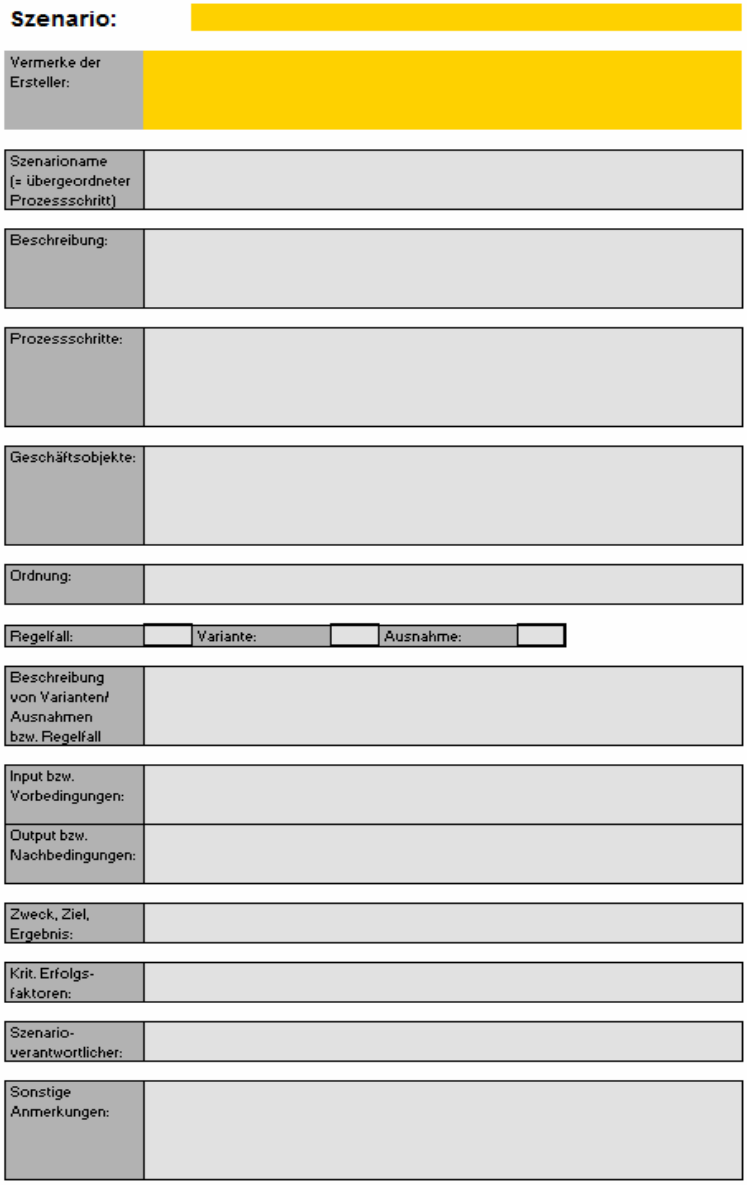

Abbildung 26: Szenariofragebogen.

Der erste Modellierungsschritt ist nun die Abgrenzung des BKM-Prozesses zu seiner Umgebung. Eine Skizze hierzu ist in Abbildung 25 dargestellt. Im Rahmen der Erstellung eines ersten High-Level-Modells haben wir insbesondere verschiedene wichtige Teile des Prozesses identifiziert. So beginnt der Prozess beispielsweise an der Schnittstelle Logistik/Beschaffung. Hier wird u.a. in einem speziellen Verfahren die Soll-Kapazität ermittelt und weitergeleitet. Dann gibt es drei wesentliche zu unterscheidende Prozessteile: die Erhöhung der Kapazität eines Einzelteils, z.B. eine besonders beliebte Sonderausstattung, die

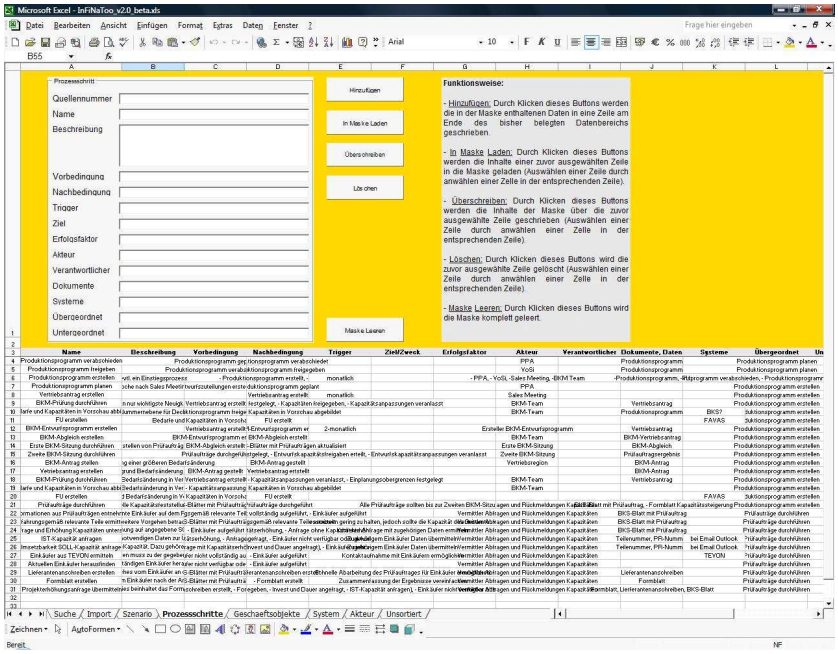

Abbildung 27: Screenshot: Dokumentation der Anforderungen an den BKM-Prozess.

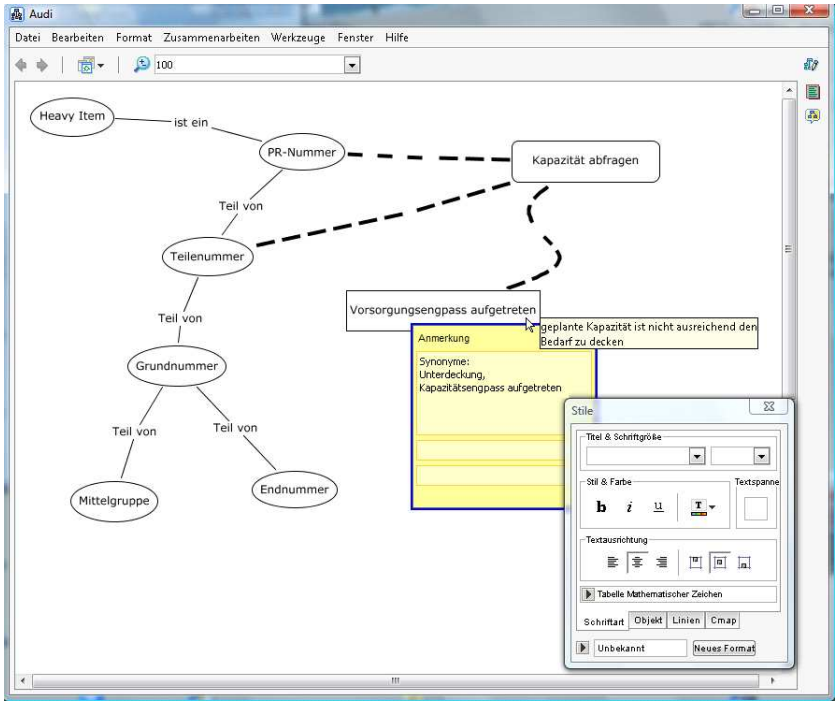

Abbildung 28: Ausschnitt des Thesaurus des BKM-Prozesses.

Erhöhung der Kapazität eines Projektes, z.B. bei Einführung eines neuen Modells, oder die Abfrage der aktuellen Maximalkapazität zu Planungszwecken.
Weiter haben wir dann alle Schritte unseres Ansatzes, so wie in diesem Unterabschnitt beschrieben, durchgeführt. Auf Basis des High-Level-Modells haben wir den Informationsbedarf und zugehörige Informationsquellen identifiziert und in einem Informationssammelplan verknüpft (wir haben auch To-Do-Listen für jede als Bedarf identifizierte Information erstellt). Dann haben wir die vorgeschlagenen Erhebungs- und Dokumentationsmethoden, weitgehend wie beschrieben, angewendet. Wir haben Dokumente analysiert und Interviews entsprechend unseres Szenario-orientierten Leitfadens durchgeführt. Der Fokus auf Beispiel-Szenarien hat sich hier als sehr fruchtbar und effektiv erwiesen, da die Denkweise in Beispiel-Szenarien für die meisten Informationslieferanten sehr natürlich und intuitiv war. Wir haben auf diesem Wege viele und sehr präzise Informationen von den Befragten erhalten. Ein Beispiel für einen bei den Interviews verwendeten Szenariofragebogen findet sich in Abbildung 26. Die gewonnenen Informationen haben wir gefiltert und anschließend strukturiert gespeichert. Abbildung 27 zeigt einen Screenshot unserer mit Microsoft Excel realisierten prototypischen Umsetzung der vorgeschlagenen Dokumentationsmethode. Einen zugehörigen Thesaurus haben wir mithilfe des Werkzeugs CMaps (http://cmap.ihmc.us) erstellt. Abbildung 28 zeigt einen Ausschnitt des Thesaurus.
Wir haben insgesamt sehr umfangreiche Informationen über den BKM-Prozess gewonnen. Die Szenario-Informationen haben wir nun entsprechend der vorgeschlagenen Formalisierungsmethodik in BPOs übersetzt. Ein Beispiel eines entsprechenden Ablaufs für einen Teilprozess des BKM-Prozesses ist in Abbildung 29 dargestellt. Wir haben 17 solcher Ablaufmodelle erstellt. Diese haben wir dann validiert. Zuerst haben wir selbst einige Analysen durchgeführt und die Modelle dann noch einmal bei den Informationslieferanten, wie vorgeschlagen, rückgekoppelt. Aus diesem Schritt resultierten einige kleinere Anpassungen der Modelle. Wir erzielten jeweils einen sehr guten Konsens, dass die formalen Ablaufmodelle dann genau das gewünschte Verhalten darstellten. Anschließend haben wir die einzelnen Szenarien mithilfe von Syntheseverfahren in ein Petrinetz integriert. Allerdings haben wir Syntheseverfahren nur auf der Ebene von Teilprozessen angewandt und die verschiedenen Teilprozesse dann händisch zusammengefügt. Das resultierende Petrinetzmodell des Gesamtprozesses haben wir daraufhin noch mit Petrinetz-Verifikationsverfahren getestet, wobei sich hier keine nennenswerten Probleme ergaben. Danach haben wir das Petrinetz entsprechend den Vorgaben von AUDI in eine EPK übersetzt. Dabei haben wir die Ereignis-Namen wiederum per Hand editiert. Abbildung 30 zeigt einen Ausschnitt des resultierenden Prozessmodells. Insgesamt umfasst das Modell 41 Aktivitäten. Eine abschließende Validierung der EPK hat bestätigt, dass das Geschäftsprozessmodell den realen Prozess getreu widerspiegelt und auch die Granularität der Modellierung angemessen ist. Der in diesem Unterabschnitt vorgestellte Modellierungsansatz ließ sich somit insgesamt im Rahmen dieser aufwändigen realen Modellierungsaufgabe sehr erfolgreich anwenden.

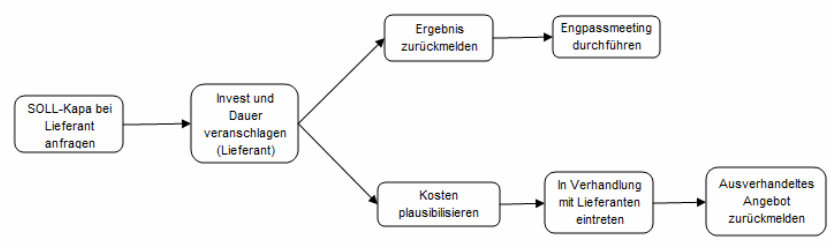

Abbildung 29: Ablauf des BKM-Prozesses.

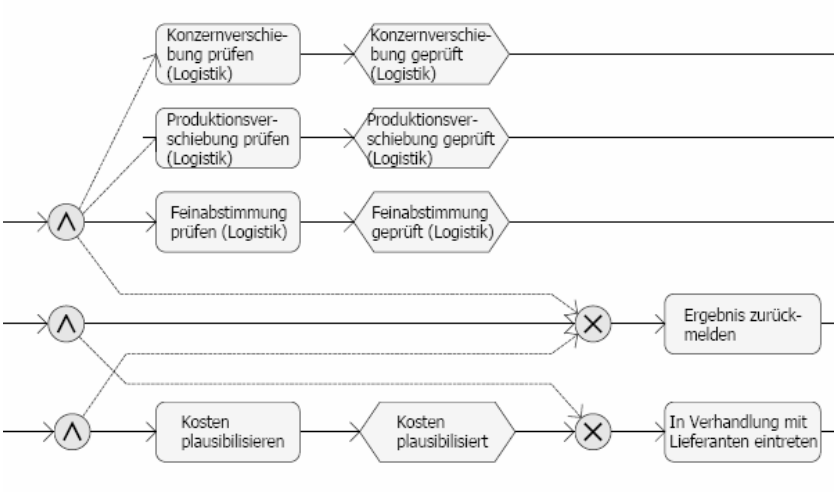

Abbildung 30: Ausschnitt des BKM-Prozesses als EPK.

2.2.2 *Process-Mining*

Im letzten Unterabschnitt haben wir ausführlich einen ablauforientierten Ansatz zur Modellierung von Geschäftsprozessen eingeführt. Implizit wurde Ablauforientierung, sogar strenge Ablauforientierung, zur Modellierung von Geschäftsprozessen allerdings schon vorher in einem sehr speziellen Kontext verwendet. Viele in der Praxis vorkommende Informationssysteme zeichnen alle durchgeführten Aktionen in speziellen Protokoll-Dateien, sog. Ereignis-Logs, auf. Ereignis-Logs werden nicht nur von klassischen Workflow-Systemen und ERP-Systemen, sondern beispielsweise auch von Web Services, Middleware-Systemen und eingebetteten Systemen von High-Tech-Ausstattung wie medizinischen Systemen, erzeugt [3, 1]. Sie können genutzt werden, um den tatsächlichen Workflow des Systems zu untersuchen. Techniken, welche sich mit der Gewinnung strukturierter Informationen aus den umfangreichen Daten von Ereignis-Logs beschäftigen, werden unter dem Begriff des Process-Mining subsumiert [3, 1]. Die verbreitetste Aufgabe des Process-Mining ist das sog. Process-Discovery. Hierbei wird versucht, ein Prozessmodell des Workflows des beobachteten Systems aus einem Ereignis-Log zu erstellen.
Den Hintergrund zum Ansatz des Process-Discovery bildet die folgende Herangehensweise. Wie in diesem Abschnitt diskutiert, ist die Erstellung valider Geschäftsprozessmodelle in der Praxis sehr schwierig. Der letzte Unterabschnitt zeigt, dass Modellierungsansätze, welche sich

intensiv mit diesem Thema auseinandersetzen, kompliziert und aufwändig sind. Das Problem bei der IST-Prozessmodellierung liegt vor allem darin, herauszufinden, wie sich der zu modellierende Geschäftsprozess in der Realität eigentlich verhält. In der betrieblichen Praxis ist dies erstaunlicherweise häufig auch bei zentralen Unternehmensprozessen völlig unklar. Die entscheidende Idee des Process-Mining zur Lösung dieses Problems ist die Vorstellung, dass das Verhalten eines laufenden Prozesses oft sehr präzise in Ereignis-Logs festgehalten ist. Ein Ereignis-Log beschreibt normalerweise alle tatsächlich stattgefundenen Abläufe des Prozesses. Somit ist es bei der Modellierung eines IST-Prozesses vielfach gar nicht nötig, dessen Verhalten mit aufwändigen Mitteln zu untersuchen, sondern es lässt sich ganz einfach aus entsprechenden Log-Dateien auslesen. Die zu bewältigende Modellierungsaufgabe beschränkt sich dann auf eine Übersetzung der Ablaufinformationen aus einem Ereignis-Log in ein konsistentes Prozessmodell. Dies kann und soll automatisch durchgeführt werden. Das bietet sich in diesem Kontext an, da die Informationen eines Ereignis-Logs schon in einer automatisch weiterverarbeitbaren Form vorliegen. Die Größe und Komplexität von Ereignis-Logs lässt ein händisches Verständnis dahingegen nur schwer zu. Diese Überlegungen zeigen, dass das Process-Discovery eine spezielle, sehr interessante Form des streng ablauforientierten Modellierungsansatzes darstellt. Allerdings lässt sich Process-Discovery nur unter der speziellen Voraussetzung durchführen, dass ein IST-Prozess, dessen Abläufe in einem Ereignis-Log aufgezeichnet wurden, modelliert werden soll.

Process-Mining und im Besonderen Process-Discovery hat in den letzten Jahren ein hohes Maß an Aufmerksamkeit erlangt. Es gibt etliche Arbeiten, welche Verfahren und Herangehensweisen aus dem Bereich des Process-Mining beschreiben, siehe beispielsweise [3, 1] für einen Überblick. Es gibt auch entsprechende Process-Mining-Werkzeuge. Eine Vielzahl von akademischen Werkzeugen sind in das ProM-Framework integriert [2]. Inzwischen beinhalten aber auch etliche industrielle Werkzeuge Process-Mining-Verfahren, beispielsweise [203]. In der Literatur werden einige industrielle Anwendungen von Process-Mining beschrieben, z.B. [198, 161].

Für genauere Einblicke ist es wichtig, wie ein Ereignis-Log eigentlich aussieht. Eine Ausführung eines Workflows entspricht einem Fall bzw. einer Prozessinstanz des Workflows. Jeder Fall setzt sich aus verschiedenen einzelnen Arbeitsschritten zusammen. Ein typisches Ereignis-Log zeichnet nun alle im Rahmen eines Workflows durchgeführten Aktivitäten jeweils zusammen mit dem zugehörigen Fall auf. Die aufgezeichneten Aktivitäten sind nach ihrer Ausführungszeit geordnet. Somit legt jeder in einem Ereignis-Log enthaltene Fall eine Folge von Aktivitäten fest. Jede dieser Folgen entspricht einer tatsächlichen Ausführung des betrachteten Workflows. Unter der Annahme, dass die einzelnen Fälle des Workflows unabhängig voneinander sind, beschreiben die Aktivitätsfolgen einzelne Abläufe des Workflows. Zusätzlich enthält ein Ereignis-Log meist noch etliche weitere Informationen wie zu Aktivitäten gehörige Akteure. Da wir hier nur an dem Kontrollfluss eines Workflows interessiert sind, abstrahieren wir an dieser Stelle allerdings von derartigen Informationen. Die folgende Tabelle zeigt ein Beispiel für eine entsprechende Log-Datei.

Aktion	a	b	a	b	a	d	a	c
Fall	1	1	2	1	3	3	4	3
Zeit	7:12	7:15	7:16	7:17	7:20	7:25	7:25	7:26
Aktion	c	d	e	b	e	e	b	e
Fall	2	2	1	4	3	2	4	4
Zeit	7:35	7:36	7:39	7:44	7:45	7:46	7:47	7:50

Die im Sinne des letzten Abschnittes von diesem Log definierten Aktivitätsfolgen sind $abbe$ (Fall 1 und 4), $acde$ (Fall 2) und $adce$ (Fall 3).
Aus einem Ereignis-Log lässt sich somit eine Menge von Abläufen des aufgezeichneten Workflows in der Form von Folgen von Aktivitäten extrahieren. Die Aufgabe des Process-Discovery ist die Erstellung eines Prozessmodells aus den in einem Ereignis-Log beschriebenen Abläufen. Es ergibt sich also in natürlicherweise eine Synthese-Fragestellung. Wie schon dargestellt, sind Petrinetze eine beliebte formale Modellierungssprache zur Darstellung von Geschäftsprozessmodellen. In diesem Kontext bietet sich dann der Einsatz von Petrinetz-Syntheseverfahren im Rahmen des Process-Discovery an. Es ist dabei allerdings zu beachten, dass das Problem des Process-Discovery einige Besonderheiten aufweist. Die wichtigsten Unterschiede zum klassischen Syntheseproblem, welche in etlichen Arbeiten, z.B. [1], betont werden, liegen darin,

- dass ein Modell gesucht wird, welches nicht notwendigerweise das spezifizierte Verhalten exakt widerspiegelt, sondern von dem nur verlangt wird, dass es die Abläufe aus dem Ereignis-Log aufweist und nicht viel mehr Verhalten zulässt,
- dass es aufgrund der häufig sehr großen Ereignis-Logs beim Process-Discovery besonders wichtig ist, dass die erzeugten Modelle nicht zu komplex sind,
- und dass schließlich ebenfalls aufgrund der Größe von Logs die Performance von Process-Discovery-Verfahren eine besonders wichtige Rolle spielt.

Während der zweite und dritte Punkt klar sind, bedarf der erste Punkt noch einigen Erläuterungen. Die grundsätzliche Idee hier ist, dass in einem Log nur einige zufällige Beispiel-Abläufe eines Prozesses aufgezeichnet sind und daher beim Mining eine Verallgemeinerung stattfinden soll, d.h. ein erzeugtes Modell soll auch Abläufe erlauben, welche sich nicht im Log wiederfinden. Natürlich soll dabei aber die Präzision des Modells nicht verloren gehen, d.h. es dürfen nicht zu viele zusätzliche Abläufe zugelassen werden. Allerdings wird vielfach ein gewisser Verlust an Präzision zu Gunsten des zweiten und dritten der aufgelisteten Punkte akzeptiert. Schließlich bleibt noch anzumerken, dass selbst die Forderung, dass zumindest alle Abläufe des Logs in einem zugehörigen Modell möglich sind, aufgrund von Störeinflüssen bei der Aufzeichnung von Log-Dateien nicht immer sinnvoll ist. Allerdings ist dies ein Problem, welches wir an dieser Stelle nicht näher betrachten wollen, da es für alle Mining-Ansätze in ähnlicher Form auftritt und es im Rahmen entsprechender Filterungsverfahren für Log-Dateien gute Lösungsansätze für das Problem gibt.
Entsprechend diesen Überlegungen wurden auf Petrinetz-Syntheseverfahren basierende Process-Discovery-Ansätze entwickelt [84, 7], neu-

erdings auch [222, 69, 51, 211, 53]. Allerdings wurden zunächst nur Syntheseverfahren aus dem in der Literatur gut untersuchten Bereich der Synthese von Petrinetzen aus Transitionssystemen zu Rate gezogen. Das Problem dabei ist, dass Event-Logs normalerweise keine Zustandsinformationen beinhalten und erst recht keine Transitionssysteme beschreiben. Daher lassen sich Verfahren zur Synthese eines Petrinetzes aus einem Transitionssystem nicht unmittelbar zum Process-Mining verwenden. Diese Unstimmigkeit wird dadurch überbrückt, dass künstliche Zustände zu dem Ereignis-Log hinzugefügt werden. Auf diese Weise wird ein Transitionssystem festgelegt. Dann werden Syntheseverfahren angewandt, die das Transitionssystem in ein Petrinetz übersetzen, welches das Verhalten des Transitionssystems widerspiegelt. Problematisch ist bei diesem Vorgehen zum einen, dass die entsprechenden Syntheseverfahren die Zustandsstruktur des Transitionssystems reproduzieren, obwohl die künstlichen Zustände nicht in dem Ereignis-Log spezifiziert sind. Dies führt beim Process-Discovery dann vielfach zu einer Verzerrung oder einem systematischen Fehler bei dem erzeugten Netz. Zum anderen wird eigentlich unnötiger Aufwand in die Erzeugung und Berücksichtigung der künstlichen Zustände gesteckt.
Da ein Ereignis-Log unmittelbar eine Sprache definiert, scheint die Anwendung von Ansätzen zur Synthese von Petrinetzen aus Sprachen im Rahmen des Process-Discovery wesentlich natürlicher und geeigneter. Ein Ereignis-Log definiert, wie schon erläutert, eine Menge von Abläufen. In der diskutierten Grundform eines Ereignis-Logs ist jeder Ablauf durch eine Folge von Aktivitäten gegeben und lässt sich somit als Wort über der Menge der Aktivitäten auffassen. Dementsprechend legt ein Ereignis-Log eine klassische sequentielle Sprache fest. Mit dieser Sichtweise lassen sich Verfahren zur Synthese von Petrinetzen aus klassischen Sprachen, wie sie in [13, 63, 18] dargestellt sind, im Prinzip problemlos direkt auf Ereignis-Logs anwenden. Bei diesem zwar naheliegenden aber dennoch innovativen Vorschlag ist es allerdings noch wichtig, die in obigen drei Punkten dargestellten Spezifika des Process-Discovery zu berücksichtigen. Dies erfordert eine geeignete Anpassung der Syntheseverfahren.
Entsprechend den Überlegungen des letzten Absatzes haben wir ein auf den bekannten Verfahren der Synthese von Petrinetzen aus klassischen Sprachen basierendes prototypisches Process-Discovery-Werkzeug entwickelt. Für dieses Werkzeug haben wir etliche Testreihen und Vergleiche mit anderen Werkzeugen durchgeführt. Die Ergebnisse dieser experimentellen Tests sind sehr vielversprechend. Details hierzu finden sich in [25]. Für obiges Beispiel-Log erzeugt das Werkzeug das in Abbildung 31 dargestellte Netz.
Wir wollen nun zu dem eigentlichen Schwerpunkt dieser Arbeit, nämlich der Synthese von Petrinetzen aus halbgeordneten Abläufen, zurückkommen. Bei den bisherigen Überlegungen zum Process-Discovery spielen Verfahren zur Synthese von Petrinetzen aus partiellen Sprachen noch keine Rolle, sondern es standen Verfahren zur Synthese aus klassischen Sprachen im Vordergrund. Eine klassische Sprache ist zwar ein Spezialfall einer partiellen Sprache, so dass auch in dem bisher diskutierten Kontext eine Verwendung der Verfahren zur Synthese aus partiellen Sprachen möglich ist. Diese weisen in diesem Rahmen aber typischerweise eine etwas schlechtere Performance auf. Sinnvoll ist deren Anwendung allerdings, wenn ein Ereignis-Log gewisse Informa-

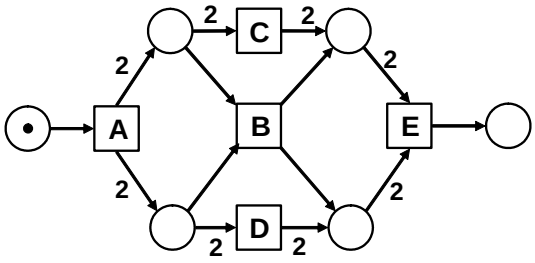

Abbildung 31: Durch Mining erzeugtes Netz.

tionen über Unabhängigkeiten von Ereignissen beinhaltet. Solche lassen sich dann berücksichtigen, indem anstelle der sequentiellen Abläufe eines Ereignis-Logs entsprechende halbgeordnete Abläufe betrachtet werden. Dies ist tatsächlich auch vielfach möglich und sollte dann auch ausgenutzt werden. Beispielsweise lässt sich auf Unabhängigkeiten von Ereignissen schließen, wenn Aktionen eines Systems explizit als unabhängig spezifiziert sind, wenn Ereignisse von unterschiedlichen verteilten Komponenten eines Systems aufgezeichnet werden, wenn sich die Ausführungszeiten von Aktionen überlappen oder auch rein ausgehend von den aufgezeichneten Ereignisreihenfolgen, d.h. wenn zwei Aktionen z.B. sehr häufig direkt hintereinander jeweils sowohl in der einen als auch in der anderen Reihenfolge stattfinden. Solche Prinzipien werden von einigen industriellen Werkzeugen verwendet, beispielsweise ARIS PPM (IDS Scheer, [203]), Staffware SPM und InConcert (TIBCO). Diese Werkzeuge erzeugen also Ereignis-Logs, welche direkt eine Menge von BPOs und somit eine partielle Sprache beschreiben. Das beste Beispiel hierfür ist vielleicht das Werkzeug ARIS PPM, welches aus den schon erwähnten Instanz-EPKs bestehende Protokolldateien generiert. Die Instanz-EPKs werden bei diesem Werkzeug dann zur Berechnung von Performance-Charakteristika verwendet. ARIS-PPM enthält auch schon ein Verfahren zum Zusammenfügen von Instanz-EPKs zu einem Prozessmodell. Abbildung 32 zeigt zwei Screenshot entsprechender Logs aus Instanz-EPKs (übernommen aus [83]). Neben der Verwendung von Werkzeugen wie ARIS PPM, welche unmittelbar halbgeordnete Ereignis-Logs zur Verfügung stellen, ist es natürlich auch möglich, Halbordnungen durch entsprechende Analysen „roher“ sequentieller Ereignis-Logs zu generieren.

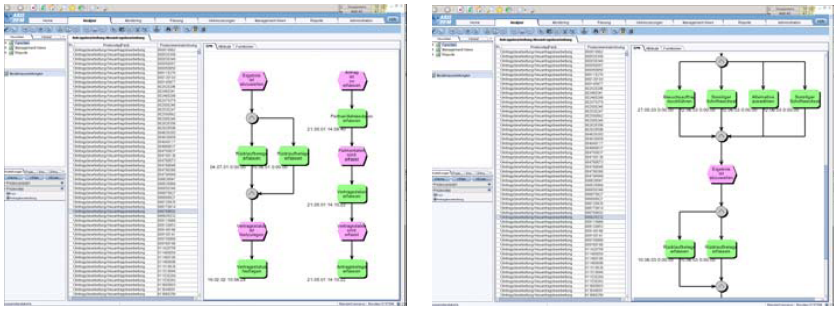

Abbildung 32: ARIS PPM.

In ProM (`prom.win.tue.nl/tools/prom`) ist ein sehr interessanter Ansatz, der sog. Partial-Order-Generator, integriert, welcher ausgehend

von den aufgezeichneten Ereignisreihenfolgen einer sequentiellen Log-Datei in sinnvollerweise halbgeordnete Abläufe erzeugt [82]. Bei einem solchen Ansatz lassen sich Unabhängigkeiten von Ereignissen bestmöglich identifizieren und berücksichtigen. Dies ist hochgradig sinnvoll, da jeder Mining-Ansatz im Prinzip die tatsächlichen Unabhängigkeiten von Aktionen „erraten“ muss. Unmittelbar beobachten können wir ja erst einmal nur sequentielle Abläufe. Dieses „Erraten“ ist ein sehr wichtiger Schritt jedes Mining-Ansatzes und die beste Möglichkeit hierzu bietet sicherlich die Ebene einzelner Abläufe. Nur auf dieser Ebene lassen sich alle Informationen eines Logs über Unabhängigkeiten unmittelbar und adäquat abbilden, da nur in der Form einzelner Abläufe die Unabhängigkeit einzelner aufgezeichneter Ereignisse berücksichtigt werden kann. Somit stellt die Übersetzung einer Log-Datei in eine Menge halbgeordneter Abläufe einen sehr sinnvollen und auch vielfach schon tatsächlich verwendeten Ansatz dar.
In unserem zuvor betrachteten Log deutet beispielsweise die Tatsache, dass einmal c unmittelbar vor d und einmal d und dann erst c vorkommen und die Ausführungszeiten der beiden Ereignisse jeweils auch noch relativ nahe beieinander liegen, darauf hin, dass diese Ereignisse unabhängig voneinander sind. Ähnliches gilt jeweils für die zwei b-Ereignisse. Somit lässt sich die Log-Datei in die zwei in Abbildung 33 dargestellten halbgeordneten Abläufe übersetzen. Hier ist nun aber wichtig, dass es für ein anderes Log, welches auch die Aktivitätsfolgen $abbe$, $acde$ und $adce$ definiert, nicht unbedingt sinnvoll sein muss, beispielsweise c und d als nebenläufig anzunehmen. Es kann sich auch um den Fall eines wechselseitigen Ausschlusses handeln, d.h. es können c und d in beliebiger Reihenfolge ausgeführt werden, allerdings niemals parallel, da die Ausführung der Aktionen jeweils einen Eintritt in einen entsprechenden kritischen Bereich erfordert. Ein Process-Mining-Algorithmus, welcher nur die drei Aktivitätsfolgen $abbe$, $acde$ und $adce$ betrachtet, wird in beiden Fällen, also sowohl im Falle eines wechselseitigen Ausschlusses von c und d als auch bei Parallelität von c und d, dasselbe Prozessmodell erzeugen. Dies ist dann aber für einen der Fälle ungeeignet. Durch die Berücksichtigung halbgeordneter Abläufe beim Process-Mining lassen sich solche Fälle dahingegen sehr genau unterscheiden.

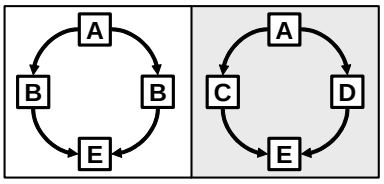

Abbildung 33: Zum Beispiel-Log gehörige partielle Sprache.

Insgesamt ist es, wie gezeigt, in vielen Fällen sinnvoll, ein Ereignis-Log in eine partielle Sprache zu übersetzen, um Informationen über Nebenläufigkeiten abbilden zu können. Außerdem gibt es ohnehin schon etliche industrielle Ereignis-Logs, welche in der Form einer partiellen Sprache vorliegen. Um die Informationen über Nebenläufigkeiten dann aber auch bei der Übersetzung in ein Prozessmodell nutzen zu können, sind Mining-Verfahren für partielle Sprachen von Nöten. In dieser Situation bietet sich dann natürlich die Verwendung von Verfahren zur Synthese von Petrinetzen aus halbgeordneten Abläufen an. Es ergibt

sich also nun auch im Bereich des Process-Mining wiederum gerade die Synthesefragestellung dieser Arbeit.
Im Falle unseres Beispiel-Logs lässt sich die zugehörige partielle Sprache (Abbildung 33) mit einem entsprechenden Syntheseverfahren in das Prozessmodell aus Abbildung 31 übersetzen. Andere Mining-Ansätze, welche Nebenläufigkeit nicht explizit berücksichtigen, könnten für dieses Event-Log u.U. eine hier unerwünschte Stelle, welche eine geteilte Ressource für die Transitionen c und d modelliert und dadurch Nebenläufigkeit verhindert, einfügen. Eine solche Stelle ist dahingegen im Falle des in der ProM-Distribution enthaltenen Beispiel-Logs pn_ex_02.xml, welches einen wechselseitigen Ausschluss zweier Teilprozesse aufgezeichnet hat, gerade erwünscht. Wenden wir den Partial-Order-Generator von ProM auf dieses Beispiel an, so entstehen ausschließlich totale Ordnungen, d.h. das Besondere an diesem Beispiel ist, dass in diesem Prozess gerade keine Nebenläufigkeit vorkommt. Mit einem Syntheseverfahren für halbgeordnete Abläufe wird diese Anforderung in einer natürlichen Weise berücksichtigt. Wenden wir ein solches Syntheseverfahren auf die entsprechenden totalen Ordnungen an, so entsteht beispielsweise das in Abbildung 34 illustrierte Netz, welches aufgrund der in der Mitte dargestellten geteilten Ressourcen-Stelle keine Nebenläufigkeit ermöglicht. Mining-Verfahren, welche kein halbgeordnetes Verhalten berücksichtigen, könnten für dieses Beispiel nun möglicherweise Nebenläufigkeiten zulassen, obwohl in dem betrachteten Prozess keine Nebenläufigkeiten vorliegen. Zumindest ist es a-priori normalerweise nicht klar, ob ein klassischer Mining-Ansatz Nebenläufigkeiten erzeugt oder nicht. Die Gegenüberstellung der zwei dargestellten Beispiele illustriert sehr gut, dass die Betrachtung halbgeordneter Abläufe im Rahmen des Process-Mining sinnvoll ist, um dieses Problem zu lösen.

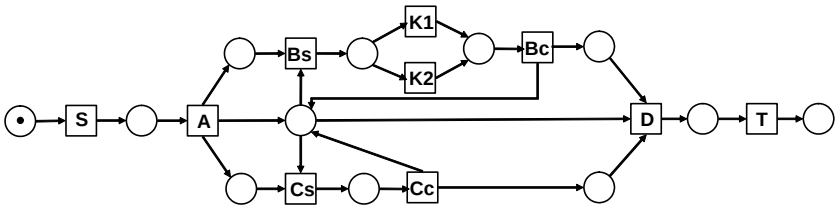

Abbildung 34: Wechselseitiger Ausschluss zweier Teilprozesse.

Zu beachten ist, dass auch in dem Kontext halbgeordneter Abläufe, wie oben schon angesprochen, eine entsprechende Berücksichtigung der Spezifika des Process-Discovery wichtig ist. Auf diesen Punkt gehen wir nun noch einmal genauer ein. Es stellt sich generell die Frage, ob eine Anpassung von Petrinetz-Syntheseverfahren für die Zielstellungen des Process-Discovery überhaupt möglich ist. Dies soll anhand der drei genannten Spezifika des Process-Discovery diskutiert werden.
Zuerst einmal ist zu betonen, dass dem hier vorgeschlagenen Process-Discovery-Ansatz präzise Syntheseverfahren zugrunde liegen, welche grob gesagt ein Netz konstruieren, das die Abläufe eines Ereignis-Logs so gut wie möglich reproduziert. Im Gegensatz dazu sind die meisten klassischen Process-Discovery-Ansätze heuristische Verfahren, welche keine exakte Fragestellung beantworten. Um einen geringen Zeit- und Speicherbedarf zu erzielen und um die resultierenden Prozessmodelle kompakt zu halten, werden bei den meisten Verfahren Prozessmodelle

erzeugt, welche wesentlich mehr Verhalten als im Ereignis-Log angegeben zulassen. Es wird hier von einer Überapproximation des Logs gesprochen. Das Problem ist, dass in vielen Fällen zu stark überapproximiert wird. Insbesondere findet meist eine Überapproximation statt, ohne dass es in dem Log Hinweise auf entsprechend zusätzliches Verhalten des beobachteten Prozesses gibt. Es wird in diesem negativen Sinne von Underfitting gesprochen. Die meisten heuristischen Process-Discovery-Verfahren weisen derartige Schwierigkeiten insbesondere bei komplexen Kontrollflussstrukturen wie unbalancierten Splits und Joins, Verletzungen der Free-Choice-Eigenschaft, eingebetteten Schleifen, usw, auf. In der Realität haben Prozesse allerdings häufig solche Eigenschaften.
Derartige Probleme gibt es bei Process-Discovery-Ansätzen, welche auf präzisen Syntheseverfahren basieren, natürlich nicht. Diese werden dahingegen für ihre Tendenz zum sog. Overfitting kritisiert [1]. Damit ist gemeint, dass zu wenig Überapproximation stattfindet. Der Grund für diese Kritik liegt darin, dass ein Ereignis-Log normalerweise unvollständig ist, d.h. nicht alle möglichen Abläufe des Prozesses sind in einem Log aufgezeichnet. Auf den ersten Blick spricht das sicherlich dafür, dass ein hohes Maß an Überapproximation sinnvoll ist. Aber ohne zusätzliche Informationen ist es unklar, wie die in dem Log fehlenden Abläufe aussehen. Es ist auch kaum möglich, Informationen über solche fehlende Abläufe allein aus dem Ereignis-Log zu gewinnen. Dementsprechend erscheint die von heuristischen Verfahren durchgeführte Überapproximation häufig willkürlich, zumindest ist sie aber meist kaum kontrollierbar und nur schwer vorhersehbar. Wird dahingegen von einem präzisen Syntheseverfahren ausgegangen, so lässt sich natürlich auch ganz gezielt eine gewisse Überapproximation in ein Prozessmodell integrieren. In einer Pre-Processing-Phase kann beispielsweise gezielt und kontrolliert zusätzliche Verhalten zu einem Log ergänzt werden. Wichtig bei einer solchen Pre-Processing-Phase ist ein gewisses Maß an Nutzer-Interaktion. Dann können Nutzer ihr Zusatzwissen und ihre Erfahrung einbringen, um die Informationen aus dem Log geeignet zu ergänzen. Ähnlich kann auch in einer Post-Processing-Phase das synthetisierte Netz noch einmal im Hinblick auf Überapproximation gezielt verändert werden. Ein solcher auf einem präzisen Vorgehen basierender Ansatz kann in vielen Fällen, in denen sonst ein präziser Ansatz vielleicht gar nicht angebracht wäre, ein besseres Ergebnis liefern als ein rein heuristischer Ansatz. Insbesondere ist es bei einem solchen Vorgehen klar steuerbar und nachvollziehbar, welches über das Log hinausgehende Verhalten zu dem Modell hinzugefügt wird.
Insgesamt gilt beim Process-Discovery generell, dass in verschiedensten Kontexten unterschiedliche Verfahren sinnvoll sind. Anders wäre die Vielzahl von Mining-Verfahren, welche sich in der Literatur oder auch im Werkzeug ProM finden lassen, auch gar nicht zu erklären. So hat unser präzises prototypisches Mining-Werkzeug beispielsweise für das schon angesprochene Beispiel-Log `pn_ex_02.xml` der ProM-Distribution das in Abbildung 34 dargestellte Netz, welches den wechselseitigen Ausschluss des aufgezeichneten Prozesses getreu wiedergibt, erzeugt. Dahingegen generiert der bekannte alpha-Algorithmus für dieses Beispiel eine fehlerhafte Verklemmung und auch viele andere Verfahren aus ProM scheitern an diesem Beispiel. Ein präziser Ansatz erweist sich hier also als vielversprechend (siehe [25] für weitere ähn-

liche Beispiele), in vielen anders gelagerten Beispielen, insbesondere bei hochgradig unvollständigen Logs, sind aber häufig auch gänzlich andere Mining-Ansätze vorzuziehen.
Ein weiteres zentrales, potentielles Problem von exakten Process-Discovery-Ansätzen ist die Performance. Selbst die polynomielle Laufzeit der Verfahren aus [13, 63, 18] zur Synthese von Petrinetzen aus klassischen Sprachen ist nicht unbedingt ausreichend, da viele heuristische Verfahren eine lineare Laufzeit aufweisen. Hier sind Optimierungen wichtig, um eine möglichst gute Performance zu erzielen. Die mit unserem prototypischen Process-Discovery-Werkzeug erzielten experimentellen Ergebnisse [25] zeigen, dass sich die Effizienzprobleme auf diese Weise auch für umfangreichere Logs größtenteils lösen lassen.
Schließlich ergibt sich noch das Problem, dass präzise Verfahren dazu tendieren, relativ große und unübersichtliche Prozessmodelle zu erzeugen. Dies lässt sich teilweise auch gar nicht vermeiden, da häufig eine sehr große Anzahl an Netzkomponenten notwendig ist, um ein komplexes Verhalten exakt zu reproduzieren. Dies gilt speziell für unstrukturierte Prozesse in wenig restriktiven Umgebungen. In letzterem Fall erzeugen aber auch viele heuristische Verfahren unübersichtlich große Netze. An dieser Stelle sind Syntheseverfahren wichtig, welche möglichst kompakte Netze erzeugen. Wiederum zeigen die mit unserem prototypischen Process-Discovery-Werkzeug erzielten experimentellen Ergebnisse [25], dass sich dann weitgehend zufriedenstellende Netzgrößen ergeben. Allerdings muss, wie schon angedeutet, berücksichtigt werden, dass es einen Trade-Off zwischen Netzgröße und Präzision des Netzes gibt. In vielen Fällen können auch noch Abstraktions- und Visualisierungstechniken hilfreich sein, um große Netze kompakt und übersichtlich zu repräsentieren [1].
Insgesamt gibt es viele Argumente dafür, dass die von uns vorgeschlagene Anwendung von Verfahren zur Synthese von Petrinetzen aus Sprachen im Rahmen des Process-Discovery generell ein guter in vielen Situationen nützlicher Ansatz ist. Gleiches gilt, wie erläutert, im Speziellen auch für Verfahren zur Synthese von Petrinetzen aus partiellen Sprachen. Wir haben diesen Ansatz auch in einem prototypischen Process-Discovery-Werkzeug umgesetzt, welches auf den Syntheseverfahren für klassische Sprachen aus [13, 63, 18] basiert. Es bleibt hier zuletzt noch zu erwähnen, dass unsere Grundidee, Sprach-Synthese zum Process-Discovery zu verwenden, auch von weiteren Autoren aufgegriffen und ebenfalls in vielversprechenden Process-Discovery-Werkzeugen realisiert wurde. Hierbei sind vor allem die Arbeiten [222, 69] zu nennen, aber auch in [51, 53] werden ähnliche Prinzipien angewendet.

2.3 LERNPROZESSMODELLIERUNG

Eigenaktivität des Lerners steht in modernen Lehrkonzepten (wie z.B. konstruktivistischen und selbst-regulatorischen Ansätzen) meist im Vordergrund. Dennoch gibt es Erkenntnisse, dass bei völlig freiem Lernen produktive Aktivitäten wie z.B. Reflektion und Elaboration oft nicht ausgeführt werden (können) [20]. Die Strukturierung von Lernaktivitäten durch Skripte [202, 178] und Hilfen hat sich dagegen als lernfördernd herausgestellt. In den meisten lernunterstützenden Systemen (z.B. [221]) sind diese Hilfsmechanismen fest eingebaut, beispielsweise als Bestandteile der graphischen Oberfläche, und somit nicht wiederverwendbar oder transferierbar in andere Lernkontexte

und Lernplattformen. Die explizite Repräsentation der Lernprozessmodelle und Hilfsmechanismen ist eine Möglichkeit, die pädagogische Expertise wiederverwendbar zu machen und somit den Entwicklungsaufwand für Lehr- / Lernsysteme beträchtlich zu senken sowie die pädagogischen Designprinzipien deutlicher zu machen.
Während die Bestrebungen, Lernprozessmodelle explizit zu machen, erst in den letzten Jahren unter den Begriffen Educational Modelling [190] und Learning Design [145] zu Bedeutung gelangt sind, haben sich, wie im letzten Unterabschnitt dargestellt, in den verwandten Bereichen des Geschäftsprozess-Management und des Workflow-Management bereits seit vielen Jahren eine rigorose Methodik und formale Techniken zur Modellierung von Geschäftsprozessen in Betrieben entwickelt und etabliert [6, 87, 223]. Die Grundfragestellungen sind bei der Modellierung von Lernprozessen allerdings sehr ähnlich wie bei der Modellierung von Geschäftsprozessen. Dementsprechend werden wir im Folgenden im Detail Ähnlichkeiten und Unterschiede zwischen den intensiv untersuchten Problemen der Geschäftsprozessmodellierung und den Herausforderungen in dem sich in der Anfangsphase befindlichen Educational Modeling untersuchen. Es wird sich hierbei zeigen, dass der Bereich des Educational Modeling nicht nur auf den ersten Blick Gemeinsamkeiten zum Gebiet der Geschäftsprozessmodellierung aufweist, sondern tatsächlich ganz analoge Problemstellungen aufwirft. Daher schlagen wir an dieser Stelle generell vor, Erkenntnisse und Methoden aus der Geschäftsprozessmodellierung auf Lernprozesse zu übertragen. Zu beachten ist hierbei allerdings, dass es auch einige fundamentale Unterschiede zwischen dem Educational Modeling und der Geschäftsprozessmodellierung gibt, welche bei einem Methodentransfer berücksichtigt werden müssen.
Zum Auffinden von Ähnlichkeiten zwischen dem Educational Modeling und der Geschäftsprozessmodellierung bietet es sich an, die Workflow-Referenzarchitektur aus Abbildung 23 mit der Praxis im Educational Modeling zu vergleichen. Im Bereich des Educational Modeling hat sich eine zur Workflow-Referenzarchitektur ähnliche Aufteilung in Komponenten etabliert. Die formalen, meist XML-basierten Darstellungen wie z.B. IMS/LD [122], LDL [162], PALO [195] oder MoCoLADe [111] werden selten direkt auf der XML-Ebene editiert, sondern mit speziellen Editoren auf abstrahiertem Niveau erstellt. Die Beschreibungen werden durch sog. Learning Design Engines interpretiert und zur Ausführung gebracht. Die Engine steuert – je nach Ansatz – eine Web-basierte Benutzerschnittstelle für den Lerner (z.B. beim WebPlayer von IMS/LD) oder externe Lernanwendungen wie im Falle des Remote Control Ansatzes [113], bei dem die existierenden kollaborativen Lernumgebungen FreeStyler und CoolModes verwendet wurden. Einige der Ansätze setzen auf eine vollständige Realisierung von Editor, Engine und Lernumgebung, während andere Ansätze wie z.B. das Collage Werkzeug [116] oder die MoCoLADe Modellierungssprache ein Mapping der erstellten Modelle auf IMS/LD als Zielsprache vornehmen, im Prinzip also IMS/LD als Educational Assembler verwenden.
Zur Administration der Lernprozesse und zur Beobachtung können – wie im Workflow-Management – Konfigurations- und Monitoring-Werkzeuge eingesetzt werden, wobei diese bisher nur sehr grundlegende Funktionalitäten bereitstellen und sicherlich in Zukunft weiteren Entwicklungsbedarf haben.

Forschungsgruppen im Educational Design, die auch im CSCW-Bereich (Computer Supported Cooperative Work) forschen und entwickeln, schlagen zunehmend Service-orientierte Ansätze [218] für Architekturen und einzelne Komponenten vor. Beispielsweise integriert das GridCole-System [44] die frei verfügbare CopperCore-Engine (siehe `www.coppercore.org`) für IMS/LD mit als Grid-Services implementierten Lernwerkzeugen. Interoperabilitätsansätze für Workflows wie z.B. BPEL wurden deshalb auch bereits im Educational Design thematisiert [112].
Diese Überlegungen zeigen zum einen, dass im Bereich prozessgestützter Lernsysteme ein ähnliches konzeptuelles Zusammenspiel wie im Falle von Workflow-Systemen vorliegt. Zum anderen stellen die Ausführungen einen ersten Schritt hin zu einem Referenzmodell für eine technologische Umsetzung des Educational Modeling dar.
Trotz all der Gemeinsamkeiten, die sich konzeptionell zwischen Geschäftsprozessen und Lernprozessen ergeben und sich folglich in ähnlichen technischen Realisierungen niederschlagen, sind auch wesentliche prinzipielle Unterschiede festzustellen. Insbesondere schränken die Spezifika von kollaborativen Lehr- / Lernprozessen gegenüber Geschäftsprozessen eine unmittelbare Nutzung existierender Ansätze aus dem Bereich der Geschäftsprozessmodellierung (z.B. [6]) ein.

- Bei einem Geschäftsprozess steht die Durchführung des Prozesses und der damit verbundenen Aktivitäten im Vordergrund. Wichtig ist das Endprodukt bzw. das Geschäftsziel des Prozesses. Die Beteiligung der einzelnen eingebundenen Akteure spielt nur eine untergeordnete Rolle. Bei Lernprozessen ist hingegen wesentlich, dass die Lernenden einen Lernprozess durchlaufen, bei dem einzelne Aktivitäten Lerngelegenheiten bieten. Bei einem Lernprozess steht weder ein Produkt noch eine möglichst effiziente Abarbeitung der Aufgaben im Vordergrund, sondern das (vollständige) Durchlaufen des Prozesses für die Teilnehmer und die Lernerfahrung sowie der Lernerfolg der einzelnen Akteure ist wichtig. Es ergibt sich somit ein anderer Fokus, nämlich der Lerner und sein Lernerfolg, weniger die Durchführung und das Produkt des Prozesses. Die Durchführung des Prozesses (ein Fall) ist lediglich Vehikel für das individuelle und kollektive Lernen. Daher müssen die einzelnen Akteure bei der Modellierung von Lernprozessen größere Berücksichtigung finden als bei der Modellierung von Geschäftsprozessen.

- Sog. Rollen von Akteuren bestimmen bei Geschäftsprozessen, welcher Benutzer welche Aufgaben durchführen kann. Das Rollenkonzept bei der Geschäftsprozessmodellierung beruht im Allgemeinen auf der Verantwortlichkeit bzw. Kompetenz für eine bestimmte Menge von Aktivitäten, die nach anfänglicher Zuweisung von Rollen für konkrete Akteure festbleibt. Dynamische Einschränkungen der Aktivitätsbearbeitung (in etwa: derselbe Akteur, der das Angebot formuliert, soll auch den Vertrag abschließen) und spezielle Regeln zur Allokation von Akteuren zu Aktivitäten (in etwa: der Akteur mit einer geforderten Rolle, der den wenigsten weiteren Rollen zugeordnet ist, soll zugewiesen werden) sind nur mit Zusatzkonstrukten formulierbar, z.B. RBAC (Role-Based Access Control) [201]. Im Gegensatz zu diesem starren Rollenkonzept mit fest zugeordneten Rollen nehmen Lerner

innerhalb eines Lernprozesses oft sich wandelnde Rollen ein (siehe [221] mit rotierenden Rollen). In Lernprozessen werden Rollen häufig eingesetzt, um bestimmte Fertigkeiten einzuüben. Rollen können im Laufe eines Lernprozesses meist in Abhängigkeit von den bereits erfüllten Lernaufgaben dynamisch gewechselt bzw. erworben werden. Folglich ist für Lernprozesse eine Erweiterung eines statischen Rollenmodells hin zu einem dynamischen Modell, das in der Lage ist, die Lernhistorie für Rollenfestlegungen heranzuziehen, vorzunehmen.

- Einzelne Aktivitäten, gelegentlich auch der gesamte Lernprozess, können durch Gruppenarbeit, -diskussion usw. realisiert werden. In kollaborativen Ansätzen sind diese Gruppenphasen häufig von hoher Bedeutung für die Lernerfahrung. Dies erfordert insbesondere eine differenzierte Betrachtung von Nebenläufigkeit. Beim Einzellernen werden aufgrund des eindeutigen Aufmerksamkeitsfokus des Lerners unabhängige Aufgaben meist einen beliebig sequentialisierten Ablauf der Aufgaben ergeben, da die nebenläufige oder verzahnte Bearbeitung erhöhte kognitive Last beim Lerner erzeugen würde. Beim Gruppenlernen hingegen ist durch Arbeitsteilung eine nebenläufige Bearbeitung durchaus realistisch. Die Möglichkeit der geeigneten Repräsentation von Gruppen, in der Gruppe notwendigen Rollen und gegebenenfalls dynamische Bildung / Umformung von Gruppen ist somit eine weitere Anforderung an Lernprozessmodelle.

All diese Aspekte haben allerdings bei der Modellierung eines Lernprozesses eines Einzellerners wenig Relevanz. Daher wollen wir mit diesem einfachen Fall beginnen.
Sowohl Geschäftsprozesse als auch Lernprozesse werden meist mit Hilfe von Diagrammen modelliert und kommuniziert. Im Bereich des Geschäftsprozess-Management haben sich, wie im letzten Abschnitt erläutert, u.a. Petrinetze und verwandte Sprachen zur Modellierung durchgesetzt [6, 223]. Wir schlagen hier vor, auch Lernprozesse mit Petrinetzen zu repräsentieren. Wenn nur ein einzelner Lernender modelliert werden soll, spielen dabei weder Rollen noch Lerngruppen eine Rolle.
Als Beispiel für die Modellierung von Lernprozessen mit Petrinetzen betrachten wir das folgende computergestützte Lernszenario. Die einzelnen Schüler einer Schulklasse sollen unterstützt durch das Werkzeug FreeStyler [113] lernen, wie sich verschiedene Faktoren (z.B. Lichtverhältnisse, CO_2-Gehalt, ...) auf das Wachstum von Pflanzen auswirken (Lernprozess „Pflanzen“). FreeStyler stellt hierfür verschiedene Registerkarten zur Verfügung, auf denen Fragen formuliert, einfache Modelle gezeichnet oder Daten aus einem Simulationsprogramm importiert werden können [113] (siehe Abbildung 35 für einen Screenshot). Die Menge der Registerkarten ist somit in unserem Beispiel die Menge der unterstützten Aktionen. Es stehen Registerkarten mit folgenden Inhalten zur Verfügung, wobei in Klammern der Aktionsname steht, den wir der Registerkarte zuordnen wollen.

- Eine kurze schriftsprachliche Einführung (**Ei**nführung).
- Eine Zeichenfläche, auf der die Schüler ein grobes Modell, bestehend aus mit Faktoren beschrifteten Knoten, zeichnen können. Aus diesem soll hervorgehen, wie sich die Faktoren der Meinung der Schüler nach auf das Wachstum auswirken (**Pl**anung).

- Eine Zeichenfläche, auf der die Schüler ihr Modell verfeinern können. Zusätzlich stehen ihnen jetzt Beschriftungen der Kanten zur Verfügung, mit denen sie den Einfluss der Faktoren quantifizieren und danach ihr Modell in einer Simulation testen können (**Mo**dellierung).
- Textfelder, in denen die Schüler geleitet durch gezielte Fragen die Fragestellungen formulieren, die sie im Experiment beantworten wollen (**Fr**agestellung).
- Ein Koordinatensystem, in das die Schüler ihre Hypothese über den Zusammenhang eines Faktors und des Wachstums der Pflanzen als Funktion zeichnen (**Hy**pothese).
- Einen Link zu einem von zwei Simulationswerkzeugen (z.B. BioBLAST, `www.smate.wwu.edu/slibrary/BioBlast/main.html`), in dem die Schüler ein Experiment durchführen können (Experiment **E1**).
- Einen Link zu einem zweiten Simulationswerkzeug (Experiment **E2**).
- Eine Tabelle mit zielführenden Ergebnissen aus vorgefertigten Experimenten der Simulationswerkzeuge (**Da**ten).
- Ein Koordinatensystem, in das die Daten aus den Experimenten und die Daten aus der vorgefertigten Tabelle importiert werden können. Die daraus entstehenden Funktionen können mit der Hypothese verglichen werden (**An**alyse).
- Textfelder, in denen die Schüler, geleitet durch gezielte Fragen (Explanation Prompts), ihre Ergebnisse zusammenfassen (**Er**gebnisse).

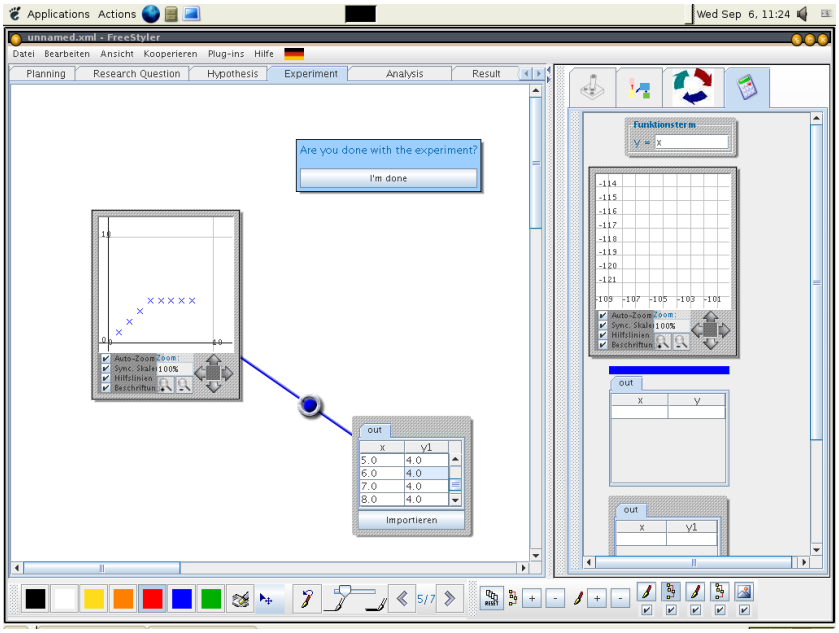

Abbildung 35: Registerkarten im Freestyler.

Ein Lernprozessmodell soll nun modellieren, in welcher Reihenfolge ein Schüler die Registerkarten bearbeiten soll. Wir betrachten den folgenden Lernprozess. Es wird mit der *Einführung* begonnen. Es folgen in beliebiger Reihenfolge die Aktionen *Planung* und *Fragestellung*. Nach der Aufgabe *Planung* kann optional auch noch die Aufgabe *Modell* durchgeführt werden, um das erstellte Modell zusätzlich zu verfeinern. Nach *Planung*, *Fragestellung* und evtl. *Modell* wird *Hypothese* durchgeführt. Dann folgen in beliebiger Reihenfolge die drei unabhängigen Aufgaben *Experiment1*, *Experiment2* und *Daten*. Sind alle drei Aufgaben ausgeführt, wird *Analyse* und schließlich *Ergebnisse* abgearbeitet.
Der beschriebene Lernprozess „Pflanzen“ ist in Abbildung 36 als Petrinetz repräsentiert. Dieses Petrinetzmodell stellt den Lernprozess sehr präzise, kompakt und übersichtlich dar. Es kann als Ausgangspunkt für vielfältige Analyse- oder Verifikations-Algorithmen dienen. Da Petrinetzmodelle per se ausführbar sind, kann man mit dem Petrinetzmodell den Lernprozess für die Schüler auch im Sinne einer Lern-Engine steuern. So könnte beispielsweise der FreeStyler für jeden Schüler, der mit dem Lernprozess „Pflanzen“ beginnt, dieses Petrinetz als Prozessinstanz erzeugen und nur Aktionen anbieten, die für den Schüler in seiner Prozessinstanz gerade ausführbar sind. Den Schülern werden die Aktionen dann nur in der durch das Modell beschriebenen Reihenfolge angeboten. Dies führt zu einem nach der Prozessspezifikation empfohlenen Lernprozess.

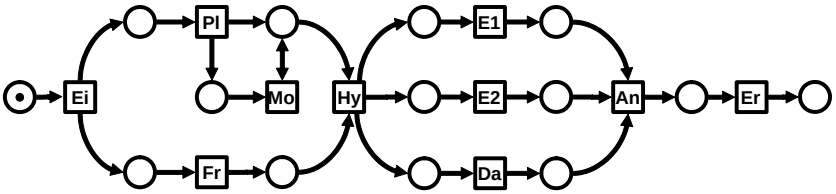

Abbildung 36: Lernprozess „Pflanzen“.

Nun stellt sich aber die Frage, wie ein Dozent überhaupt zu einem derartigen Lernprozessmodell gelangt. Das explizite Modellieren von Lernprozessen ist für den Dozenten eine unvertraute Tätigkeit, die normalerweise auf eine sequentialisierte / tabellarische Unterrichtsplanung beschränkt ist; komplexere Lernprozesse, wie z.B. Spiralansätze zur inkrementellen Vermittlung immer mehr verfeinerter Sachverhalte, sind durch solche Ansätze jedoch gar nicht darstellbar. Es kann im Allgemeinen daher nicht davon ausgegangen werden, dass ein Dozent ohne Weiteres fähig ist, ein valides Lernprozessmodell zu erzeugen. Selbst wenn ein Dozent mit Petrinetzen einigermaßen vertraut ist, so wird ihm die Modellierung komplexerer Kontrollflüsse wie beispielsweise der Optionalität der Aktivität Mo in dem Netz aus Abbildung 36 schwer fallen.
An dieser Stelle bieten sich nun wie bei der Geschäftsprozessmodellierung ablauforientierte Modellierungsansätze an. Insbesondere genügt es bei einem streng ablauforientierten Vorgehen, wenn der Dozent die gewünschten Abläufe des Lernprozesses darstellt. In unserem Beispiel beschränkt sich diese Aufgabe auf die Modellierung der folgenden zwei in Abbildung 37 dargestellten relativ einfachen Abläufe des Netzes aus Abbildung 36.

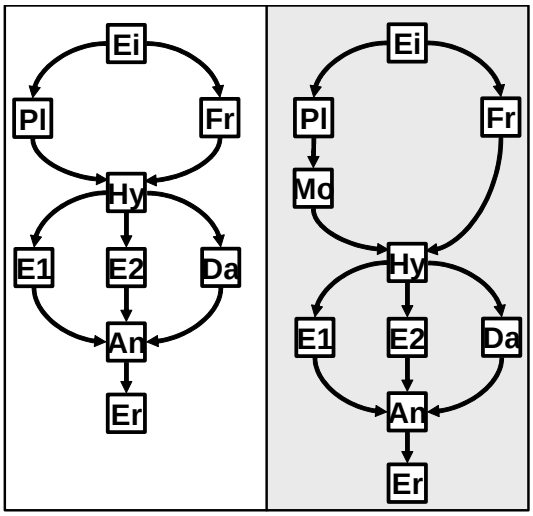

Abbildung 37: Abläufe des Lernprozesses „Pflanzen“.

Allerdings ist die explizite Definition von Lernabläufen häufig dem sog. Expert Blindspot [174] unterworfen, d.h. der Dozent legt sich auf für ihn plausible Abläufe fest, und rechnet nicht mit unkonventionellen Lösungsansätzen, die für ihn im toten Winkel liegen. Dies legt eine direkte Nutzung realer Lernabläufe zur Generierung von Lernprozessen nahe, um eine – zumindest als Grundform verwendbare – Lernprozessbeschreibung zu erhalten. Unserer Ansicht nach kann dies auch zur Verringerung des Erstellungsaufwands führen – analog zu den Erkenntnissen im Bereich der kognitiven Tutoren [8], bei denen die Erstellung allgemeiner Regelsätze durch beispielbasierte Repräsentationen ersetzt wurde.
Die Gewinnung realer Lernabläufe ist nun auf gleiche Weise möglich wie bei dem im letzten Abschnitt besprochenen Process-Discovery. Wir schlagen im Rahmen von Lernprozessen allerdings zwei verschiedene Möglichkeiten zur Gewinnung von Beispielabläufen vor, die aber beide direkt auf ein Logging / Protokollieren der Lernsysteme aufsetzen:

- Der Dozent erstellt mehrere Abläufe durch Demonstration am Lernsystem (Learnflow by Demonstration) und kategorisiert die resultierenden Ereignis-Logs bezüglich des Korrektheitsgrads oder des Potentials für Lernerfolge.
- Reale Lernabläufe der Schüler mit dem Lernsystem werden geloggt (Learnflow by Example) und mit Hilfe einer Selbsteinschätzung (im Sinne eines Collaborative Filtering Ansatzes) oder durch anschließende Dozentenbewertung (des Ergebnisses und/oder Prozesses) klassifiziert.

In beiden Fällen erhalten wir möglicherweise attributierte oder klassifizierte Ereignis-Logs, die als Eingabe für diverse Process-Discovery-Algorithmen verwendet werden können, um wie im Bereich der Geschäftsprozessmodellierung entsprechende Lernprozessmodelle zu erzeugen. Durch die direkte Nutzung realer Lernabläufe und die automatische Generierung von – zu diesen Abläufen kompatiblen – allgemeineren Modellen ist es möglich, den Dozenten beim Design von Lernszenarien zu entlasten und konkrete Lernerfahrungen unmittelbar zu nutzen.

In dem hier betrachteten einfachen Fall von Einzellernern können die aus dem Bereich der Geschäftsprozessmodellierung bekannten Process-Discovery-Verfahren völlig analog auch zur Erstellung von Lernprozessen eingesetzt werden. Dies wird kurz an unserem Beispiel-Lernszenario erläutert.
Wir nehmen an, dass jeder Schüler einer Klasse die von FreeStyler bereitgestellten Registerkarten in einer Reihenfolge bearbeitet, die ihm als sinnvoll erscheint. Jedes Bearbeiten einer Registerkarte wird vom FreeStyler als Ereignis erkannt und zusammen mit dem zugehörigen Schüler in einem Ereignis-Log aufgezeichnet. Jeder Schüler erzeugt so einen Fall. Wie beschrieben, hat der Dozent nun die Möglichkeit, aus der Menge all dieser Fälle nach bestimmten Methoden (z.B. nachträglich gemessener Lernerfolg) nur erwünschte Fälle des Lernprozesses „Pflanzen“ auszuwählen. Alternativ kann er speziell erwünschte Lernabläufe (Fälle) auch zusätzlich vorgeben. Selbst bei einem kleinen Beispiel ist die Darstellung aller möglichen Lernabläufe in einem integrierten Modell nicht trivial. Ein Ereignis-Log eines Lernprozesses kann nun aber, wie in Unterabschnitt 2.2.2 beschrieben, als Eingabe für einen der bekannten Process-Discovery-Algorithmen verwendet werden. Durch diese Algorithmen erhalten wir automatisiert ein Prozessmodell des Lernprozesses. Ein durch Mining erzeugtes Prozessmodell gibt dann das aufgezeichnete (dem Dozenten evtl. unbekannte) Lernverhalten der Schüler oder durch entsprechendes Filtern auch das erwünschte Lernverhalten der Schüler wieder. Beispielsweise könnte ein entsprechendes Ereignis-Log für den betrachteten Lernprozess „Pflanzen“ gerade die zwei halbgeordneten Abläufe aus Abbildung definieren. Mit einem geeigneten Syntheseverfahren kann daraus dann das in Abbildung 36 gezeigte Petrinetzmodell erzeugt werden. Es ist bei diesem Beispiel zu beachten, dass viele heuristische Mining-Verfahren Probleme mit der optionalen Aktivität Mo haben und somit ein exaktes Syntheseverfahren hier Vorteile aufweist. Es ergibt sich also wieder ein interessantes Anwendungsgebiet für die Synthese von Petrinetzen aus halbgeordneten Abläufen.
Für Lernprozesse von einzelnen Lernern lassen sich, wie gezeigt, generell die im Rahmen der Geschäftsprozessmodellierung diskutierten streng ablauforientierten Modellierungsansätze mit kleinen Abwandlungen in sinnvoller Weise verwenden. Schwieriger wird dies bei kollaborativen Lernprozessen. Die zu Beginn des Kapitels durchgeführten Überlegungen haben gezeigt, dass in diesem Fall ausdrucksmächtigere Mittel notwendig sind, um alle wichtigen Facetten abzubilden. Dies betrifft in erster Linie die geeignete Berücksichtigung von Rollen und Gruppen. Damit sind insbesondere andere Modellierungssprachen erforderlich. Ein entsprechender Ansatz wird nun diskutiert.
Zur Modellierung des reinen Kontrollflusses von kollaborativen Lernprozessen verwenden wir wiederum Petrinetze. Diese sollen um geeignete Rollenkonzepte erweitert werden. Bei Geschäftsprozessen wird die Allokation von Akteuren meist durch Zuweisung von benötigten Rollen zu Transitionen, welche die Aufgaben des Prozesses modellieren, realisiert. Dabei wird ein globaler Pool mit fest in Rollen eingeteilten Akteuren angenommen. Die Durchführung einer Aufgabe ist nur mithilfe eines Akteurs aus diesem Pool, welcher die zur Aufgabe zugeordnete Rolle einnimmt, möglich. Dieses Rollenkonzept soll für Lernprozesse flexibler gestaltet werden. Insbesondere sollen die Lerner mehr in den Mittelpunkt gerückt werden.

Dementsprechend werden wir das beschriebene Workflow-Rollenkonzept dadurch erweitern, dass sich die Rollen der Akteure bei der Durchführung von Aktivitäten verändern können. Die damit verbundene dynamische Rollenbelegung der Akteure werden wir explizit durch ein Zustandsdiagramm modellieren (es ließen sich an dieser Stelle auch entsprechende Petrinetze, sog. Zustandsmaschinen, verwenden, Zustandsdiagramme stellen hier aber das intuitivere Konzept dar). Ein Akteur kann seine Rolle, i.e. seinen Zustand, ändern, wenn er eine Aktivität durchführt. Rollenwechsel finden also durch Synchronisation der Übergänge der Zustandsdiagramme mit Aktivitäten des Prozessmodells statt.
Wir erweitern das Beispiel „Pflanzen“ nun zu einem kollaborativen Lernprozess. Der Lernprozess wird von einer Gruppe von drei Schülern durchlaufen, welche die Lernaufgaben nach bestimmten Regeln aufteilen dürfen. Es geht also nicht mehr nur darum, in welcher Reihenfolge die Registerkarten bearbeitet werden sollen, sondern auch, von wem sie bearbeitet werden sollen. Dies soll von den Rollen abhängen, die die Schüler innerhalb der Gruppe einnehmen.
Wir betrachten den folgenden kollaborativen Lernprozess. Alle drei Schüler beginnen gemeinsam mit der *Einführung*. Haben alle drei die Einführung gelesen, folgen unabhängig voneinander die Aktionen *Planung* und *Fragestellung*. *Fragestellung* wird von einem der drei Schüler durchgeführt. Die drei Schüler können hier aussuchen, welcher von ihnen diese Aktion übernimmt. Parallel dazu führen die beiden anderen Schüler die Aufgabe *Planung* gemeinsam aus. Nach *Planung* können die zwei Schüler, welche die *Planung* vollzogen haben, optional auch noch die Aufgabe *Modell* durchführen. Nach *Planung*, *Fragestellung* und evtl. *Modell* bearbeiten die drei Lernenden wiederum gemeinsam die Aufgabe *Hypothese*. Jeder der drei Schüler übernimmt anschließend eine der drei unabhängigen Aufgaben *Experiment1*, *Experiment2* und *Daten*. Die drei am Prozess beteiligten Lernenden teilen sich diese Aktionen auf. Sind alle drei Aufgaben ausgeführt, folgt die von allen drei Schülern kollaborativ auszuführende Aufgabe *Analyse*. Danach folgt die Aufgabe *Ergebnisse*, welche von dem Schüler, der die *Fragestellung* durchgeführt hat, zusammen mit einem der zwei an der Aufgabe *Planung* beteiligten Schülern, vollzogen werden soll. In diesem Lernszenario erfordern einige Lernaktivitäten (Ei, Hy, An jeweils mit allen drei Schülern und Pl, Mo, Pr jeweils mit zwei Schülern) bestimmte Arten von Kollaboration zwischen den Schülern. In Abbildung 38 zusammen mit Abbildung 39 ist der skizzierte kollaborative Lernprozess „Pfanzen“ mit der neuen petrinetzbasierten Modellierungssprache für Lernprozesse dargestellt.

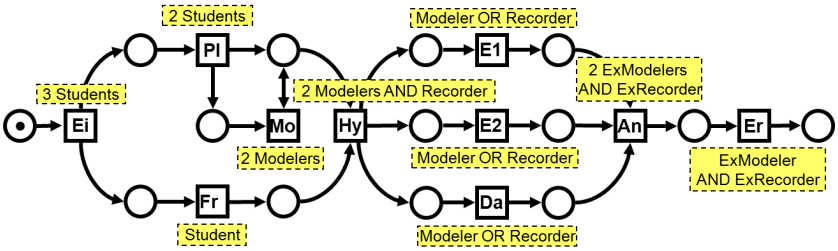

Abbildung 38: Netz mit Rollenannotationen.

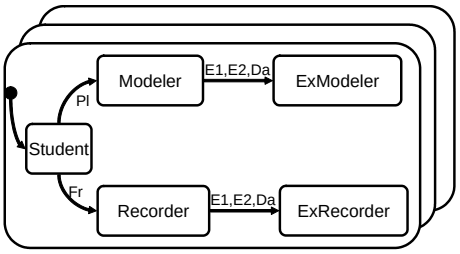

Abbildung 39: Rollendiagramme.

Das Prozessmodell in Abbildung 38 stellt den Kontrollflussaspekt des Beispiels dar. Dieser wird durch den Zustandsautomaten in Abbildung 39, der ein Rollendiagramm für die drei Lernenden repräsentiert, ergänzt. Eine Aktivität im Prozessmodell kann nur dann durchgeführt werden, falls die an der Transition angeschriebene Anzahl von Rollen im globalen Akteurspool (der drei Lernenden) vorhanden ist. Beispielsweise erfordert die kollaborative Aktivität *Planung* zwei Akteure in der Rolle Student. Beim Schalten der Transition werden für die betreffenden Akteure Rollenveränderungen vorgenommen, die im Automatenmodell einem Zustandswechsel mit dem Transitionsnamen als Eingabezeichen entsprechen; in unserem Beispiel gehen also durch die Planungsaktivität beide Schüler in die Rolle Modeler über. Somit können sie dann *Fragestellung* nicht mehr durchführen, da diese Aktivität einen Student erfordert. *Fragestellung* kann daher nur von dem dritten Lernenden ausgeführt werden.
Ähnlich verhält es sich bei den Aufgaben *Experiment1*, *Experiment2* und *Daten*. Diese können jeweils von einem Modeler oder einem Recorder durchgeführt werden. Die mit diesen Aufgaben verbundenen Rollenwechsel garantieren, dass jeder Lernende nur eine der drei Aufgaben vollziehen kann. Schließlich stellt die Rollenhistorie sicher, dass der Lerner, welcher die *Fragestellung* formuliert hat, wie gewünscht, an der Aufgabe *Ergebnisse* beteiligt sein muss, da er zu diesem Zeitpunkt die Rolle RecorderEx einnimmt. Außerdem wird bei dieser Aufgabe noch einer der zwei anderen Lernenden, welche dann die Rolle ModelerEx besitzen, benötigt. Der Lernfortschritt und die Lernhistorie einzelner Akteure werden bei diesem Modellierungsansatz somit in das Rollendiagramm einkodiert.
Als Konvention zur Vereinfachung des Zustandsautomaten setzen wir voraus, dass bei Aktionen, die im Rollendiagramm nicht explizit einen Rollenwechsel verursachen, die bisherige Rolle erhalten bleibt. Die Aktivität *Modell* führt beispielsweise für einen Akteur, der sich in der Rolle Modeler befindet, keinen Rollenwechsel herbei und kann deshalb im Rollendiagramm entfallen.
Zusammenfassend lässt sich festhalten, dass den Lernenden durch das Prinzip veränderlicher Rollen ein Rahmen gegeben wird, in dem sie die Aufgaben untereinander aufteilen können. Der gewünschte Freiheitsgrad der eigenständigen Aufgabenaufteilung durch die Schüler kann sehr genau spezifiziert werden. Mit festen Rollen wie bei der Geschäftsprozessmodellierung ist dies nicht möglich. Ganz ohne Rollen bestünde das Problem, dass ein Lernender alle Aufgaben alleine erledigen könnte, was oft nicht dem gewünschten Lernprozess entspricht.
Bei der vorgestellten Modellierungssprache werden dynamische Rollen von Lernenden in einer sinnvollen Weise repräsentiert. Auch kollabo-

rative Aktivitäten lassen sich intuitiv abbilden. Insbesondere lässt sich spezifizieren, wann nicht alle Personen einer Lerngruppe für die jeweiligen Aufgaben benötigt werden, so dass im Petrinetzmodell parallel aktivierte Aufgaben auch parallel bearbeitet werden können, wenn die benötigten Lernenden zur Verfügung stehen. Schließlich wird auch der Fortschritt der Lernenden im Lernprozess durch das Rollenkonzept abgebildet. Die vorgeschlagene Modellierungssprache ist daher für die Modellierung kollaborativer Lernprozesse geeignet. Neben der Anwendbarkeit speziell für Lehr- / Lernprozesse, sehen wir dabei auch eine Nutzbarkeit für Geschäftsprozesse, in denen Gruppenaktivitäten und dynamische Rollen wesentlich sind.
Gruppen werden in dem Modell allerdings nur implizit über kollaborative Aktivitäten berücksichtigt. In vielen Fällen wäre aber auch eine explizite Gruppenmodellierung wünschenswert. Hierzu bieten sich unmittelbar zwei Möglichkeiten an. Zum einen lassen sich Gruppen durch verschiedene Prozessinstanzen repräsentieren. Hier wären allerdings noch Konzepte zur Modellierung von Abhängigkeiten zwischen Prozessinstanzen wichtig, um auch dynamische Gruppenumformierungen darstellen zu können. Andererseits lassen sich Gruppen ganz analog zu Rollen explizit in den Zustandsdiagrammen der Akteure modellieren. Bei Gruppen ist es aber im Gegensatz zu Rollen wichtig, die Gesamt-Gruppendynamik darzustellen, welche sich dann nur implizit über die Gruppenzugehörigkeiten der einzelnen Akteure ergibt. Daher wären hier Erweiterungen der Modellierungsansätze interessant, welche zu jedem Zeitpunkt explizit die Lerngruppen darstellen, z.B. geeignete Gruppierungen der Zustandsdiagramme im Akteurspool.
Mit diesen Überlegungen sind die Grundideen der neuen Modellierungssprache klar. Für eine formale Definition einer Schaltregel entsprechender Modelle sei gesagt, dass sich der globale Akteurspool immer durch eine entsprechenden High-Level-Stelle darstellen lässt. Jedes Modell der neuen Modellierungssprache lässt sich also in ein Netz einer bestimmten Klasse von High-Level-Netzen [129] übersetzen, nämlich ein Netz mit nur einer sehr speziellen High-Level-Stelle für den Ressourcen-Pool. Für das Beispielmodell aus Abbildung 38 zusammen mit Abbildung 39 stellt Abbildung 40 einen Ausschnitt des zugehörigen High-Level-Netzes in der Form eines gefärbten Petrinetzes [129] dar. Dieses Beispiel sollte das allgemeine Prinzip der Übersetzung deutlich machen. Es ist zu beachten, dass bei klassischen Workflownetzen mit statischen Rollen eine analoge Übersetzung möglich ist. Im Gegensatz zu diesem Beispiel entsprechen in dem klassischen Fall dann allerdings die von einer Transition aus dem Ressourcen-Pool konsumierten Marken immer den von der Transition im Pool produzierten Marken. Wichtig ist noch zu bemerken, dass in dem Netz in Abbildung 40 einerseits die dynamischen Rollen und das Verhalten der Akteure nicht mehr deutlich werden und dass das Netz andererseits sehr unübersichtlich ist. Daher ist zu Illustrationszwecken die zuvor gewählte Darstellung zu bevorzugen.
Wie erläutert, ist nun auch für kollaborative Lernprozesse ein streng ablauforientierter Modellierungsansatz sinnvoll. Hierbei stellt sich nun das Problem, kollaborative Lernprozessmodelle obiger Form aus entsprechenden Ablaufinformationen zu synthetisieren. Für diesen Zweck müssen die Ereignisse von Abläufen zusammen mit den ausführenden Akteuren gegeben sein. An kollaborativen Aktivitäten sind dabei mehrere Akteure beteiligt. Durch die Akteursinformationen sind die

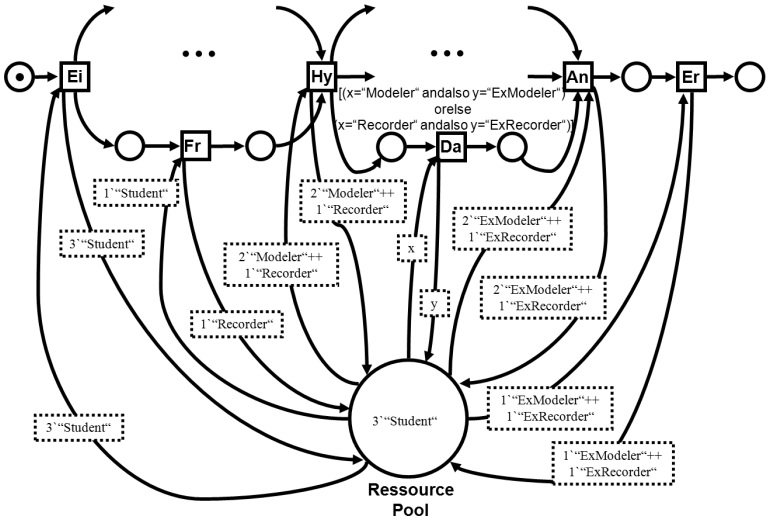

Abbildung 40: Übersetzung in ein gefärbtes Petrinetz.

Lernhistorien der an den verschiedenen Ereignissen beteiligten Akteure bestimmt. Aus diesen kann die Rollendynamik des Lernprozesses gewonnen werden. Im Gegensatz zu den bisherigen Syntheseproblemen stellt sich hier also die zusätzliche Frage, wie eine entsprechende Ressourcenpool-Stelle synthetisiert werden kann.
So ergibt sich in unserem Beispiel die Frage, wie unser Modell des kollaborativen Lernprozesses „Pflanzen“ (Abbildungen 38 und 39) aus den Abläufen aus Abbildung 37 zusammen mit entsprechenden zu den Ereignissen zugeordneten Akteursinformationen synthetisiert werden kann. Entsprechende Akteursinformationen sehen beispielsweise folgendermaßen aus: Drei Akteure A1, A2 und A3 sind zu Ei zugeordnet. A1 führt Fr aus, A2 und A3 führen Pl und Mo aus. A1, A2 und A3 sind danach an Pl beteiligt. A1 ist dann zu E1, A2 zu E2 und A3 zu Da zugeordnet. An wird wieder von A1, A2 und A3 durchgeführt. Zu Er wird schließlich A1 und A2 zugeordnet. Dieses Beispiel legt eine mögliche Lernhistorie für die Akteure des Prozesses fest. Um das kollaborative Lernprozessmodell „Pflanzen“ insgesamt aus entsprechenden Ablaufinformationen korrekt erzeugen zu können, müssen alle solchen in dem Prozess möglichen Lernhistorien durch die Abläufe vorgegeben sein. Ein Beispiel für eine Lernhistorie, welche nicht konsistent zu unserem Prozess ist, ergibt sich, wenn in obigem Beispiel A2 und A3 zu Er zugeordnet werden, da bei dem Prozess „Pflanzen“ der Akteur, welcher Fr durchgeführt hat, auch an Er beteiligt sein muss. Ein solcher Ablauf darf in unserem Beispiel somit nicht in einer Ablauf-Spezifikation auftauchen. Wird zunächst von den Akteursinformationen abstrahiert, so ergibt sich in diesem Beispiel wiederum das Problem der Synthese eines Petrinetzes aus einer partiellen Sprache. Die Berücksichtigung der Akteure in der Form entsprechender Rollendiagramm erfordert dann allerdings geeignet erweiterte Syntheseverfahren.

3

FORMALE GRUNDLAGEN

In diesem Kapitel werden die formalen Grundlagen für die weiteren Ausführungen der Arbeit bereitgestellt. Diese sind kompakt und kurz dargestellt. Das Kapitel soll hauptsächlich als Nachschlagemöglichkeit für formale Details der Arbeit dienen.
Zuerst werden allgemeine mathematische Notationen und Schreibweisen festgelegt. Dann werden die grundlegenden Definitionen zu Petrinetzen und halbgeordneten Abläufen, den zwei zentralen Formalismen dieser Arbeit, eingeführt. Schließlich werden noch Verfahren zur Lösung linearer Ungleichungssysteme vorgestellt. Diese spielen eine wichtige Rolle bei der Synthese von Petrinetzen aus halbgeordneten Abläufen.

3.1 NOTATIONEN

Wir bezeichnen die Menge der nicht-negativen ganzen Zahlen mit $\mathbb{N}$. Für die Menge der positiven ganzen Zahlen schreiben wir $\mathbb{N}^+$. Das Symbol ∞ repräsentiert eine unendlich große Zahl, d.h. $\infty > n$ für jedes $n \in \mathbb{N}$. Wir schreiben $\mathbb{N}_\infty$ für $\mathbb{N} \cup \{\infty\}$ und $\mathbb{N}_\infty^+$ für $\mathbb{N}^+ \cup \{\infty\}$.
Für Mengen, Relationen, Familien und Funktionen werden die üblichen Schreibweisen verwendet. Wir führen hier Abkürzungen für einige spezielle Funktionen ein. Sei eine Definitionsmenge A gegeben. Die Identitätsfunktion auf A wird mit id_A bezeichnet, die Nullfunktion auf A wird mit 0_A bezeichnet und die charakteristische Funktion einer Teilmenge $X \subseteq A$ wird mit 1_X bezeichnet.
Sei A eine Teilmenge eines geordneten Körpers, welcher eine Grundmenge von Skalaren darstellt. Eine Matrix bzw. ein Vektor über A ist eine Matrix bzw. ein Vektor, die bzw. der nur Einträge aus A besitzt. Wir bezeichnen Matrizen mit großen fett gedruckten Buchstaben und Spaltenvektoren mit kleinen fett gedruckten Buchstaben. Das Produkt einer Matrix $\mathbf{A}$ mit einem Vektor $\mathbf{b}$ wird durch $\mathbf{A} \cdot \mathbf{b}$ und das Skalarprodukt eines Vektors $\mathbf{a}$ und eines Vektors $\mathbf{b}$ durch $\mathbf{a}^t \cdot \mathbf{b}$ dargestellt. Ungleichheitszeichen $<, \leqslant, >, \geqslant$ zwischen Vektoren werden komponentenweise interpretiert. Mit $\mathbf{0}$ notieren wir Vektoren bzw. Matrizen, deren Einträge alle 0 sind.
Für eine Menge A heißt eine Funktion $m : A \to \mathbb{N}$ Multimenge über A. Wie üblich wird die Menge aller Multimengen über A mit $\mathbb{N}^A$ bezeichnet. Für jedes Element $a \in A$ wird durch $m(a)$ die Häufigkeit seines Auftretens in der Multimenge m bestimmt. Wir schreiben kurz $a \in m$, falls $m(a) > 0$ ist. Eine Multimenge m heißt endlich, falls die Menge $\{a \in m\}$ endlich ist, und leer, falls die Menge $\{a \in m\}$ leer ist. Die Mächtigkeit $|m| = \sum_{a \in m} m(a)$ einer endlichen Multimenge m ist die Anzahl ihrer Elemente gezählt mit deren Vielfachheit. Sei $X \subseteq A$. Eine Multimenge der Form 1_X kann als Menge X interpretiert werden

und vice versa. Wir unterscheiden daher nicht zwischen der Menge $X \subseteq A$ und der Multimenge $1_X \in \mathbb{N}^A$. Neben der funktionalen Darstellung einer Multimenge m gibt es auch ihre Darstellung als formale Summe $m = \sum_{a \in A} m(a) \cdot a$. Vergleich, Addition und Subtraktion von Multimengen m und m' sind elementweise definiert:

- $m \leqslant m' \iff \forall a \in A : m(a) \leqslant m'(a)$,
- $m < m' \iff m \leqslant m' \wedge m \neq m'$,
- $m + m' = \sum_{a \in A} (m(a) + m'(a)) \cdot a$,
- $m - m' = \sum_{a \in A} \max(m(a) - m'(a), 0) \cdot a$.

Ein Tripel $(A, +, 0)$ aus einer Menge A, einer binären Operation $+ : A \times A \to A$ und einem Element $0 \in A$ heißt kommutatives Monoid, falls die folgenden Bedingungen erfüllt sind:

- $\forall a, b, c \in A : (a + b) + c = a + (b + c)$ (Assoziativität),
- $\forall a \in A : a + 0 = 0 + a = a$ (neutrales Element),
- $\forall a, b \in A : a + b = b + a$ (Kommutativität).

Ein Morphismus zwischen zwei Monoiden $(A, +, 0)$ und $(A', +', 0')$ ist eine Funktion $f : A \to A'$, welche $f(a + b) = f(a) +' f(b)$ für alle $a, b \in A$ und $f(0) = 0'$ erfüllt. Die Menge $\mathbb{N}^A$ aller Multimengen über einer Menge A definiert das kommutative Monoid $(\mathbb{N}^A, +, 0_A)$. Dieses wird auch als freies kommutatives Monoid über der Menge der Generatoren A bezeichnet.
Ein gerichteter Graph ist ein Paar $(V, \to)$, bestehend aus einer Menge V von Knoten und einer binären Relation $\to \subseteq V \times V$ auf V, die als Menge der gerichteten Kanten bezeichnet wird. Anschaulich werden die Knoten durch Punkte dargestellt und die gerichteten Kanten durch Pfeile, welche die Knoten verbinden. Für $v \in V$ und $W \subseteq V$ ist ${}^\bullet v = \{v' \in V \mid v' \to v\}$ der Vorbereich von v, $v^\bullet = \{v' \in V \mid v \to v'\}$ der Nachbereich von v, ${}^\bullet W = \bigcup_{w \in W} {}^\bullet w$ der Vorbereich von W und $W^\bullet = \bigcup_{w \in W} w^\bullet$ der Nachbereich von W. Eine endliche Folge $v_0 \dots v_n, n \in \mathbb{N}$ von Knoten ist ein Weg des Graphen, falls $v_{i-1} \to v_i$ für $i \in \{1, \dots, n\}$. Ein Weg $v_0 \dots v_n$ mit $n \geqslant 1$ und $v_0 = v_n$ ist ein Zyklus.
Ein Tripel $(V, V', \to)$ bestehend aus zwei disjunkten Knotenmengen V und V' und einer Flussrelation $\to \subseteq (V \times V') \cup (V' \times V)$ bezeichnen wir als Netzgraph. Ein Netzgraph $(V, V', \to)$ kann als ein (sog. bipartiter) gerichteter Graph $(V \cup V', \to)$ aufgefasst werden. Der Vor- und der Nachbereich von Knoten eines Netzgraphen $(V, V', \to)$ sowie Wege und Zyklen eines Netzgraphen sind durch den gerichteten Graphen $(V \cup V', \to)$ definiert.
Eine relationale Struktur ist ein Tripel $\mathcal{S} = (V, \prec, \sqsubset)$, wobei V eine Menge von Knoten ist und $\prec \subseteq V \times V$ und $\sqsubset \subseteq V \times V$ binäre Relationen auf V sind.
Ein Transitionssystem $TS = (S, L, \to)$ besteht aus einer Menge S von Zuständen, einer Menge L von Beschriftungen (oder Labels) und einer Transitionsrelation $\to \subseteq S \times L \times S$, die für jeden Zustand aus S und jedes Eingabezeichen aus L bestimmt welche Nachfolgezustände existieren. Ein Transitionssystem kann als gerichteter Graph mit Kantenbeschriftungen interpretiert werden. Ein Transitionssystem heißt deterministisch, falls aus $(s, l, s'), (s, l, s'') \in \to$ folgt, dass $s' = s''$ gilt. Ein

initialisiertes Transitionssystem $(S, L, \to, s_0)$ ist ein Transitionssystem mit einem Anfangszustand $s_0 \in S$. Ein initialisiertes Transitionssystem heißt erreichbar, falls für jedes $s \in S$, $s \neq s_0$, Zustände $s_1, \ldots, s_n = s$, $n \geqslant 1$, und Beschriftungen $l_1, \ldots, l_n$ existieren, so dass $(s_{i-1}, l_i, s_i) \in \to$ für $i \in \{1, \ldots, n\}$. Für ein Transitionssystem $TS = (S, A, \to)$ über einem kommutativen Monoid $(A, +, 0)$ wird $(s, 0, s) \in \to$ für alle $s \in S$ gefordert. Ist A ein freies kommutatives Monoid über einer endlichen Menge, so nennen wir TS ein Schritt-Transitionssystem.

3.2 PETRINETZE

In dieser Arbeit werden Petrinetze [183, 79, 192, 22, 182, 74] zur Darstellung nebenläufiger Systeme verwendet. Petrinetze und Dialekte von Petrinetzen wie UML-Aktivitätsdiagramme und EPKs haben sich in vielen Bereichen der Informatik, beispielsweise der Geschäftsprozessmodellierung, als Modellierungsstandard durchgesetzt. Petrinetze sind ein graphisch orientierter Formalismus. Sie unterliegen aber zugleich einer strikt formalen mathematischen Beschreibung.
Allerdings muss man beachten, dass es nicht den einen Petrinetzformalismus gibt, sondern der Begriff Petrinetz inzwischen eine Vielzahl an netzbasierten Modellierungssprachen umfasst [74]. Neben den klassischen Stellen/Transitionsnetzen gibt es viele weitere Petrinetzklassen, die mehr oder weniger starke Abwandlungen bzw. Erweiterungen von Stellen/Transitionsnetzen sind. Solche Petrinetzklassen verbessern die Modellierungstauglichkeit, die Modellierungseigenschaften oder die Darstellungsmächtigkeit für spezielle praktische Anwendungen oder theoretische Fragestellungen. Beispiele für Erweiterungen von Stellen/Transitionsnetzen sind Kapazitäten für Stellen, Kontextkanten, individuelle Marken und Datenstrukturen oder sogar objektorientierte Eigenschaften [131].

3.2.1 *Stellen/Transitions-Netze*

Wir beginnen mit den grundlegenden Definitionen von Petrinetzen in ihrer klassischen Form als Stellen/Transitions-Netze. Stellen/Transitions-Netze können als natürliche Erweiterung der ursprünglichen Petrinetzdefinitionen von Petri verstanden werden [183].

Definition 3.2.1 (Stellen/Transitions-Netz)

Definition 3.2.1 (Stellen/Transitions-Netz)
Ein Stellen/Transitions-Netz (kurz S/T-Netz) N *ist ein Tripel* (P, T, W) *aus*

- *einer Menge* P *von Stellen (oder Plätzen),*
- *einer endlichen Menge* T *von Transitionen mit* $P \cap T = \emptyset$ *und*
- *einer Multimenge von Kanten* $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ *(auch Gewichtsfunktion genannt).*

Die Flussrelation oder Flusskantenmenge eines S/T-Netzes ist die Relation $F = \{(x, y) \in (P \times T) \cup (T \times P) \mid W(x, y) > 0\}$. Ein S/T-Netz (P, T, W) definiert damit einen Netzgraphen (P, T, F). Die Transitionen eines S/T-Netzes repräsentieren lokale Aktionen (aktiv), z.B. eine von einem Akteur auszuführende Aktivität, und werden graphisch durch Rechtecke dargestellt. Durch die Stellen werden lokale Zustände (passiv), z.B. das Vorhandensein von Instanzen einer bestimmten Objektklasse oder die Gültigkeit einer Bedingung, modelliert. Stellen werden durch Kreise

symbolisiert. Die Multimenge von Kanten zeigt die durch Ausführen von Aktionen resultierenden Zustandsänderungen an. Sie definiert die Menge von Flusskanten zwischen Transitionen und Stellen und ordnet jeder Kante ein Gewicht zu; es ist zu beachten, dass Kanten niemals zwei Stellen oder zwei Transitionen verbinden. Flusskanten werden durch Pfeile dargestellt. Diese werden mit den entsprechenden Gewichten beschriftet, wobei das Gewicht 1 in Illustrationen weggelassen wird.
Obige Definition lässt unendliche Stellenmengen zu. S/T-Netze mit endlichen Stellenmengen nennen wir endliche S/T-Netze.
Der Vor- und der Nachbereich von Stellen und Transitionen sind durch den Netzgraphen (P, T, F) definiert (siehe Abschnitt 3.1). Der Vor- und der Nachbereich einer Stelle bestehen per Definition nur aus Transitionen und der Vor- und der Nachbereich einer Transition nur aus Stellen. Wir definieren weiter den Multivor- und den Multinachbereich einer Transition $t \in T$ und einer Multimenge von Transitionen $\tau \in \mathbb{N}^T$ durch die Multimengen von Stellen ${}^\circ t = \sum_{p \in P} W(p,t) \cdot p$, $t^\circ = \sum_{p \in P} W(t,p) \cdot p$, ${}^\circ \tau = \sum_{t \in \tau} \tau(t)\, {}^\circ t$ und $\tau^\circ = \sum_{t \in \tau} \tau(t) t^\circ$. Analog lassen sich auch der Multivor- und der Multinachbereich von Multimengen von Stellen als Multimengen von Transitionen definieren.
Der aktuelle Zustand eines S/T-Netzes ist durch eine Markierung definiert. Eine Markierung eines S/T-Netzes ordnet jeder Stelle eine nicht-negative Anzahl von Marken zu. Formal ist eine Markierung damit eine Multimenge $m \in \mathbb{N}^P$ über der Menge der Stellen. Graphisch wird eine Markierung durch eine entsprechende Anzahl von schwarzen Punkten in jeder Stelle dargestellt.

Definition 3.2.2 (Markiertes S/T-Netz)

Definition 3.2.2 (Markiertes S/T-Netz)
Ein markiertes S/T-Netz ist ein Paar (N, m_0), wobei N ein S/T-Netz und m_0 eine Markierung von N ist. Die Markierung m_0 heißt Anfangsmarkierung von (N, m_0).

Ein markiertes S/T-Netz (N, m_0), $N = (P, T, W)$, ist ein Teilnetz eines markierten S/T-Netzes (N', m_0'), $N' = (P', T', W')$, falls $T = T'$, $P \subseteq P'$, $W = W'|_{(P \times T) \cup (T \times P)}$ und $m_0 = m_0'|_P$.
Zustandsübergänge eines markierten S/T-Netzes finden durch das Ausführen von Aktionen statt. Man spricht hierbei davon, dass die entsprechenden Transitionen schalten. Die sog. Schaltregel legt die Schaltbedingungen für Transitionen und die durch einen Schaltvorgang bewirkte Markierungsänderung fest. Eine Transition ist in einer bestimmten Markierung aktiviert, d.h. sie kann schalten, wenn die Stellen in ihrem Vorbereich genügend Marken bzgl. der Gewichte der zugehörigen Flusskanten enthalten. Wenn eine Transition schaltet, konsumiert (verbraucht) sie Marken in den Stellen in ihrem Vorbereich und produziert (erzeugt) Marken in den Stellen in ihrem Nachbereich jeweils entsprechend der Gewichte der zugehörigen Flusskanten.
Die Schaltregel legt aber nicht nur das Verhalten einzelner Transitionen fest, sondern sie bestimmt auch, ob in einer Markierung mehrere Transitionen unabhängig voneinander schalten können. Unabhängigkeit erfordert, dass die Transitionen sowohl simultan, als auch in jeder beliebigen Reihenfolge, schalten können. In diesem Fall sagen wir, dass die Transitionen nebenläufig schalten können. Transitionen können auch zu sich selbst nebenläufig schalten. In einem S/T-Netz ist eine Multimenge von Transitionen, genannt Transitionsschritt, nebenläufig aktiviert, wenn in der aktuellen Markierung die Summe der für das

(u.U. mehrfache) Schalten jeder einzelnen Transition benötigten Marken vorhanden ist. Die Folgemarkierung, welche durch das Schalten eines Schrittes entsteht, stimmt mit der Markierung, welche durch das (u.U. mehrfache) Schalten der enthaltenen Einzeltransitionen entsteht, überein.

DEFINITION 3.2.3 (SCHALTREGEL FÜR S/T-NETZ)
Sei $N = (P, T, W)$ *ein S/T-Netz und* m *eine Markierung von* N.
Eine Transition $t \in T$ *ist in* m *aktiviert, falls* $m \geqslant {}^\circ t$. *Ist eine Transition* t *in einer Markierung* m *aktiviert, so kann sie schalten. Ihr Schalten überführt* m *in die Folgemarkierung* $m' = m - {}^\circ t + t^\circ$. *Wir schreiben* $m \xrightarrow{t} m'$, *um auszudrücken, dass die Transition* t *in* m *aktiviert ist und das Schalten von* t *zur Folgemarkierung* m' *führt.*
Ein Transitionsschritt $\tau \in \mathbb{N}^T$ *ist in* m *aktiviert, falls* $m \geqslant {}^\circ \tau$. *Ist ein Transitionsschritt* τ *in einer Markierung* m *aktiviert, so kann er schalten. Sein Schalten überführt* m *in die Folgemarkierung* $m' = m - {}^\circ \tau + \tau^\circ$. *Wir schreiben* $m \xrightarrow{\tau} m'$, *um auszudrücken, dass der Transitionsschritt* τ *in* m *aktiviert ist und das Schalten von* τ *zur Folgemarkierung* m' *führt.*

Definition 3.2.3 (Schaltregel für S/T-Netz)

Die Schaltregel für Transitionsschritte verallgemeinert die wohlbekannte Schaltregel für einzelne Transitionen. Für einelementige Schritte entsprechen sich die Regeln. Wir unterscheiden in diesem Kontext daher auch nicht zwischen einelementigen Transitionsschritten und einzelnen Transitionen.

BEISPIEL: *Abbildung 3 zeigt ein endliches markiertes S/T-Netz mit der Anfangsmarkierung* $2p_1 + p_2 + p_3$. *Die Multivor- und Multinachbereiche der drei Transitionen des Netzes sind durch* ${}^\circ a = p_1 + p_2$, ${}^\circ b = p_2 + p_3$ *und* ${}^\circ c = p_4$ *sowie* $a^\circ = 2p_2$, $b^\circ = p_4$ *und* $c^\circ = 0$ *gegeben. Folglich sind in der Anfangsmarkierung nur die nicht-leeren Transitionsschritte* a *und* b *aktiviert. Das Schalten von* a *überführt die Anfangsmarkierung in die Folgemarkierung* $p_1 + 2p_2 + p_3$. *In dieser Markierung sind wiederum die Transitionsschritte* a *und* b *aktiviert, zusätzlich ist nun aber auch noch der Schritt* $a + b$ *aktiviert. Das Schalten des letzteren Schrittes führt zur Markierung* $2p_2 + p_4$. *In dieser Markierung ist nur noch der nicht-leere Schritt* c *aktiviert, welcher zur Markierung* $2p_2$ *führt. In dieser Markierung ist schließlich keine Transition mehr aktiviert.*

Beispiel

Die sequentielle Semantik eines markierten S/T-Netzes ist durch seine aktivierten Schaltfolgen gegeben, die Schrittsemantik durch seine aktivierten Schrittfolgen. Aktivierte Schaltfolgen ergeben sich durch hintereinander geordnetes Schalten von einzelnen Transitionen, aktivierte Schrittfolgen durch hintereinander geordnetes Schalten von Transitionsschritten.

DEFINITION 3.2.4 (AKTIVIERTE SCHALTFOLGE, AKTIVIERTE SCHRITTFOLGE)
Sei $N = (P, T, W)$ *ein S/T-Netz und* m *eine Markierung von* N. *Eine endliche Folge von Transitionen* $\sigma = t_1 \dots t_n$ $(n \in \mathbb{N}, t_1, \dots, t_n \in T)$ *heißt Schaltfolge der Länge* n. *Eine nichtleere Schaltfolge* $\sigma = t_1 \dots t_n$ *heißt in* m *aktiviert, falls eine Folge von Markierungen* $m_1, \dots, m_n$ *existiert, so dass*

$$m \xrightarrow{t_1} m_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} m_n.$$

Wir schreiben dafür auch $m \xrightarrow{\sigma} m_n$. *Eine endliche Folge von Transitionsschritten* $\sigma = \tau_1 \dots \tau_n$ $(n \in \mathbb{N}, \tau_1, \dots, \tau_n \in \mathbb{N}^T)$ *heißt Schrittfolge der*

Definition 3.2.4 (Aktivierte Schaltfolge, aktivierte Schrittfolge)

Länge n. *Eine nichtleere Schrittfolge* $\sigma = \tau_1 \dots \tau_n$ *heißt in* m *aktiviert, falls eine Folge von Markierungen* $m_1, \dots, m_n$ *existiert, so dass*

$$m \xrightarrow{\tau_1} m_1 \xrightarrow{\tau_2} \dots \xrightarrow{\tau_n} m_n.$$

Wir schreiben dafür auch $m \xrightarrow{\sigma} m_n$. *Die leere Schaltfolge bzw. die leere Schrittfolge* $\sigma = \epsilon$ *ist in jeder Markierung* m *aktiviert und wir schreiben* $m \xrightarrow{\epsilon} m$.

Definition 3.2.5 (Sequentielle Semantik, Schrittsemantik)

Definition 3.2.5 (Sequentielle Semantik, Schrittsemantik)
Die Menge $\mathfrak{Seq}(N, m_0)$ *aller in* m_0 *aktivierten Schaltfolgen heißt sequentielle Semantik eines markierten S/T-Netzes* (N, m_0).
Die Menge $\mathfrak{Step}(N, m_0)$ *aller in* m_0 *aktivierten Schrittfolgen heißt Schrittsemantik eines markierten S/T-Netzes* (N, m_0).

Eine Markierung m' heißt erreichbar von einer Markierung m, falls eine Schaltfolge (bzw. äquivalent eine Schrittfolge) σ existiert mit $m \xrightarrow{\sigma} m'$. In einem markierten S/T-Netz (N, m_0) nennt man die Menge der Markierungen M, die von der Anfangsmarkierung aus erreichbar sind, kurz erreichbar.
Die sequentielle Semantik bzw. die Schrittsemantik lässt sich durch den sog. Erreichbarkeitsgraphen bzw. den Schritterreichbarkeitsgraphen repräsentieren. Der Erreichbarkeitsgraph eines markierten S/T-Netzes (N, m_0) ist das deterministische, erreichbare, initialisierte Transitionssystem $\mathfrak{EG}(N, m_0) = (M, T, \rightarrow, m_0)$, wobei $(m, t, m') \in \rightarrow \Longleftrightarrow m \xrightarrow{t} m'$. Der Schritterreichbarkeitsgraph eines markierten S/T-Netzes (N, m_0) ist das deterministische, erreichbare, initialisierte Schritt-Transitionssystem $\mathfrak{NEG}(N, m_0) = (M, \mathbb{N}^T, \rightarrow, m_0)$, wobei $(m, \tau, m') \in \rightarrow \Longleftrightarrow m \xrightarrow{\tau} m'$. Die Menge $\mathfrak{Seq}(N, m_0)$ bzw. $\mathfrak{Step}(N, m_0)$ entspricht genau der Menge der endlichen in m_0 beginnenden Wege von $\mathfrak{EG}(N, m_0)$ bzw. $\mathfrak{NEG}(N, m_0)$.

Beispiel

Beispiel: *Die sequentielle Semantik* $\mathfrak{Seq}(N, m_0)$ *des Netzes* (N, m_0) *aus Abbildung 3 ist durch* $\{a, b, aa, ab, bc, aab, aba, abc, aabc, abac, abca\}$ *gegeben. Der Erreichbarkeitsgraph* $\mathfrak{EG}(N, m_0)$ *ist in Abbildung 41 links dargestellt. Unter Vernachlässigung leerer Schritte ergibt sich die Schrittsemantik von* (N, m_0) *durch* $\mathfrak{Seq}(N, m_0) \cup \{aa + b, aa + bc, aba + c\}$. *Der Schritterreichbarkeitsgraph* $\mathfrak{NEG}(N, m_0)$ *ist wiederum unter Weglassung der leeren Schritte in Abbildung 41 rechts illustriert.*

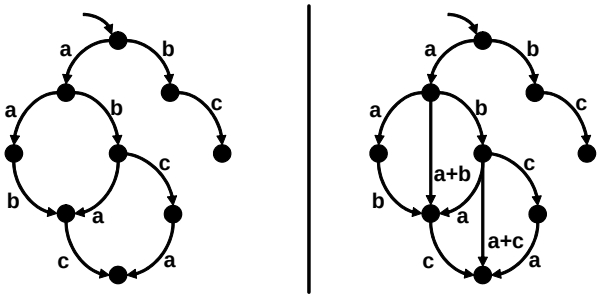

Abbildung 41: Erreichbarkeitsgraph und Schritterreichbarkeitsgraph des Netzes aus Abbildung 3.

Zuletzt ist noch der Begriff der impliziten Stelle eines S/T-Netzes wichtig. Eine implizite Stelle ist eine in dem Sinne unnötige Stelle, dass

eine solche Stelle von einem Netz entfernt werden kann, ohne dass sich dadurch das Verhalten des Netzes (vgl. [39, 57]) bzw. genauer die Schrittsemantik des Netzes ändert. Formal umfasst das Entfernen einer Stelle natürlich auch deren gewichtete Verbindungskanten zu Transitionen und deren Anfangsmarkierung.

Definition 3.2.6 (Implizite Stelle)

Definition 3.2.6 (Implizite Stelle)
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes S/T-Netz und* $p \in P$. *Sei weiter* (N', m_0'), $N' = (P \setminus \{p\}, T, W|_{(P\setminus\{p\}\times T)\cup(T\times P\setminus\{p\})})$, $m_0' = m_0|_{P\setminus\{p\}}$. *Die Stelle* p *heißt implizite Stelle, falls* $\mathfrak{Step}(N', m_0') = \mathfrak{Step}(N, m_0)$.

Beispiel

Beispiel: *Das Netz in Abbildung 42 ist das Netz aus Abbildung 3 ergänzt um die zwei Stellen* $p_2 + 2p_3$ *und* $2p_4$. *Es hat dennoch dieselbe Menge an aktivierten Schrittfolgen. Dementsprechend sind diese zwei Stellen implizit. Genauso sind aber auch die Stellen* p_3 *und* p_4 *implizit. Dies zeigt, dass es nicht sinnvoll ist, alle impliziten Stellen auf einmal zu entfernen, sondern, wenn das Verhalten eines Netzes nicht verändert werden soll, sollten implizite Stellen sukzessive weggelassen werden.*

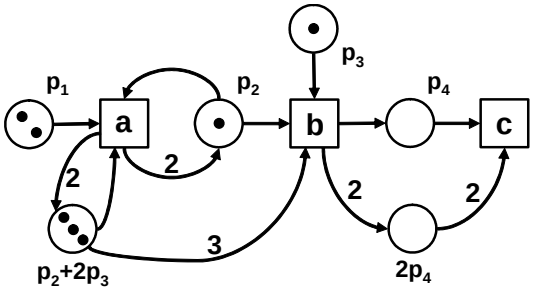

Abbildung 42: Netz mit impliziten Stellen.

3.2.2 *Stellen/Transitions-Netze mit gewichteten Inhibitorkanten*

Eine sehr wichtige Erweiterung von S/T-Netzen sind sog. Inhibitorkanten [182, 127], welche als Schaltbedingung an Transitionen das Testen, ob eine Stelle in der aktuellen Markierung leer ist, ermöglichen. S/T-Netze mit Inhibitorkanten können im Gegensatz zu gewöhnlichen S/T-Netzen Turingmaschinen simulieren, so dass gewisse Entscheidbarkeitsprobleme wie das Erreichbarkeitsproblem, welche für S/T-Netze entscheidbar sind, für S/T-Netze mit Inhibitorkanten unentscheidbar sind [109]. Inhibitorkanten stellen die klassische Möglichkeit der Erweiterung von S/T-Netzen, um Turingmächtigkeit zu erzielen, dar. In [182] wird behauptet, dass Petrinetze mit Inhibitorkanten der intuitivste und direkteste Ansatz sind, um die Modellierungsmächtigkeit von Petrinetzen zu erhöhen. Neben der theoretischen Bedeutung sind Inhibitorkanten auch für praktische Modellierungszwecke sehr beliebt [41, 81]. Außer den einfachen Inhibitorkanten gibt es allgemeiner gewichtete Inhibitorkanten [144]. Diese ermöglichen das Testen, ob eine Stelle höchstens die vom Gewicht einer Inhibitorkante bestimmte Anzahl an Marken enthält. Die Petrinetzklasse der S/T-Netze mit gewichteten Inhibitorkanten soll im Folgenden eingeführt werden.

Definition 3.2.7 (S/T-Netze mit gewichteten Inhibitorkanten)

Definition 3.2.7 (S/T-Netze mit gewichteten Inhibitorkanten)
Ein S/T-Netze mit gewichteten Inhibitorkanten (kurz STI-Netz) ist ein Viertupel $N = (P, T, W, I)$ *aus*

- *einem S/T-Netz* (P, T, W)*, dem* N *zugrunde liegenden Netz, und*
- *einer Inhibitorgewichtsfunktion* $I : P \times T \to \mathbb{N}_\infty$.

Der negative Kontext einer Transition $t \in T$ *ist die Funktion* ${}^-t : P \to \mathbb{N}_\infty$*, definiert durch* ${}^-t(p) = I(p, t)$ *für alle* $p \in P$*. Der negative Kontext eines Transitionsschrittes* $\tau \in \mathbb{N}^T$ *ist die Funktion* ${}^-\tau : P \to \mathbb{N}_\infty$*, definiert durch* ${}^-\tau(p) = \min\{{}^-t(p) \mid t \in \tau\}$ *für alle* $p \in P$.
Eine Markierung eines STI-Netzes ist eine Multimenge $m \in \mathbb{N}^P$ *über der Menge der Stellen.*
Ein markiertes STI-Netz ist ein Paar (N, m_0)*, wobei* N *ein STI-Netz ist und* m_0 *eine Markierung von* N *ist. Die Markierung* m_0 *heißt Anfangsmarkierung von* (N, m_0).

Die Inhibitorkantenmenge eines STI-Netzes ist die Relation $J = \{(x, y) \in (P \times T) \mid I(x, y) \neq \infty\}$. In graphischen Darstellungen werden Inhibitorkanten mit Kreisen als Pfeilspitzen gezeichnet und mit ihrem Inhibitorgewicht beschriftet. Das Inhibitorgewicht 0 wird in Illustrationen weggelassen.
In der Literatur gibt es zwei verschiedene Semantiken für Inhibitorkanten, die sog. a-priori und die sog. a-posteriori Semantik. In dieser Arbeit ist vor allem die a-priori Semantik von Bedeutung. Bei dieser Semantik findet das Testen auf Aktiviertheit von Transitionen bzgl. der Inhibitorkanten vor dem eigentlichen Schaltvorgang der Transitionen statt. Bei der a-priori Semantik entsteht im Gegensatz zur a-posteriori Semantik und zu gewöhnlichen S/T-Netzen die Besonderheit, dass Situationen möglich sind, in denen ein Transitionsschritt simultan schalten kann, obwohl die Transitionen des Schrittes nicht in jeder Reihenfolge hintereinander schalten können. Daher gibt die Schaltregel für Transitionsschritte unter der a-priori Semantik an, dass die entsprechenden Transitionen simultan schalten können, was nicht bedeutet, dass sie auch nebenläufig schalten können. Von Nebenläufigkeit kann erst gesprochen werden, wenn die Transitionen simultan und in jeder Reihenfolge geordnet schalten können. Die Schaltregel für einzelne Transitionen ist wie im Falle von S/T-Netzen durch den Spezialfall einelementiger Transitionsschritte definiert und wird daher nicht extra aufgeführt.

Definition 3.2.8 (Schaltregel für STI-Netze (a-priori Semantik))

Definition 3.2.8 (Schaltregel für STI-Netze (a-priori Semantik))
Sei $N = (P, T, W, I)$ *ein STI-Netz und* m *eine Markierung von* N*. Ein Transitionsschritt* $\tau \in \mathbb{N}^T$ *ist in* m *unter der a-priori Semantik aktiviert, falls* $m \geqslant {}^\circ\tau$ *und* $m(p) \leqslant {}^-\tau(p)$ *für alle* $p \in P$*. Ist ein Transitionsschritt* τ *in einer Markierung* m *aktiviert, so kann er schalten. Sein Schalten überführt* m *in die Folgemarkierung* $m' = m - {}^\circ\tau + \tau^\circ$*. Wir schreiben* $m \xrightarrow{\tau} m'$*, um auszudrücken, dass der Transitionsschritt* τ *in* m *aktiviert ist und das Schalten von* τ *zur Folgemarkierung* m' *führt.*

Mithilfe der Schaltregel ergeben sich die Begriffe der aktivierten Schaltfolge, der aktivierten Schrittfolge, der sequentiellen Semantik, der Schrittsemantik, des Erreichbarkeitsgraphen, des Schritterreichbarkeitsgraphen und der impliziten Stelle für STI-Netze mit der a-priori Semantik analog zum Falle von S/T-Netzen.

BEISPIEL: *Abbildung 43 zeigt ein endliches markiertes STI-Netz mit der Anfangsmarkierung* $2p_1 + 2p_2 + p_4$. *Die Multivor- und Multinachbereiche der drei Transitionen des Netzes sind durch* ${}^\circ a = p_1$, ${}^\circ b = p_2 + p_3$ *und* ${}^\circ c = p_4$ *sowie* $a^\circ = p_3$, $b^\circ = 2p_3$ *und* $c^\circ = 0$ *gegeben. Es ist zu beachten, dass die Transition* c *einen nichttrivialen negativen Kontext besitzt. Es gilt* ${}^-c(p_3) = 2$. *Wenn die Stelle* p_3 *also mehr als zwei Marken enthält, so ist die Transition* c *nicht aktiviert. In der Anfangsmarkierung ist* c *aber aktiviert. Nebenläufig hierzu ist auch die Transition* a *sogar nebenläufig zu sich selbst aktiviert. Somit sind der Transitionsschritt* $2a + c$ *sowie alle kleineren Schritte aktiviert. Durch das Schalten des Schrittes* $2a$ *entsteht die Markierung* $2p_2 + 2p_3 + p_4$. *In dieser Markierung ist* c *wiederum aktiviert. Auch* b *ist aktiviert, sogar nebenläufig zu sich selbst. Allerdings kann* b *nicht nebenläufig zu* c *schalten, da das Schalten von* b *eine dritte Marke in der Stelle* p_3 *erzeugt und somit das Schalten von* c *verhindert. Jedoch können* b *und* c *unter der a-priori Semantik synchron zueinander schalten. Dies liegt daran, dass sich vor dem eigentlichen Schaltvorgang des Transitionsschrittes* $2b + c$ *nur zwei Marken in der Stelle* p_3 *befinden. Der Transitionsschritt* $2b + c$ *ist also aktiviert und sein Schalten führt zur Markierung* $4p_3$. *In dieser Markierung ist nun keine Transition mehr aktiviert.* *Beispiel*

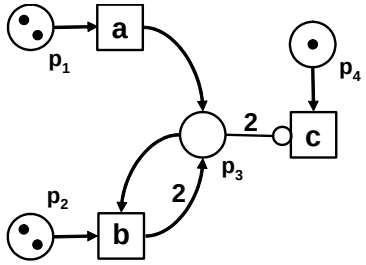

Abbildung 43: Ein markiertes STI-Netz.

3.2.3 *Netztypen*

Die im letzten Unterabschnitt diskutierte Netzklasse der STI-Netze mit der a-priori Semantik wird im Weiteren auf Grund der Unterscheidbarkeit von nebenläufigem und synchronem Schaltverhalten von besonderer Relevanz sein. Allerdings sind wir insgesamt daran interessiert, uns in dieser Arbeit nicht auf S/T-Netze zu beschränken, sondern auch andere Petrinetzklassen zu berücksichtigen. Um Synthesefragestellungen, aber auch andere Probleme, parametrisch bzgl. der Petrinetzklasse behandeln zu können, wurde die Definition des Netztyps [16, 17, 12, 18] eingeführt. Mithilfe des Begriffs des Netztyps lassen sich Petrinetze vieler verschiedener Petrinetzklassen einheitlich definieren.
Ein Netztyp ist ein Transitionssystem $\mathcal{N} = (LS, LE, \mapsto)$, welches eine Petrinetzklasse festlegt [16]. Ein Petrinetz eines Netztyps besteht, wie von S/T-Netzen gewohnt, aus Stellen P, Transitionen T und einer Gewichtsfunktion W, welche beschriftete Kanten definiert. Der Netztyp spezifiziert die möglichen Werte LS, die in Stellen des Netzes gespeichert werden können (eine Verallgemeinerung der Marken bei S/T-Netzen), die möglichen Modifikationen LE dieser Werte, die durch Transitionen des Netzes durchgeführt werden können (eine Verallgemeinerung des Konsumierens und Produzierens von Marken bei S/T-Netzen), und

die Schaltrelation $\mapsto$, welche die Aktiviertheit und das Schaltverhalten der Transitionen festlegt (Verallgemeinerung der Schaltregel von S/T-Netzen). Somit gibt LS die erlaubten lokalen Markierungen, LE die erlaubten Beschriftungen von Kanten und $\mapsto$ die Schaltregel für eine Netzklasse an.

Definition 3.2.9 (Netztyp)

Definition 3.2.9 (Netztyp)
Ein Netztyp ist ein deterministisches Transitionssystem $\mathcal{N} = (LS, LE, \mapsto)$ *über einem kommutativen Monoid* $(LE, +, 0)$*, wobei* LS *eine Menge von lokalen Zuständen repräsentiert,* LE *eine Menge von lokalen Ereignissen ist und* $\mapsto \subseteq LS \times LE \times LS$ *eine partiell definierte Veränderung von lokalen Zuständen durch lokale Ereignisse darstellt.*

Wir verwenden für $(s, e, s') \in \mapsto$ die Schreibweise $s \xrightarrow{e} s'$. Ein lokales Ereignis e ist in einem lokalen Zustand s aktiviert, falls ein s' mit $s \xrightarrow{e} s'$ existiert. Hierfür schreiben wir $s \xrightarrow{e}$.

Definition 3.2.10 (Petrinetz eines Netztyps)

Definition 3.2.10 (Petrinetz eines Netztyps)
Ein Petrinetz N *eines Netztyps* $\mathcal{N}$ *ist ein Tripel* (P, T, W) *aus*

- *einer Menge* P *von Stellen,*
- *einer endlichen Menge* T *von Transitionen mit* $P \cap T = \emptyset$ *und*
- *einer Gewichtsfunktion* $W : P \times T \to LE$.

Eine Markierung eines Petrinetzes eines Netztyps ist eine Funktion $m : P \to LS$.
Ein markiertes Petrinetz eines Netztyps $\mathcal{N}$ *ist ein Paar* (N, m_0)*, wobei* N *ein Netz des Typs* $\mathcal{N}$ *ist und* m_0 *eine Markierung von* N *ist. Die Markierung* m_0 *heißt Anfangsmarkierung von* (N, m_0).

Die (als ungerichtet zu interpretierenden) Kanten zwischen Stellen und Transitionen eines Petrinetzes eines Netztyps sind durch die Menge $\{(x, y) \in (P \times T) \mid W(x, y) \neq 0\}$ gegeben.
Die Schaltregel für Transitionsschritte lässt sich ähnlich wie im Falle von S/T-Netzen definieren [17]. Die Schaltregel für einzelne Transitionen ergibt sich wiederum durch den Spezialfall einelementiger Transitionsschritte.

Definition 3.2.11 (Schaltregel für Petrinetze eines Netztyps)

Definition 3.2.11 (Schaltregel für Petrinetze eines Netztyps)
Sei $N = (P, T, W)$ *ein Netz des Netztyps* $\mathcal{N} = (LS, LE, \mapsto)$ *und* m *eine Markierung von* N*. Ein Transitionsschritt* $\tau \in \mathbb{N}^T$ *ist in* m *aktiviert, falls für jede Stelle* $p \in P$ *gilt:*

$$\exists s' \in LS : (m(p), \sum_{t \in \tau} \tau(t) \cdot W(p, t), s') \in \mapsto .$$

Ist ein Transitionsschritt τ *in einer Markierung* m *aktiviert, so kann er schalten. Sein Schalten überführt* m *in die durch* $m'(p) = s'$ *für alle* $p \in P$ *eindeutig definierte Folgemarkierung* m'*. Wir schreiben* $m \xrightarrow{\tau} m'$*, um auszudrücken, dass der Transitionsschritt* τ *in* m *aktiviert ist und das Schalten von* τ *zur Folgemarkierung* m' *führt.*

Mithilfe der Schaltregel ergeben sich auch hier wieder die Begriffe der aktivierten Schaltfolge, der aktivierten Schrittfolge, der sequentiellen Semantik, der Schrittsemantik, des Erreichbarkeitsgraphen, des Schritterreichbarkeitsgraphen und der impliziten Stelle für Petrinetze eines Netztyps analog zum Falle von S/T-Netzen.

Die Definition des Netztyps deckt die meisten relevanten Petrinetzklassen ab. Im folgenden Beispiel werden die schon diskutierten Klassen der S/T-Netze und der STI-Netze, sowie die grundlegende Klasse der elementaren Netze als Netztypen definiert. Dabei betrachten wir STI-Netze sowohl mit der bisher behandelten a-priori als auch mit der a-posteriori Semantik.

BEISPIEL: *Beispiel*

- *Der Netztyp der S/T-Netze ist durch* $\mathcal{N}_{st} = (\mathbb{N}, \mathbb{N} \times \mathbb{N}, \mapsto_{st})$ *definiert, wobei* $(n, (i, j), n') \in \mapsto_{st}$ *dann und nur dann gilt, wenn* $n \geqslant i$ *und* $n' = n - i + j$. *Das kommutative Monoid der lokalen Ereignisse* $\mathbb{N} \times \mathbb{N}$ *besitzt das neutrale Element* $(0, 0)$ *und die Verknüpfung der komponentenweisen Addition. Der Netztyp* $\mathcal{N}_{st}$ *ist in Abbildung 44 links graphisch illustriert. Die Gewichtsfunktion* W *ordnet jedem Paar* $(p, t) \in P \times T$ *eines Netzes vom Typ* $\mathcal{N}_{st}$ *ein Paar von nichtnegativen ganzen Zahlen zu. Die erste Komponente wird interpretiert als die Anzahl von Marken, welche* t *aus* p *konsumiert, und die zweite Komponente wird interpretiert als die Anzahl von Marken, welche* t *in* p *produziert. Das Gewicht einer Kante* (p, t) *kodiert also die Gewichte sowohl der von* p *nach* t *gerichteten Kante als auch der von* t *nach* p *gerichteten Kante eines entsprechenden S/T-Netzes.*

- *Der Netztyp der elementaren Netze ist durch* $\mathcal{N}_{en} = (\{0, 1\}, \{nop, in, out, failure\}, \mapsto_{en})$ *definiert, wobei* $\mapsto_{en} = \{(0, nop, 0), (1, nop, 1), (0, out, 1), (1, in, 0)\}$. *Das kommutative Monoid der lokalen Ereignisse* $\{nop, in, out, failure\}$ *besitzt das neutrale Element* nop *und die Verknüpfung* $+$, *welche durch* $x + y = failure$ *für* $x, y \neq nop$ *gegeben ist. Der Netztyp* $\mathcal{N}_{en}$ *ist in Abbildung 44 rechts dargestellt. Die Gewichtsfunktion* W *ordnet jedem Paar* $(p, t) \in P \times T$ *eines Netzes vom Typ* $\mathcal{N}_{en}$ *eine Beschriftung aus* $\{nop, in, out, failure\}$ *zu. Das lokale Ereignis* nop *wird derart interpretiert, dass keine Flusskante zwischen* p *und* t *vorhanden ist,* in *wird als von* p *nach* t *gerichtete Flusskante interpretiert,* out *wird als von* t *nach* p *gerichtete Flusskante interpretiert und* $failure$ *wird als nicht erlaubte Beschriftung interpretiert. Auf diese Weise ergibt sich die Schaltregel der klassischen elementaren Netze [196].*

- *Der Netztyp der STI-Netze ist durch* $\mathcal{N}_{sti} = (\mathbb{N}, \mathbb{N} \times \mathbb{N} \times \mathbb{N}_\infty, \mapsto_{sti})$ *für die a-posteriori Semantik und durch* $\mathcal{N}_{\overleftarrow{sti}} = (\mathbb{N}, \mathbb{N} \times \mathbb{N} \times \mathbb{N}_\infty, \mapsto_{\overleftarrow{sti}})$ *für die a-priori Semantik definiert. Das kommutative Monoid der lokalen Ereignisse* $\mathbb{N} \times \mathbb{N} \times \mathbb{N}_\infty$ *besitzt das neutrale Element* $(0, 0, \infty)$ *und die Verknüpfung* $+$, *welche durch komponentenweise Addition auf den ersten beiden Komponenten und der Minimumfunktion auf der dritten Komponente gegeben ist, d.h.* $(x, y, z) + (x', y', z') = (x + x', y + y', \min\{z, z'\})$. *Für die a-posteriori Semantik gilt dann und nur dann* $(n, (i, j, k), n') \in \mapsto_{sti}$, *wenn* $n \geqslant i$, $n + j \leqslant k$ *und* $n' = n - i + j$. *Für die a-priori Semantik gilt dann und nur dann* $(n, (i, j, k), n') \in \mapsto_{\overleftarrow{sti}}$, *wenn* $n \geqslant i$, $n \leqslant k$ *und* $n' = n - i + j$. *Die Gewichtsfunktion* W *ordnet jedem Paar* $(p, t) \in P \times T$ *eines Netzes vom Typ* $\mathcal{N}_{sti}$ *bzw.* $\mathcal{N}_{\overleftarrow{sti}}$ *ein Paar von nicht-negativen ganzen Zahlen und ein Element aus* $\mathbb{N}_\infty$ *zu. Die zwei ersten Zahlen werden wie im Falle von S/T-Netzen interpretiert. Das Element aus* $\mathbb{N}_\infty$ *gibt ein Inhibitorgewicht, welches zu einer von* p *nach* t *gerichteten Inhibitorkante zugeordnet ist, an. Das Gewicht einer Kante* (p, t) *kodiert also neben*

den Gewichten von Flusskanten zwischen p *und* t *auch das Gewicht einer Inhibitorkante zwischen* p *und* t *eines entsprechenden STI-Netzes.*

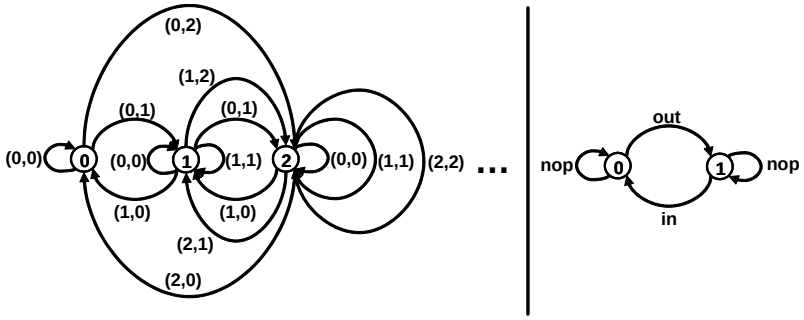

Abbildung 44: Der Netztyp $\mathcal{N}_{st}$ und der Netztyp $\mathcal{N}_{en}$.

Weitere Beispiele von Netzklassen, die sich als Netztypen darstellen lassen, sind S/T-Netze oder elementare Netze

- mit ungewichteten Inhibitorkanten und mit gewichteten oder ungewichteten Lesekanten jeweils sowohl mit der a-priori als auch der a-posteriori Semantik,
- mit Resetkanten,
- mit Signalkanten,
- mit Kapazitäten sowohl mit der „erst produzieren, dann konsumieren“ als auch mit der „erst konsumieren, dann produzieren“ Semantik und
- mit Prioritäten

Auch höhere Petrinetze (High-Level-Petrinetze, gefärbte Petrinetze oder Prädikat/Transitions-Netze) und viele weitere denkbare Arten von Petrinetzen lassen sich durch Netztypen repräsentieren.

3.3 HALBGEORDNETE ABLÄUFE

Unter einem Szenario wird in der Informatik die Beschreibung einer von mehreren möglichen Verhaltensweisen einer betrachteten Menge von Systemen und Akteuren verstanden [123, 187, 110]. Typischerweise gibt ein Szenario eine Abfolge oder einen Ablauf von Ereignissen an. Wir betrachten im Weiteren halbgeordnete Abläufe von Ereignissen, d.h. zwischen zwei Ereignissen kann eine Ordnungsbeziehung bestehen, sie können aber auch unabhängig voneinander sein. Unter einem Ereignis verstehen wir hierbei das Durchführen einer Aktivität. In Petrinetzen entspricht dies dann dem Schalten einer Transition.

3.3.1 *Beschriftete partielle Ordnungen*

Halbgeordnete Abläufe lassen sich in natürlicher Weise durch beschriftete partielle Ordnungen darstellen [188, 126, 40, 226, 131, 80, 94]. Beschriftete partielle Ordnungen gehören zu den bedeutendsten Formalismen sowohl zur Verhaltensbeschreibung von Petrinetzen, als auch ganz allgemein zur Beschreibung von Abläufen nebenläufiger Systeme (siehe

Einleitung). Beschriftete partielle Ordnungen werden auch als partielle Wörter oder partiell geordnete Multimengen bezeichnet [108, 188].

DEFINITION 3.3.1 (BESCHRIFTETE PARTIELLE ORDNUNG)
Eine partielle Ordnung (häufig strikte Halbordnung oder nur Halbordnung genannt) ist ein endlicher gerichteter Graph $\mathrm{po} = (V, <)$, *wobei* $< \subseteq V \times V$ *irreflexiv und transitiv ist.*
Eine beschriftete partielle Ordnung (kurz BPO) über einer Menge von Beschriftungen T *ist ein Tupel* $\mathrm{bpo} = (V, <, l)$, *wobei* $(V, <)$ *eine partielle Ordnung und* $l : V \to T$ *eine Funktion, welche jedem Knoten* $v \in V$ *eine Beschriftung aus* T *zuordnet, ist. Die Funktion* l *heißt Beschriftungsfunktion von* bpo. *Die Menge* $l(V) \subseteq T$ *heißt Beschriftungsmenge von* bpo.

Definition 3.3.1 (Beschriftete partielle Ordnung)

Eine partielle Ordnung ist azyklisch, d.h. sie enthält keine Zyklen. Durch λ wird die leere partielle Ordnung $(\emptyset, \emptyset)$ bzw. die leere BPO $(\emptyset, \emptyset, \emptyset)$ bezeichnet.
Die Knoten einer BPO repräsentieren die Ereignisse eines Ablaufs, wobei die Knotenbeschriftung eines Ereignisses die durchgeführte Aktivität beschreibt. Die Ordnungsbeziehungen der Ereignisse werden durch die Kanten einer BPO modelliert. Normalerweise können die Kanten im Sinne einer zeitlichen „früher als"-Beziehung interpretiert werden. Ungeordnete Ereignisse stellen nebenläufiges Verhalten dar. Es ist zu beachten, dass die Begriffe Knoten und Ereignis im Weiteren synonym verwendet werden, wobei der Begriff Knoten den Graphaspekt und der Begriff Ereignis den Modellierungsaspekt einer BPO betont. In Abbildungen werden die Kanten durch Pfeile und die Knoten durch Rechtecke repräsentiert. Die Knotenbeschriftungen werden an den Knoten dargestellt. Die Knotenidentitäten werden meist vernachlässigt.
Im Weiteren ist es wichtig, dass wir auch formal isomorphe BPOs nicht unterscheiden, da nur die Knotenbeschriftungen, nicht aber die Knotenidentitäten von BPOs bei der Modellierung von Abläufen eine Rolle spielen. Wenn wir also von einer BPO $(V, <, l)$ sprechen, meinen wir normalerweise die Isomorphieklasse von $(V, <, l)$, d.h. BPO $(V, <, l)$ ist eine Kurzschreibweise für die Isomorphieklasse von $(V, <, l)$. Dabei sind $(V, <, l)$ und $(V', <', l')$ isomorph, falls eine bijektive Funktion $\psi : V \to V'$ zwischen den Knotenmengen existiert, welche die Ordnungsrelation und die Beschriftungsfunktion erhält, d.h. $l(v) = l'(\psi(v))$ für alle $v \in V$ und $v < w \iff \psi(v) <' \psi(w)$ für alle $v, w \in V$. Um explizit von den Knotenidentitäten der Knotenmenge V abstrahieren zu können, definieren wir die von einer Teilmenge $V' \subseteq V$ repräsentierte Multimenge von Beschriftungen $|V'|_l = \sum_{t \in T} |\{v \in V' \mid l(v) = t\}| \cdot t$. Diese Multimenge bezeichnen wir als Parikhvektor von V'.
Eine Menge von Abläufen wird durch eine Menge von BPOs beschrieben.

DEFINITION 3.3.2 (PARTIELLE SPRACHE)
Eine Menge $\mathcal{L}$ *von BPOs heißt partielle Sprache. Die Vereinigung der Beschriftungsmengen aller in* $\mathcal{L}$ *enthaltener BPOs heißt Beschriftungsmenge von* $\mathcal{L}$.

Definition 3.3.2 (Partielle Sprache)

BEISPIEL: *Abbildung 4 zeigt eine partielle Sprache bestehend aus zwei BPOs. Die erste BPO* bpo_1 *modelliert den sequentiellen Ablauf der Durchführung der Aktivität* b *gefolgt von der Aktivität* c. *Die BPO* bpo_2 *beginnt mit einem* a*-Ereignis gefolgt von einem* b*- und einem anschließenden* c*-Ereignis. Nebenläufig zu den letzteren zwei Ereignissen findet nach dem ersten* a*-Ereignis auch noch ein zweites* a*-Ereignis statt.*

Beispiel

Im weiteren Verlauf dieses Unterabschnittes werden wir die für unsere Zwecke wichtigen Begriffe zu partiellen Ordnungen einführen, wobei wir alle Begriffe auch analog für BPOs verwenden. Anschließend werden einige wichtige Notationen für partielle Sprachen festgelegt. Als erstes leiten wir weitere Relationen aus der Ordnungsrelation $<$ einer partiellen Ordnung $(V, <)$ ab. Wir definieren

- die nicht-strikte Ordnungsrelation $\leqslant = < \cup \mathrm{id}_V$,
- die Menge der Gerüstkanten $<_g = \{(v, v') \in V \times V \mid v < v' \wedge \nexists v'' : v < v'' < v'\}$,
- die Menge der auf einer Linie liegenden Knotenpaare $\mathrm{li} = \{(v, v') \in V \times V \mid v \leqslant v' \vee v' \leqslant v\}$ und
- die Menge der unabhängigen Knotenpaare $\mathrm{co} = \{(v, v') \in V \times V \mid v \not\leqslant v' \wedge v' \not\leqslant v\}$.

In Abbildungen werden wir aus Gründen der Übersichtlichkeit oft nur das sog. Hassediagramm einer partiellen Ordnung darstellen.

Definition 3.3.3 (Hassediagramm)

Definition 3.3.3 (Hassediagramm)
Das Hassediagramm einer partiellen Ordnung $(V, <)$ ist der gerichtete Graph $(V, <_g)$.

Die Gerüstkantenmenge $<_g$ ist die minimale Teilmenge von $<$, deren transitiver Abschluss wieder der Kantenmenge $<$ entspricht.
Die Relationen li und co definieren Mengen von geordneten und ungeordneten Knoten.

Definition 3.3.4 (Li-Menge, Linie, Tiefe, Co-Menge, Schnitt, Weite)

Definition 3.3.4 (Li-Menge, Linie, Tiefe, Co-Menge, Schnitt, Weite)
Sei $\mathrm{po} = (V, <)$ eine partielle Ordnung.
Eine Li-Menge ist eine Knotenmenge $L \subseteq V$ mit: $\forall v, v' \in L : v \,\mathrm{li}\, v'$. Eine Linie ist eine maximale Li-Menge. Die Tiefe von po ist die maximale Mächtigkeit einer Linie von po.
Eine Co-Menge ist eine Knotenmenge $C \subseteq V$ mit: $\forall v, v' \in C, v \neq v' : v \,\mathrm{co}\, v'$. Ein Schnitt ist eine maximale Co-Menge. Die Weite von po ist die maximale Mächtigkeit eines Schnitts von po.

Es lässt sich zeigen, dass die Weite einer partielle Ordnung der minimalen Anzahl von Li-Mengen, in welche die Knotenmenge partitioniert werden kann, entspricht (siehe [94]). Wichtige Beispiele für Schnitte einer partiellen Ordnung $(V, <)$ sind die Menge der minimalen Knoten $\mathrm{Min}(V, <) = \{v \in V \mid {}^\bullet v = \emptyset\}$ und die Menge der maximalen Knoten $\mathrm{Max}(V, <) = \{v \in V \mid v^\bullet = \emptyset\}$.
Wir verallgemeinern die Ordnungsrelation $<$ und die Unabhängigkeitsrelation co derart, dass wir eine Co-Menge C einer partiellen Ordnung $(V, <)$ in Beziehung zu einem Knoten $v \in V$ setzen können. Wir schreiben

- $v < C$, falls $v < v'$ für ein $v' \in C$,
- $v > C$, falls $v > v'$ für ein $v' \in C$, und
- $v \,\mathrm{co}\, C$, falls $v \,\mathrm{co}\, v'$ für alle $v' \in C$.

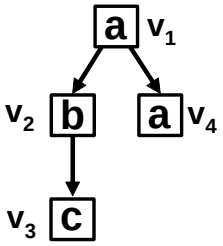

Abbildung 45: Hassediagramm von bpo_2 aus Abbildung 4.

BEISPIEL: *Abbildung 45 zeigt das Hassediagramm der BPO* bpo_2 *aus Abbildung 4. In dieser Darstellung sind auch Knotenidentitäten angegeben. Die betrachtete BPO hat die zwei Linien* $\{v_1, v_2, v_3\}$ *und* $\{v_1, v_4\}$*. Damit hat die BPO eine Tiefe von Drei. Die Schnitte der BPO sind durch die Mengen* $\text{Min}(bpo_2) = \{v_1\}, \{v_2, v_4\}$ *und* $\text{Max}(bpo_2) = \{v_3, v_4\}$ *gegeben. Somit besitzt die BPO eine Weite von Zwei.* *Beispiel*

An dieser Stelle definieren wir nun zwei Kompositionsoperatoren für partielle Ordnungen. Seien $po = (V, <)$ und $po' = (V', <')$ zwei partielle Ordnungen mit $V \cap V' = \emptyset$. Wir definieren die sequentielle Komposition von po und po′ durch $po; po' = (V \cup V', < \cup <' \cup (V \times V'))$ und die parallele Komposition von po und po′ durch $po \parallel po' = (V \cup V', < \cup <')$.
Bei der Verwendung der Kompositionsoperatoren für BPOs $bpo = (V, <, l)$ und $bpo' = (V', <', l')$ ist Folgendes zu beachten. Da den Knotenidentitäten bei BPOs keine Bedeutung zukommt und sich die Voraussetzung $V \cap V' = \emptyset$ immer durch eine Umbenennung von Knoten sicherstellen lässt, spielt letztere Voraussetzung für die Komposition von BPOs keine Rolle. Somit lässt sich beispielsweise auch eine BPO mit sich selbst komponieren. Hierzu führen wir mit der Schreibweise $bpo^0 = \lambda$ und $bpo^n = bpo^{n-1}; bpo$ für $n \in \mathbb{N}^+$ einen Potenz-Operator für BPOs ein.
Wir betrachten zwei spezielle Klassen von partiellen Ordnungen.

DEFINITION 3.3.5 (TOTAL, SCHRITTWEISE LINEAR)
Sei $(V, <)$ *eine partielle Ordnung. Ist* $co = \emptyset$*, so heißt* $(V, <)$ *total (oder auch linear), ist* $co \cup id_V$ *transitiv, so heißt* $(V, <)$ *schrittweise linear.* *Definition 3.3.5 (Total, schrittweise linear)*

Eine partielle Ordnung, die mehr Ordnung enthält als eine andere partielle Ordnung mit derselben Knotenmenge, nennt man eine Sequentialisierung der anderen partiellen Ordnung.

DEFINITION 3.3.6 (SEQUENTIALISIERUNG)
Seien $po = (V, <)$ *und* $po' = (V, <')$ *partielle Ordnungen. Wir bezeichnen* po′ *als Sequentialisierung (oder auch Erweiterung) von* po*, falls* $< \subseteq <'$*. Ist zusätzlich* $< \neq <'$*, so heißt* po′ *echte Sequentialisierung von* po*.*
Ist po′ *total und eine Sequentialisierung von* po*, so bezeichnen wir* po′ *als Linearisierung von* po*.*
Ist po′ *schrittweise linear und eine Sequentialisierung von* po*, so bezeichnen wir* po′ *als Schrittlinearisierung von* po*.* *Definition 3.3.6 (Sequentialisierung)*

Die Menge der Sequentialisierungen einer partiellen Ordnung po bezeichnen wir durch $\text{Seq}(po)$ und die Menge der Schrittlinearisierungen von po durch $\text{Slin}(po)$.

Präfixe bzw. Anfangsstücke einer partiellen Ordnung werden ähnlich zu dem in der Informatik üblichen Begriff des Präfixes einer Zeichenkette bzw. eines Wortes definiert.

Definition 3.3.7 (Präfix)

Definition 3.3.7 (Präfix)
Sei $\mathrm{po} = (V, <)$ *eine partielle Ordnung.*
Eine Teilmenge $V' \subseteq V$ *heißt abgeschlossen, falls*

$$\forall v, v' \in V : (v \in V' \wedge v' < v) \implies v' \in V'.$$

Ist $V' \subseteq V$ *abgeschlossen, so heißt* $\mathrm{po}' = (V', <|_{V' \times V'})$ *Präfix von* po. *Ist zusätzlich* $V' \neq V$, *so heißt* po' *echtes Präfix von* po. *In diesem Fall schreiben wir* $\mathrm{po}' \lhd \mathrm{po}$.
Ein Präfix $\mathrm{po}' = (V', <|_{V' \times V'})$ *von* po *heißt Präfix einer Co-Menge* C *von* po, *falls* $\{v \in V \mid v < C\} \subseteq V'$ *und* $C \cap V' = \emptyset$.

Die Menge der Präfixe einer partiellen Ordnung po bezeichnen wir durch $\mathrm{Pref}(\mathrm{po})$. Es sei angemerkt, dass die Relation $\lhd$ eine partielle Ordnung auf $\mathrm{Pref}(\mathrm{po})$ definiert.
Es lässt sich beobachten, dass jede partielle Ordnung eine Sequentialisierung und ein Präfix von sich selbst ist. Die leere partielle Ordnung ist ein Präfix jeder partiellen Ordnung. Eine Co-Menge kann mehrere Präfixe haben. Insbesondere ist jedes Präfix einer partiellen Ordnung ein Präfix der leeren Co-Menge $\emptyset$ und jedes Präfix einer Co-Menge C ist auch ein Präfix jeder Teilmenge $C' \subseteq C$ der Co-Menge. Aber es gibt zu jeder Co-Menge C ein eindeutiges minimales Präfix $\{v \in V \mid v < C\}$. Ein Schnitt hat nur dieses eine Präfix. Aber nicht jedes Präfix einer partiellen Ordnung ist Präfix eines Schnittes. Dies sind nur diejenigen Präfixe, welche keine maximalen Knoten der partiellen Ordnung beinhalten.
Zu beachten ist an dieser Stelle, dass es bei den Begriffen des Präfixes und der Sequentialisierung für BPOs wiederum nicht auf die Knotenidentitäten ankommt, sondern nur die Beschriftungen der Knoten beachtet werden müssen.

Beispiel

Beispiel: *Wir betrachten die BPO* $\mathrm{bpo}_a = (\{v_a\}, \emptyset, (v_a, a))$, *welche nur aus einem mit* a *beschrifteten Knoten besteht (die dritte BPO in Abbildung 47). Dann gilt für die BPOs aus Abbildung 4 die Beziehung* $\mathrm{bpo}_2 = \mathrm{bpo}_a ; (\mathrm{bpo}_1 \| \mathrm{bpo}_a)$. *Die BPO* bpo_1 *ist total und damit auch schrittweise linear. Die BPO* bpo_2 *ist nicht schrittweise linear. Die BPO aus Abbildung 46 ist schrittweise linear, aber nicht total. Diese BPO ist weiterhin eine echte Sequentialisierung von* bpo_2. *Somit ist sie eine Schrittlinearisierung von* bpo_2. *Die BPO* bpo_a *mit einem Knoten ist ein echtes Präfix von* bpo_2. *Betrachten wir die Knotenidentitäten aus Abbildung 45, so ist* bpo_a *ein Präfix der Co-Mengen* $\emptyset$, $\{v_2\}$, $\{v_4\}$ *und* $\{v_2, v_4\}$. *Weitere Beispiele für Sequentialisierungen und Präfixe der BPOs* bpo_1 *und* bpo_2 *finden sich in Abbildung 47.*

Mithilfe der Begriffe der Sequentialisierung und des Präfixes führen wir folgende Bezeichnungen für partielle Sprachen ein. Sei eine partielle Sprache $\mathcal{L}$ gegeben, dann bezeichnen wir den Sequentialisierungsabschluss von $\mathcal{L}$ mit $\mathrm{Seq}(\mathcal{L}) = \bigcup_{\mathrm{bpo} \in \mathcal{L}} \mathrm{Seq}(\mathrm{bpo})$ und den Präfixabschluss von $\mathcal{L}$ mit $\mathrm{Pref}(\mathcal{L}) = \bigcup_{\mathrm{bpo} \in \mathcal{L}} \mathrm{Pref}(\mathrm{bpo})$. Den Präfix- und Sequentialisierungsabschluss $\mathrm{Pref}(\mathrm{Seq}(\mathcal{L})) = \mathrm{Seq}(\mathrm{Pref}(\mathcal{L}))$ von $\mathcal{L}$ bezeichnen wir mit $\mathrm{PS}(\mathcal{L})$. Eine partielle Sprache $\mathcal{L}$ heißt sequentialisierungsabgeschlossen, wenn $\mathrm{Seq}(\mathcal{L}) = \mathcal{L}$, präfixabgeschlossen, wenn $\mathrm{Pref}(\mathcal{L}) = \mathcal{L}$, sowie präfix- und sequentialisierungsabgeschlossen, wenn $\mathrm{PS}(\mathcal{L}) = \mathcal{L}$. Außerdem definieren wir die Menge der Schrittlinearisierungen einer partiellen Sprache $\mathcal{L}$ durch $\mathrm{Slin}(\mathcal{L}) = \bigcup_{\mathrm{bpo} \in \mathcal{L}} \mathrm{Slin}(\mathrm{bpo})$. Der Schrittabschluss

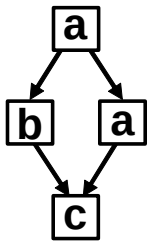

Abbildung 46: Eine Schrittlinearisierung von bpo_2 aus Abbildung 4.

$Step(\mathcal{L}) = \{bpo \mid bpo \text{ ist BPO mit } Slin(bpo) \subseteq Slin(\mathcal{L})\}$ von $\mathcal{L}$ besteht aus allen BPOs, deren Schrittlinearisierungen auch Schrittlinearisierungen von $\mathcal{L}$ sind. Eine partielle Sprache $\mathcal{L}$ heißt schrittabgeschlossen, wenn $Step(\mathcal{L}) = \mathcal{L}$. Es lässt sich leicht prüfen, dass jede schrittabgeschlossene partielle Sprache sequentialisierungsabgeschlossen ist.
Eine BPO einer partiellen Sprache $\mathcal{L}$ hat minimale Ordnung, wenn sie von keiner anderen BPO aus $Pref(\mathcal{L})$ eine Sequentialisierung ist. Die Menge aller BPOs von $\mathcal{L}$ mit minimaler Ordnung bezeichnen wir als $MinO(\mathcal{L}) = \{bpo \in \mathcal{L} \mid bpo \notin Seq(Pref(\mathcal{L}) \setminus \{bpo\})\}$. Eine BPO einer partiellen Sprache $\mathcal{L}$ hat maximale Länge, wenn sie von keiner anderen BPO aus $Seq(\mathcal{L})$ ein Präfix ist. Die Menge aller BPOs von $\mathcal{L}$ mit maximaler Länge bezeichnen wir als $MaxL(\mathcal{L}) = \{bpo \in \mathcal{L} \mid bpo \notin Pref(Seq(\mathcal{L}) \setminus \{bpo\})\}$. Die Teilmenge der BPOs einer partiellen Sprache $\mathcal{L}$, welche minimale Ordnung und maximale Länge haben, bezeichnen wir als Repräsentationssprache $Rep(\mathcal{L}) = \{bpo \in \mathcal{L} \mid bpo \notin PS(\mathcal{L} \setminus \{bpo\})\}$ von $\mathcal{L}$. Es gilt $Rep(\mathcal{L}) = Rep(Rep(\mathcal{L})) = Rep(PS(\mathcal{L})) = Rep(Seq(\mathcal{L})) = Rep(Pref(\mathcal{L}))$.

BEISPIEL: *Abbildung 47 zeigt den Präfix- und Sequentialisierungsabschluss der partiellen Sprache aus Abbildung 4. Die Repräsentationssprache der partiellen Sprache aus Abbildung 47 ist umgekehrt die partielle Sprache aus Abbildung 4.* *Beispiel*

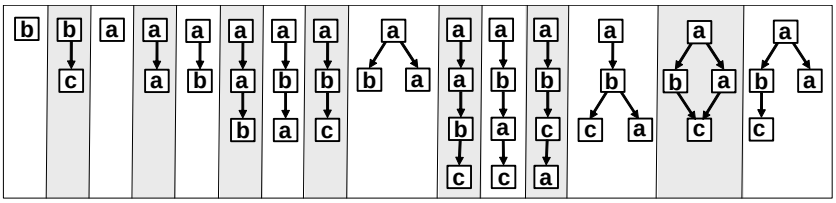

Abbildung 47: Der Präfix- und Sequentialisierungsabschluss der partiellen Sprache aus Abbildung 4.

3.3.2 *Abläufe von Stellen/Transitions-Netzen*

Mit Hilfe der Schrittschaltregel ist es, wie dargestellt, möglich, nebenläufiges Verhalten von Petrinetzen auszudrücken. Durch die Zusammenfassung von nebenläufigen Schaltereignissen zu Schritten ist dies allerdings nur sehr eingeschränkt möglich. Genauer gesagt lassen sich nur transitive Nebenläufigkeitsbeziehungen darstellen. Dies ist keineswegs intuitiv. Aus diesem Grund wird wohl die Schrittschaltregel

häufig auch im Sinne eines gleichzeitigen Schaltens der Transitionen eines Schrittes interpretiert, obwohl bei S/T-Netzen eigentlich gar nicht sinnvoll von Gleichzeitigkeit gesprochen werden kann. Um dem Phänomen der Nebenläufigkeit gerecht zu werden, eignet sich, wie in der Einleitung schon angesprochen, dahingegen die Betrachtung von halbgeordnetem Schaltverhalten.
In diesem Unterabschnitt wird die Bedeutung von BPOs zur Modellierung von halbgeordnetem Schaltverhalten von S/T-Netzen erläutert (siehe [142, 216, 217, 175, 106, 107, 40, 131]). Eine BPO modelliert einen möglichen Ablauf eines markierten S/T-Netzes, wenn die Knoten der BPO Schaltereignisse des Netzes repräsentieren und diese Schaltereignisse in der durch die Kanten der BPO bestimmten Reihenfolge im S/T-Netz stattfinden können. Genauer gesagt müssen die Schaltereignisse der BPO, solange die in der BPO spezifizierte Ordnung eingehalten wird, ansonsten in beliebiger Reihenfolge und mit beliebigen Nebenläufigkeiten im Netz möglich sein. Dies bedeutet, dass unabhängige Schaltereignisse auch tatsächlich nebenläufig durchführbar sein müssen. Geordnete Ereignisse geben die Reihenfolge der Ereignisse in dem betrachteten Ablauf an, es muss aber nicht notwendigerweise tatsächlich eine Abhängigkeit der Ereignisse bestehen. Im Folgenden wird formal definiert, unter welchen Bedingungen eine BPO einen halbgeordneten Ablauf eines S/T-Netzes repräsentiert. Dazu werden BPOs zuerst in Beziehung zu Schalt- und Schrittfolgen von S/T-Netzen gesetzt.
Totale BPOs definieren Schaltfolgen und schrittweise lineare BPOs definieren Schrittfolgen. Sei T eine Menge von Transitionen und $bpo = (V, <, l)$ eine BPO über T. Ist bpo total, so ist $co = \emptyset$. Dann gilt o.B.d.A. $V = \{v_1, \ldots, v_n\}$ und $<= \{(v_i, v_j) \mid i < j\}$. Die Schaltfolge $\sigma(bpo) = l(v_1) \ldots l(v_n)$ heißt in diesem Fall zu bpo assoziiert. Ist bpo schrittweise linear, so ist $co \cup id_V$ eine Äquivalenzrelation. Dann gilt o.B.d.A. $V = \bigcup_{i=1}^{n} V_i$ und $<= \bigcup_{i<j} V_i \times V_j$, wobei $V_1, \ldots, V_n$ die Äquivalenzklassen von $co \cup id_V$ sind. Die Schrittfolge $\sigma(bpo) = |V_1|_l \ldots |V_n|_l$ der entsprechenden Parikhvektoren heißt in diesem Fall zu bpo assoziiert. Jede Schalt- bzw. Schrittfolge ist zu einer (eindeutigen) BPO assoziiert. Eine totale bzw. schrittweise lineare BPO repräsentiert damit einen Ablauf eines S/T-Netzes, wenn die assoziierte Schalt- bzw. Schrittfolge in dem Netz aktiviert ist. Allerdings lassen sich mit Schaltfolgen gar keine und mit Schrittfolgen nur transitive Nebenläufigkeitsbeziehungen zwischen Schaltereignissen darstellen. Somit ist eine Modellierung von Abläufen mit totalen bzw. schrittweise linearen BPOs nur in sehr eingeschränkter Form möglich. Daher führen wir den Begriff der Aktiviertheit von BPOs ein, welcher für beliebige BPOs definiert, unter welchen Bedingungen sie einen Ablauf eines S/T-Netzes repräsentieren. In Übereinstimmung mit den bisherigen Überlegungen ist eine BPO aktiviert, wenn alle Schrittfolgen, welche zu Schrittlinearisierungen der BPO assoziiert sind und damit die Ordnungsrelation der BPO berücksichtigen, im Netz aktiviert sind [142]. Eine aktivierte BPO modelliert also gültiges Verhalten eines S/T-Netzes.

DEFINITION 3.3.8 (AKTIVIERTE BPO) *Definition 3.3.8 (Aktivierte BPO)*
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes S/T-Netz. Eine BPO* $bpo = (V, <, l)$ *mit* $l : V \to T$ *heißt aktiviert in* m_0 *bzgl.* N, *falls für jede Schrittlinearisierung* $bpo' \in Slin(bpo)$ *die assoziierte Schrittfolge* $\sigma(bpo')$ *in* m_0 *aktiviert ist.*
Ist eine BPO bpo *bzgl.* (N, m_0) *aktiviert, so kann sie schalten. Ihr Schalten überführt* m_0 *in die Folgemarkierung* m' *gegeben durch* $m'(p) = m_0(p) + \sum_{v \in V}(W(l(v), p) - W(p, l(v)))$ *für* $p \in P$.

Für $bpo' \in Slin(bpo)$ bezeichnen wir die Schrittfolge $\sigma(bpo')$ auch kurz als Schrittfolge von bpo. Eine analoge Bezeichnung verwenden wir auch für Schaltfolgen.
Es lässt sich leicht zeigen, dass die Definition der Aktiviertheit von BPOs in dem Sinne konsistent zur Definition der Aktiviertheit von Schrittfolgen ist, dass eine schrittweise lineare BPO genau dann aktiviert ist, wenn die assoziierte Schrittfolge aktiviert ist. Es lässt sich weiter zeigen, dass jede Sequentialisierung und jedes Präfix einer aktivierten BPO aktiviert ist [216, 131]. Eine besonders wichtige Beobachtung ist, dass die Aktiviertheit von BPOs äquivalent über die Betrachtung von Schnitten definiert werden kann, vgl. [217].

LEMMA 3.3.1 *Lemma 3.3.1*
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes S/T-Netz. Eine BPO* $bpo = (V, <, l)$ *mit* $l : V \to T$ *ist genau dann aktiviert in* m_0 *bzgl.* N, *wenn* $m_0(p) + \sum_{v \in V, v < C}(W(l(v), p) - W(p, l(v))) \geqslant \sum_{v \in C} W(p, l(v))$ *für jeden Schnitt* C *von* bpo *und jede Stelle* $p \in P$.

BEISPIEL: *Wir betrachten wiederum die BPO* bpo_2 *aus Abbildung 4 und das markierte S/T-Netz aus Abbildung 3. Die BPO ist bzgl. dieses Netzes aktiviert. Dies lässt sich folgendermaßen begründen: In der Anfangsmarkierung des Netzes ist die Transition* a *aktiviert. In der Folgemarkierung nach dem Schalten von* a *sind dann die Transitionen* b *und* a *nebenläufig in einem Schritt aktiviert. Außerdem ist nach einem Schalten von* b *dann die Transition* c *auch wiederum nebenläufig zu* a *aktiviert.* *Beispiel*
Die skizzierte eher informelle Überprüfung der Aktiviertheit von bpo_2 *entspricht gerade der in Lemma 3.3.1 formulierten Betrachtung der Schnitte einer BPO. Die drei im Rahmen von Abbildung 45 untersuchten Schnitte* $\{v_1\}$, $\{v_2, v_4\}$ *und* $\{v_3, v_4\}$ *der BPO erfordern entsprechend Lemma 3.3.1 die Überprüfung der Aktiviertheit der drei Transitionsschritte* a, $b + a$ *und* $c + a$ *in den entsprechenden Markierungen.*
Die ursprüngliche in Definition 3.3.8 formulierte formale Definition der Aktiviertheit verlangt die Betrachtung der Schrittlinearisierungen der BPO. Die Schrittlinearisierungen von bpo_2 *sind gerade diejenigen BPOs aus Abbildung 47 mit vier Knoten außer* bpo_2 *selbst, also die zehnte bis zur vorletzten BPO. Zu diesen fünf BPOs sind die Schrittfolgen* $aabc$, $abac$, $abca$, $aba + c$ *und* $aa + bc$ *assoziiert. Die Menge der Schrittfolgen von* bpo_2 *ist somit durch diese fünf Schrittfolgen gegeben. Alle fünf Schrittfolgen sind bzgl. des betrachteten Netzes aktiviert.*

Das klassische Konzept, um halbgeordnetes Schaltverhalten von Petrinetzen zu beschreiben, sind sog. Prozessnetze [106, 107]. Ein Prozessnetz eines markierten S/T-Netzes ist selbst wieder ein spezielles Petrinetz. Transitionen eines Prozessnetzes repräsentieren das Schalten von Transitionen in dem zu Grunde liegenden S/T-Netz und heißen daher üblicherweise Ereignisse. Stellen eines Prozessnetzes repräsentieren einzelne Marken in Stellen des S/T-Netzes und heißen Bedingungen.

Die Flusskanten zwischen Ereignissen und Bedingungen in Prozessnetzen berücksichtigen die Flusskanten zwischen entsprechenden Stellen und Transitionen des S/T-Netzes und deren Kantengewichte. Ein Prozessnetz ist gewissermaßen eine Abwicklung oder Entfaltung des markierten S/T-Netzes ausgehend von der Anfangsmarkierung, wobei wir uns im Falle alternativer Schaltereignisse für eine Alternative entscheiden müssen. Daher hat ein markiertes S/T-Netz üblicherweise viele verschiedene Prozessnetze. Die Definition von Prozessnetzen basiert auf dem Begriff des Kausalnetzes.

Definition 3.3.9 (Kausalnetz)

Definition 3.3.9 (Kausalnetz)
Ein Kausalnetz *ist ein Netzgraph* $O = (B, E, G)$*, der folgende Eigenschaften erfüllt:*

- B *ist unverzweigt, d.h.* $|{}^\bullet b|, |b^\bullet| \leqslant 1$ *für alle* $b \in B$.
- O *ist azyklisch.*

Die Elemente in B *heißen Bedingungen und die Elemente in* E *heißen Ereignisse.*

Da O azyklisch ist, definiert der transitive Abschluss G^+ von G eine partielle Ordnung $(B \cup E, G^+)$. Die Menge der Bedingungen, die minimal bzw. maximal bzgl. $(B \cup E, G^+)$ sind, wird mit $\mathrm{Min}(O)$ bzw. $\mathrm{Max}(O)$ bezeichnet.

Definition 3.3.10 (Prozessnetz)

Definition 3.3.10 (Prozessnetz)
Sei (N, m_0), $N = (P, T, W)$*, ein markiertes S/T-Netz. Ein Prozessnetz von* (N, m_0) *ist ein Paar* $K = (O, \rho)$ *bestehend aus einem Kausalnetz* $O = (B, E, G)$ *und einer Beschriftungsfunktion* $\rho : B \cup E \to S \cup T$*, welches folgende Eigenschaften erfüllt:*

- $\rho(B) \subseteq P$ *und* $\rho(E) \subseteq T$.
- $\forall e \in E, \forall p \in P : |\{b \in {}^\bullet e \mid \rho(b) = p\}| = W(p, \rho(e))$ *und* $\forall e \in E, \forall p \in P : |\{b \in e^\bullet \mid \rho(b) = p\}| = W(\rho(e), p)$.
- $\forall p \in P : |\{b \in \mathrm{Min}(O) \mid \rho(b) = p\}| = m_0(p)$.

Zwei Prozessnetze $K' = (O', \rho')$, $O' = (B', E', G')$, und $K = (O, \rho)$, $O = (B, E, G)$ sind isomorph, wenn es eine Bijektion $\mathrm{Iso} : B \cup E \to B' \cup E'$ mit $\mathrm{Iso}(B) = B'$, $\mathrm{Iso}(E) = E'$, $\rho' \circ \mathrm{Iso} = \rho$ und $(x, y) \in G \Leftrightarrow (\mathrm{Iso}(x), \mathrm{Iso}(y)) \in G'$ für $x, y \in B \cup E$ gibt.
Über die Bedingungen eines Prozessnetzes werden kausale Abhängigkeiten zwischen den Ereignissen, welche das Schalten von Transitionen repräsentieren, festgelegt. Indem von den Bedingungen abstrahiert wird, lassen sich aus Prozessnetzen BPOs definieren, welche die kausalen Abhängigkeiten zwischen den Ereignissen des Prozessnetzes beschreiben. Eine solche BPO, genannt Prozess-BPO, repräsentiert also einen zu einem Prozessnetz zugehörigen Ablauf.

Definition 3.3.11 (Prozess-BPO)

Definition 3.3.11 (Prozess-BPO)
Sei $K = (O, \rho)$, $O = (B, E, G)$*, ein Prozessnetz eines markierten S/T-Netzes* (N, m_0)*. Die BPO* $\mathrm{bpo}_K = (E, G^+|_{E \times E}, \rho|_E)$ *heißt Prozess-BPO von* (N, m_0) *(bzgl.* K*).*

Es lässt sich zeigen, dass jedes Präfix einer Prozess-BPO wieder eine Prozess-BPO ist. Eine Sequentialisierung einer Prozess-BPO kann wieder eine Prozess-BPO sein, muss dies im Allgemeinen aber nicht. Für

diese Arbeit ist der Zusammenhang von Prozess-BPOs zur Aktiviertheit von BPOs zentral. Es lässt sich leicht zeigen, dass jede Prozess-BPO eines S/T-Netzes aktiviert ist [142, 216, 217]. Ein wichtiges und schwierig zu beweisendes Resultat besagt weiter, dass umgekehrt jede aktivierte BPO eine Sequentialisierung einer Prozess-BPO ist [142, 216, 217]. Damit gilt die folgende Beziehung, welche zeigt, dass der durch aktivierte BPOs eingeführte Ablaufbegriff für S/T-Netze konsistent zu dem klassischen Konzept der Prozessnetze ist.

SATZ 3.3.1 *Satz 3.3.1*

Sei (N, m_0),$N = (P, T, W)$, *ein markiertes S/T-Netz. Die partielle Sprache der in* m_0 *bzgl.* N *aktivierten BPOs mit minimaler Ordnung stimmt mit der partiellen Sprache der Prozess-BPOs von* (N, m_0) *mit minimaler Ordnung überein.*

BEISPIEL: *Abbildung 48 zeigt links die Prozessnetze mit maximaler Länge des S/T-Netzes aus Abbildung 3, d.h. diejenigen Prozessnetze, welche nicht mehr um weitere Bedingungen oder Ereignisse erweitert werden können. Wie bei BPOs abstrahieren wir von den Knotenidentitäten und stellen nur die Knotenbeschriftungen dar. Die zu den drei Prozessnetzen zugehörigen Prozess-BPOs sind in Abbildung 3 rechts dargestellt. Es ist zu beachten, dass die ersten zwei Prozess-BPOs den BPOs* bpo_1 *und* bpo_2 *aus Abbildung 4 entsprechen, während die dritte Prozess-BPO eine Sequentialisierung von* bpo_2 *ist und somit keine minimale Ordnung aufweist.* *Beispiel*

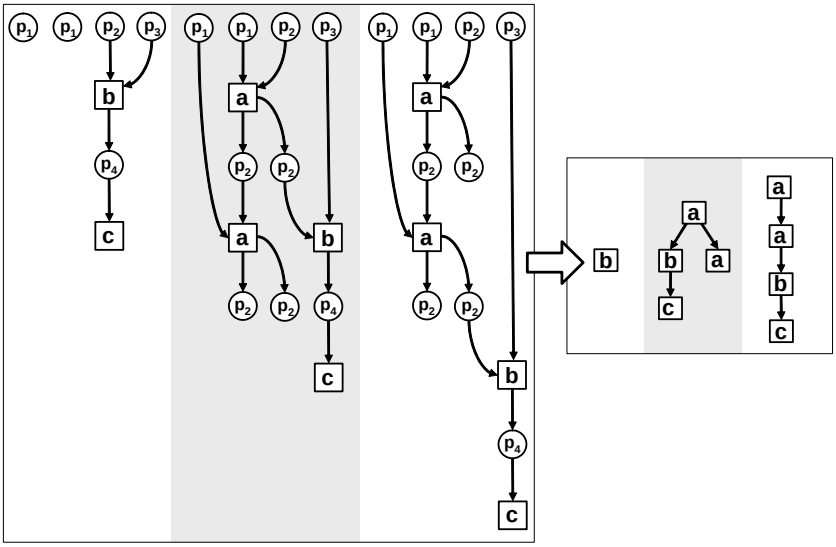

Abbildung 48: Die Prozessnetze maximaler Länge des S/T-Netzes aus Abbildung 3 und die zugehörigen Prozess-BPOs.

Zur Beschreibung halbgeordneten Schaltverhaltens von S/T-Netzen eignen sich zusammenfassend sowohl die Begriffe des Prozessnetzes und der Prozess-BPO als auch der Begriff der aktivierten BPO. Aktivierte BPOs lassen sich, wie erläutert, als Abläufe eines S/T-Netzes interpretieren. Sind zwei Schaltereignisse einer aktivierten BPO ungeordnet, so finden sie in dem modellierten Ablauf nebenläufig zueinander statt. Sind zwei Schaltereignisse einer aktivierten BPO geordnet, so heißt dies, dass sie in dem modellierten Ablauf in dieser Reihenfolge stattfinden, nicht aber, dass tatsächlich eine Abhängigkeit der beiden Schaltereignisse gefordert wird, d.h. sie können nebenläufig oder kausal

abhängig stattfinden. Die klassischen Prozessnetze bzw. Prozess-BPOs repräsentieren einen etwas anderen Ablaufbegriff. Sie modellieren nur solche Abläufe, bei denen die Ordnungsbeziehungen zwischen Schaltereignissen durch einen entsprechenden Fluss von Marken in dem S/T-Netz widergespiegelt werden können. Sind also zwei Schaltereignisse einer Prozess-BPO geordnet, so finden sie in dem modellierten Ablauf kausal abhängig voneinander statt. Ungeordnete Schaltereignisse einer Prozess-BPO finden in dem modellierten Ablauf weiterhin nebenläufig statt. Die Menge der Prozess-BPOs umfasst insbesondere alle aktivierten BPOs mit minimaler Ordnung. Diese repräsentieren Abläufe mit minimaler Kausalität. In einer aktivierten BPO mit minimaler Ordnung modellieren die Ordnungsbeziehungen zwischen Schaltereignissen kausale Abhängigkeiten, die auch durch umverteilen des Flusses der Marken nicht aufgehoben werden können. Insgesamt ergeben sich drei verschiedene Ablaufsemantiken für S/T-Netze.

Definition 3.3.12 (Ablaufsemantik)

Definition 3.3.12 (Ablaufsemantik)
Die partielle Sprache $\mathfrak{Bpo}(N, m_0)$ *aller in* m_0 *bzgl.* N *aktivierten BPOs heißt (Standard-) Ablaufsemantik eines markierten S/T-Netzes* (N, m_0).
Die partielle Sprache $\mathfrak{PBpo}(N, m_0)$ *aller Prozess-BPOs von* (N, m_0) *heißt Prozessablaufsemantik eines markierten S/T-Netzes* (N, m_0).
Die partielle Sprache $\mathfrak{MBpo}(N, m_0) = \mathrm{MinO}(\mathfrak{Bpo}(N, m_0))$ *der aktivierten BPOs mit minimaler Ordnung heißt Minimalablaufsemantik eines markierten S/T-Netzes* (N, m_0).

Beispiel

Beispiel: *Die Ablaufsemantik des S/T-Netzes aus Abbildung 3 entspricht der partiellen Sprache aus Abbildung 47. Da das Netz aus Abbildung 42 dieselbe Schrittsemantik wie das Netz aus Abbildung 3 aufweist, hat es auch dieselbe Ablaufsemantik. Die Prozessablaufsemantik des Netzes aus Abbildung 3 ist durch die drei BPOs aus Abbildung 48 zusammen mit deren Präfixen gegeben. Auch wenn eine übereinstimmende Schrittsemantik dies nicht sicherstellt, so hat in diesem Fall auch das Netz aus Abbildung 42 diese Prozessablaufsemantik. Die Minimalablaufsemantik der Netze ist durch die zwei BPOs* bpo_1 *und* bpo_2 *aus Abbildung 4 zusammen mit den Präfixen von* bpo_1 *und* bpo_2 *gegeben.*

3.3.3 *Geschichtete Ordnungsstrukturen*

Wie in Unterabschnitt 3.2.2 erläutert, lässt sich in STI-Netzen synchrones Schaltverhalten explizit modellieren, insbesondere lässt es sich von nebenläufigem Schaltverhalten unterscheiden. Um entsprechende Abläufe von Schaltereignissen zu modellieren, reicht eine „früher als“-Beziehung zwischen Ereignissen nicht mehr aus [126, 127, 144]. Daher führen wir in diesem Unterabschnitt eine Verallgemeinerung von BPOs ein, welche sich als Ablaufmodell für STI-Netze eignet.
Die Ordnungsrelation einer BPO modelliert eine „früher als“-Beziehung zwischen Ereignissen. Ungeordnete Ereignisse sind nebenläufig zueinander. Nebenläufige Ereignisse eines Ablaufs können in beliebiger Reihenfolge sowie auch synchron zueinander stattfinden. Nebenläufigkeit und Synchronität können aber nicht unterschieden werden. Mit BPOs kann somit eine Situation, in der zwei Ereignisse

a) nur synchron oder

b) synchron und in einer Reihenfolge, jedoch nicht in der umgekehrten Reihenfolge,

stattfinden können, nicht dargestellt werden (in beiden Fällen a) und b) sind die Ereignisse nicht nebenläufig zueinander aber Synchronität ist möglich). Für solche Situationen wird eine „nicht später als“-Beziehung zwischen Ereignissen eingeführt. Eine „nicht später als“-Beziehung zwischen zwei Ereignissen beschreibt genau die Situation b). Eine symmetrische „nicht später als“-Beziehung zwischen zwei Ereignissen beschreibt die Situation a). Abläufe, in denen die in a) und b) beschriebenen Zusammenhänge vorkommen, modellieren wir dann durch sog. beschriftete geschichtete Ordnungen, welche grob gesagt BPOs ergänzt um eine konsistente „nicht später als“-Relation zwischen den Ereignissen sind. In diesem Abschnitt werden geschichtete Ordnungen (engl. stratified order structures) analog zu [125, 127] eingeführt. Ursprünglich wurden diese unabhängig voneinander in [101, 124] entwickelt, allerdings jeweils unter anderem Namen und mit leicht anderen, aber äquivalenten Axiomen.

Definition 3.3.13 (Beschriftete geschichtete Ordnung)

Definition 3.3.13 (Beschriftete geschichtete Ordnung)
Eine relationale Struktur $\mathrm{go} = (V, \prec, \sqsubset)$ *heißt geschichtete Ordnung, falls folgende Bedingungen für alle* $u, v, w \in V$ *erfüllt sind:*

$$(C1)\ u \not\sqsubset u \qquad (C3)\ u \sqsubset v \sqsubset w \wedge u \neq w \implies u \sqsubset w$$
$$(C2)\ u \prec v \implies u \sqsubset v \qquad (C4)\ u \sqsubset v \prec w \vee u \prec v \sqsubset w \implies u \prec w$$

Eine beschriftete geschichtete Ordnung, kurz BGO, über einer Menge von Beschriftungen T *ist ein Tupel* $\mathrm{bgo} = (V, \prec, \sqsubset, l)$*, wobei* $(V, \prec, \sqsubset)$ *eine geschichtete Ordnung und* $l : V \to T$ *eine Funktion, welche jedem Knoten* $v \in V$ *eine Beschriftung aus* T *zuordnet, ist. Die Funktion* l *heißt Beschriftungsfunktion von* bgo*. Die Menge* $l(V) \subseteq T$ *heißt Beschriftungsmenge von* bgo*.*

In dieser Definition soll $\prec$ die "früher als"-Beziehung und $\sqsubset$ die "nicht später als"-Beziehung darstellen. Es wird auch von Kausalität für $\prec$ und schwacher Kausalität für $\sqsubset$ gesprochen. In Diagrammen wird $\prec$ mit durchgezogenen Pfeilen und $\sqsubset$ mit gestrichelten Pfeilen dargestellt. Die Eigenschaft (C2) besagt, dass, falls $x \prec y$ gilt, auch $x \sqsubset y$ gelten muss. In diesem Fall wird der gestrichelte Pfeil zwischen x und y nicht gezeichnet; außerdem können Pfeile, die sich mit Hilfe von (C3) und (C4) aus den gezeichneten Pfeilen ableiten lassen, weggelassen werden. Es genügt also auch bei geschichteten Ordnungen, nur eine Menge von „Gerüstkanten“ darzustellen.
Zwei BGOs $(V, \prec, \sqsubset, l), (V', \prec', \sqsubset', l')$ sind isomorph, wenn eine bijektive Funktion $\psi : V \to V'$ zwischen den Knotenmengen existiert, welche die Ordnungsrelationen und die Beschriftungsfunktion erhält, d.h. $\forall v, w \in V : v \prec w \iff \psi(v) \prec' \psi(w) \wedge v \sqsubset w \iff \psi(v) \sqsubset' \psi(w) \wedge l(v) = l'(\psi(v))$. Wie im Falle von BPOs unterschieden wir im Weiteren isomorphe BGOs nicht. Wir schreiben also kurz BGO $(V, \prec, \sqsubset, l)$ für die Isomorphieklasse von $(V, \prec, \sqsubset, l)$. Auch hier definieren wir die von einer Teilmenge $V' \subseteq V$ repräsentierte Multimenge von Beschriftungen $|V'|_l = \sum_{t \in T} |\{v \in V' \mid l(v) = t\}| \cdot t$, welche wir wiederum als Parikhvektor von V' bezeichnen.

Definition 3.3.14 (Geschichtete Sprache)

Definition 3.3.14 (Geschichtete Sprache)
Eine Menge $\mathcal{L}$ *von BGOs heißt geschichtete Sprache. Die Vereinigung der Beschriftungsmengen aller in* $\mathcal{L}$ *enthaltener BGOs heißt Beschriftungsmenge von* $\mathcal{L}$*.*

Beispiel

BEISPIEL: *Abbildung 49 illustriert eine geschichtete Sprache. Die BGO* bgo_1 *stellt beispielsweise einen Ablauf dar, bei dem eine Folge eines* a*- und eines* b*-Ereignisses nebenläufig zu sich selbst stattfindet. Zusätzlich findet noch ein* c*-Ereignis nebenläufig zu den* a*-Ereignissen, jedoch nicht später als die* b*-Ereignisse statt.*

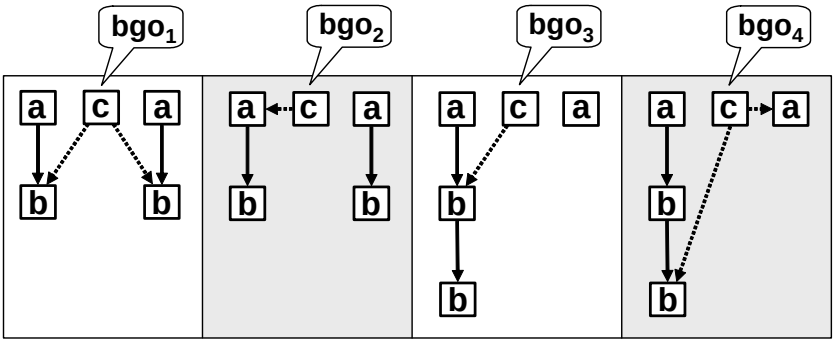

Abbildung 49: Eine geschichtete Sprache.

Im Folgenden werden wir ähnlich wie für partielle Ordnungen die für unsere Zwecke wichtigen Begriffe zu geschichteten Ordnungen einführen, wobei wir alle Begriffe auch analog für BGOs verwenden.

Definition 3.3.15 (Schrittweise linear)

DEFINITION 3.3.15 (SCHRITTWEISE LINEAR)
Eine geschichtete Ordnung $\mathrm{go} = (V, \prec, \sqsubset)$ *heißt schrittweise linear, falls* $\{(v, v') \in V \times V \mid v \not\prec v' \wedge v' \not\prec v\} = (\sqsubset \setminus \prec) \cup \mathrm{id}_V$.

Definition 3.3.16 (Sequentialisierung)

DEFINITION 3.3.16 (SEQUENTIALISIERUNG)
Seien $\mathrm{go} = (V, \prec, \sqsubset)$ *und* $\mathrm{go}' = (V, \prec', \sqsubset')$ *geschichtete Ordnungen. Wir bezeichnen* go′ *als Sequentialisierung (oder auch Erweiterung) von* go*, falls* $\prec \subseteq \prec'$ *und* $\sqsubset \subseteq \sqsubset'$.
Ist go′ *schrittweise linear und eine Sequentialisierung von* go*, so bezeichnen wir* go′ *als Schrittlinearisierung von* go.

Die Menge der Sequentialisierungen einer geschichteten Ordnung go bezeichnen wir durch $\mathrm{Seq}(\mathrm{go})$ und die Menge der Schrittlinearisierungen von go durch $\mathrm{Slin}(\mathrm{go})$.

Definition 3.3.17 (Präfix)

DEFINITION 3.3.17 (PRÄFIX)
Sei $\mathrm{go} = (V, \prec, \sqsubset)$ *eine geschichtete Ordnung.*
Eine Teilmenge $V' \subseteq V$ *heißt abgeschlossen, falls*

$$\forall v, v' \in V : (v \in V' \wedge v' \sqsubset v) \implies v' \in V'.$$

Ist $V' \subseteq V$ *abgeschlossen, so heißt* $\mathrm{go}' = (V', \prec|_{V' \times V'}, \sqsubset|_{V' \times V'})$ *Präfix von* go.
Ein Präfix $\mathrm{go}' = (V', \prec|_{V' \times V'}, \sqsubset|_{V' \times V'})$ *von* go *heißt Präfix von* $u \in V \setminus V'$*, falls* $(u' \prec u \implies u' \in V') \wedge (u \sqsubset u' \implies u' \notin V')$.

Die Menge der Präfixe einer geschichteten Ordnung go bezeichnen wir durch $\mathrm{Pref}(\mathrm{go})$.
Für eine geschichtete Sprache $\mathcal{L}$ definieren wir den Sequentialisierungsabschluss $\mathrm{Seq}(\mathcal{L}) = \bigcup_{\mathrm{bgo} \in \mathcal{L}} \mathrm{Seq}(\mathrm{bgo})$, den Präfixabschluss $\mathrm{Pref}(\mathcal{L}) = \bigcup_{\mathrm{bgo} \in \mathcal{L}} \mathrm{Pref}(\mathrm{bgo})$, den Präfix- und Sequentialisierungsabschluss $\mathrm{PS}(\mathcal{L}) = \mathrm{Pref}(\mathrm{Seq}(\mathcal{L})) = \mathrm{Seq}(\mathrm{Pref}(\mathcal{L}))$, die Menge der Schrittlinearisierungen $\mathrm{Slin}(\mathcal{L}) = \bigcup_{\mathrm{bgo} \in \mathcal{L}} \mathrm{Slin}(\mathrm{bgo})$, den Schrittabschluss $\mathrm{Step}(\mathcal{L}) = \{\mathrm{bgo} \mid$

bgo ist BGO mit Slin(bgo) $\subseteq$ Slin($\mathcal{L}$)} und die Repräsentationssprache Rep($\mathcal{L}$) = {bgo $\in \mathcal{L}$ | bgo $\notin$ PS($\mathcal{L} \setminus$ {bgo})} analog zu partiellen Sprachen.

Beispiel

BEISPIEL: *Die BGOs aus Abbildung 49 sind alle nicht schrittweise linear. Abbildung 50 zeigt eine schrittweise lineare BGO. Diese BGO ist eine Sequentialisierung von* bgo_1 *aus Abbildung 49. Somit ist sie eine Schrittlinearisierung von* bgo_1*. Ein Beispiel für ein Präfix von* bgo_1 *ist auch hier die BGO* $bgo_a = (\{v_a\}, \emptyset, \emptyset, (v_a, a))$*, welche nur aus einem mit* a *beschrifteten Knoten besteht.*

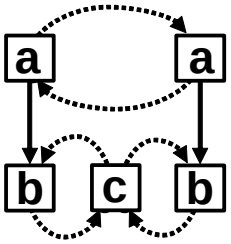

Abbildung 50: Eine Schrittlinearisierung von bgo_1 aus Abbildung 49.

3.3.4 *Abläufe von Stellen/Transitions-Netzen mit gewichteten Inhibitorkanten*

In diesem Unterabschnitt wird formal definiert, unter welchen Bedingungen eine BGO einen Ablauf eines STI-Netzes repräsentiert. In der Literatur wurde ein Ablaufbegriff für STI-Netze (sowie generell für Petrinetze mit Inhibitorkanten [127, 143, 144]) bisher nur in Form spezieller Prozessnetze, sog. Aktivatorprozesse, eingeführt [144]. In [135, 136] haben wir Abläufe von STI-Netzen in der Form von aktivierten BGOs definiert, indem wir den Begriff der aktivierten BPO geeignet verallgemeinert haben. Weiter haben wir in [135, 136] gezeigt, dass der bisherige Ablaufbegriff in der Form von Aktivatorprozessen nicht vollständig ist, in dem Sinne dass Aktivatorprozesse nicht jedes mögliche Schaltverhalten von STI-Netzen repräsentieren können. Daher haben wir die Definition von Aktivatorprozessen derart verallgemeinert, dass Vollständigkeit gewährleistet ist. Wir haben dann gezeigt, dass ähnlich wie bei S/T-Netzen, der Begriff der aktivierten BGO konsistent zu der verallgemeinerten Definition von Aktivatorprozessen ist. In dieser Arbeit beschränken wir uns auf das Konzept der aktivierten BGO zur Modellierung von Abläufen von STI-Netzen.
Schrittweise lineare BGOs definieren Schrittfolgen. Sei T eine Menge von Transitionen und bgo $= (V, \prec, \sqsubset, l)$ eine BGO über T. Ist bgo schrittweise linear, so gilt o.B.d.A. $V = \bigcup_{i=1}^{n} V_i$ sowie $\prec = \bigcup_{i<j} V_i \times V_j$ und $\sqsubset = ((\bigcup_i V_i \times V_i) \cup \prec) \setminus id_V$. Die Schrittfolge $\sigma(bgo) = |V_1|_l \dots |V_n|_l$ heißt in diesem Fall zu bgo assoziiert. Jede Schrittfolge ist zu einer (eindeutigen) BGO assoziiert.
Mit dem Begriff der assoziierten Schrittfolge lässt sich nun Aktiviertheit von BGOs analog zu BPOs definieren.

Definition 3.3.18 (Aktivierte BGO)

Definition 3.3.18 (Aktivierte BGO)
Sei (N, m_0), $N = (P, T, W, I)$, *ein markiertes STI-Netz. Eine BGO* $bgo = (V, \prec, \sqsubset, l)$ *mit* $l : V \to T$ *heißt aktiviert in* m_0 *bzgl.* N, *falls für jede Schrittlinearisierung* $bgo' \in Slin(bgo)$ *die assoziierte Schrittfolge* $\sigma(bgo')$ *in* m_0 *aktiviert ist.*
Ist eine BGO bgo *bzgl.* (N, m_0) *aktiviert, so kann sie schalten. Ihr Schalten überführt* m_0 *in die Folgemarkierung* m' *gegeben durch* $m'(p) = m_0(p) + \sum_{v \in V}(W(l(v), p) - W(p, l(v)))$ *für* $p \in P$.

Es lässt sich wie bei S/T-Netzen zeigen, dass jede Sequentialisierung und jedes Präfix einer aktivierten BGO aktiviert ist.

Definition 3.3.19 (Ablaufsemantik)

Definition 3.3.19 (Ablaufsemantik)
Die geschichtete Sprache $\mathfrak{ABgo}(N, m_0)$ *aller in* m_0 *bzgl.* N *aktivierten BGOs heißt Ablaufsemantik eines markierten STI-Netzes* (N, m_0).

Beispiel

Beispiel: *Wir betrachten die BGO* bgo_1 *aus Abbildung 49 und das markierte STI-Netz aus Abbildung 43. In diesem Netz kann insbesondere die Schaltfolge* ab *nebenläufig zu sich selbst schalten. Außerdem kann* c *einmal schalten. Dies ist allerdings nur möglich, solange nicht drei Schaltereignisse von* a *und* b *stattgefunden haben, da jedes dieser Ereignisse die Anzahl der Marken in* p_3 *um eins erhöht. Somit kann ein* c*-Ereignis also im Besonderen nebenläufig zu den zwei* a*-Ereignissen stattfinden, wenn es nicht später als die zwei* b*-Ereignisse stattfindet. Diese Überlegungen zeigen, dass* bgo_1 *bzgl. des betrachteten Netzes aktiviert ist.*
Eine formale Überprüfung der Aktiviertheit erfordert die Untersuchung aller Schrittlinearisierungen von bgo_1. *Beispielsweise ist zur Schrittlinearisierung aus Abbildung 50 die Schrittfolge* $2a2b + c$ *assoziiert, welche in dem betrachteten Netz aktiviert ist.*
Die Ablaufsemantik des Netzes aus Abbildung 43 ist durch den Präfix- und Sequentialisierungsabschluss der geschichteten Sprache aus Abbildung 49 gegeben.

3.4 LINEARE UNGLEICHUNGSSYSTEME

Bei den in dieser Arbeit behandelten Fragestellungen spielt insbesondere das Lösen linearer Ungleichungssysteme eine wichtige Rolle. Eine lineare Ungleichung ist durch einen Koeffizientenvektor $\mathbf{a}$, einen Variablenvektor $\mathbf{x}$ und einen Konstantenskalar b sowie eines der Ungleichheitszeichen $<, \leqslant, >, \geqslant$ gegeben. Wir stellen lineare Ungleichungen in der Form $\mathbf{a}^t \cdot \mathbf{x} \geqslant b$ dar. Ein System aus linearen Ungleichungen mit demselben Ungleichheitszeichen ist entsprechend durch eine Koeffizientenmatrix $\mathbf{A}$, einen Variablenvektor $\mathbf{x}$ und einen Konstantenvektor $\mathbf{b}$ sowie eines der Ungleichheitszeichen $<, \leqslant, >, \geqslant$ gegeben. Ein solches System wird in der Form $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ dargestellt. Eine Ungleichung der Form $\mathbf{a}^t \cdot \mathbf{x} \geqslant b$ bzw. ein Ungleichungssystem der Form $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ ist lösbar über einer Menge A, falls ein Lösungsvektor $\bar{\mathbf{x}}$ über A existiert mit $\mathbf{a}^t \cdot \bar{\mathbf{x}} \geqslant b$ bzw. $\mathbf{A} \cdot \bar{\mathbf{x}} \geqslant \mathbf{b}$. Eine Ungleichung $\mathbf{a}^t \cdot \mathbf{x} \geqslant b$ bzw. ein Ungleichungssystem $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ heißt homogen, falls $b = 0$ bzw. $\mathbf{b} = \mathbf{0}$. Für lineare Gleichungen und Gleichungssysteme werden entsprechende Bezeichnungen verwendet. Da sich eine lineare Gleichung $\mathbf{a}^t \cdot \mathbf{x} = b$ durch zwei lineare Ungleichungen $\mathbf{a}^t \cdot \mathbf{x} \geqslant b$ und $-\mathbf{a}^t \cdot \mathbf{x} \geqslant -b$ ausdrücken lässt, sprechen wir auch im Falle eines aus linearen Ungleichungen und Gleichungen bestehenden Systems kurz von einem Ungleichungssystem.

Wir sind vor allem an nicht-negativen ganzzahligen Lösungen homogener linearer Ungleichungssysteme interessiert. Hierbei spielt sowohl die Suche nach einzelnen Lösungsvektoren als auch nach der Menge aller Lösungsvektoren eines solchen Systems eine Rolle.
Ein Ungleichungssystem der Form $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ über dem Körper der reellen Zahlen definiert ein sog. Polyeder. Ist das Ungleichungssystem homogen, so definiert es ein Polyeder von einem speziellen Typ, einen sog. polyedrischen Kegel, vgl. [207, 46, 160, 212, 11, 100].

DEFINITION 3.4.1 (POLYEDER) *Definition 3.4.1 (Polyeder)*
Die Lösungsmenge eines linearen Ungleichungssystems $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ über den reellen Zahlen heißt Polyeder.

DEFINITION 3.4.2 (POLYEDRISCHER KEGEL) *Definition 3.4.2 (Polyedrischer Kegel)*
Eine nicht-leere Menge K von Vektoren über den reellen Zahlen heißt Kegel, wenn für alle $\mathbf{x} \in K$ und nicht-negative reelle Zahlen α auch $\alpha\mathbf{x} \in K$. Ein Kegel, welcher ein Polyeder ist, heißt polyedrisch.

LEMMA 3.4.1 *Lemma 3.4.1*
Eine Menge K von Vektoren über den reellen Zahlen ist genau dann ein polyedrischer Kegel, wenn sie die Lösungsmenge eines homogenen linearen Ungleichungssystems $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{0}$ über den reellen Zahlen ist.

Ein Polyeder $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ wird rational genannt, falls $\mathbf{A}$ und $\mathbf{b}$ nur rationale Einträge haben. Die Dimension eines Polyeders ist die Dimension des kleinsten affinen Raumes, welcher das Polyeder enthält. Polyeder sind konvex und abgeschlossen. Eine Ecke oder auch Extrempunkt eines Polyeders ist ein Punkt, welcher nicht durch eine Konvexkombination zweier anderer Punkte des Polyeders darstellbar ist.
Jeder Kegel K enthält den Nullvektor $\mathbf{0}$. Falls $K \cap \{-\mathbf{x} \mid \mathbf{x} \in K\} = \{\mathbf{0}\}$, so heißt K spitz. Die einzig mögliche Ecke eines polyedrischen Kegels ist $\mathbf{0}$. Ein polyedrische Kegel $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{0}$ ist genau dann spitz, wenn $\mathbf{0}$ eine Ecke des Kegels ist. Es lässt sich zeigen, dass dies äquivalent dazu ist, dass der Rang von $\mathbf{A}$ der Stelligkeit von $\mathbf{x}$ entspricht. Die Dimension eines Kegels K ist die maximale Anzahl linear unabhängiger Vektoren von K. Eine wichtige Klasse von konvexen, abgeschlossenen Kegeln sind die sog. endlich erzeugten Kegel.

DEFINITION 3.4.3 (ENDLICH ERZEUGTER KEGEL) *Definition 3.4.3 (Endlich erzeugter Kegel)*
Seien $\mathbf{x}_1, \ldots, \mathbf{x}_n$ reelle Vektoren gleicher Stelligkeit. Dann heißt

$$\{\sum_{i=1}^{n} \lambda_i \mathbf{x}_i \mid \lambda_1, \ldots, \lambda_n \geqslant 0\}$$

der von $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ endlich erzeugte Kegel. Die Menge $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ heißt endliches Erzeugendensystem.

Nach dem Satz von Farkas-Minkowski-Weyl ist jeder polyedrische Kegel endlich erzeugt und vice versa ([207, 46, 160, 212, 100]).

SATZ 3.4.1 *Satz 3.4.1*
Ein Kegel ist genau dann polyedrisch, wenn er endlich erzeugt ist.

Ein Element $\mathbf{x} \neq \mathbf{0}$ eines Kegels K definiert einen sog. Strahl $\{\alpha\mathbf{x} \mid \alpha \geqslant 0\}$ des Kegels. Ein Strahl $\{\alpha\mathbf{x} \mid \alpha \geqslant 0\}$ heißt Extremalstrahl, wenn sich $\mathbf{x}$ nicht als Summe von zwei nicht auf dem Strahl liegenden Vektoren des Kegels darstellen lässt. Ein polyedrischer Kegel $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{0}$ hat nur endlich viele Extremalstrahlen. Jedes endliche Erzeugendensystem eines polyedrischen Kegels enthält für jeden Extremalstrahl einen Vektor, der den Extremalstrahl definiert.

Beispiel

BEISPIEL: *In Abbildung 51 sind zwei lineare Ungleichungssysteme und deren Lösungsmengen dargestellt. Das linke inhomogene Ungleichungssystem definiert ein dreidimensionales beschränktes rationales Polyeder. Das Polyeder ist ein nichtregulärer Tetraeder mit den Eckpunkten* $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ *und* $(0.5,0.5,0.5)$. *Das rechte Ungleichungssystem ist homogen und definiert daher einen polyedrischen Kegel. Der betrachtete Kegel ist spitz. Die Dimension des Kegels ist drei. Die Vektoren* $(1,0,0)$, $(1,1,0)$, $(1,1,1)$ *und* $(1,0,1)$ *definieren die vier Extremalstrahlen des Kegels. Die vier Vektoren bilden zusammen ein endliches Erzeugendensystem des Kegels.*

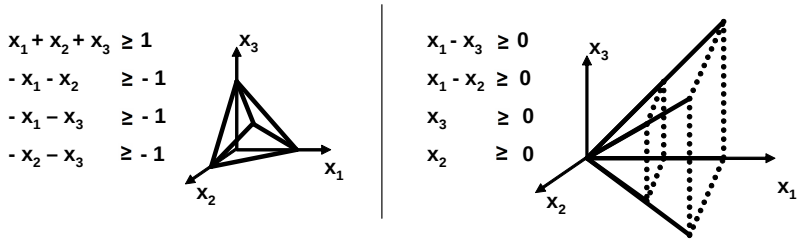

Abbildung 51: Zwei lineare Ungleichungssysteme und deren Lösungsmengen.

In dieser Arbeit sind wir an minimalen endlichen Erzeugendensystemen rationaler spitzer polyedrischer Kegel interessiert. Wir bezeichnen ein endliches Erzeugendensystem eines polyedrischen Kegels als minimal, wenn keine echte Teilmenge des Erzeugendensystems auch ein endliches Erzeugendensystem des Kegels ist. Offensichtlich gibt es zu jedem polyedrischen Kegel ein minimales endliches Erzeugendensystem. Auch wenn wir Vektoren, welche auf einem Strahl liegen, identifizieren, gibt es im Allgemeinen aber kein eindeutiges minimales endliches Erzeugendensystem. Für spitze polyedrische Kegel gilt jedoch, dass eine Menge, welche genau für jeden Extremalstrahl einen Vektor enthält, der den Extremalstrahl definiert, ein endliches Erzeugendensystem ist. Dieses Erzeugendensystem ist das bis auf die Wahl der Vektoren auf den Extremalstrahlen eindeutige minimale endliche Erzeugendensystem eines spitzen polyedrischen Kegels. Ist der Kegel rational, so können die Vektoren, welche die Extremalstrahlen definieren, über der Menge der ganzen Zahlen gewählt werden. Die Anzahl der Extremalstrahlen eines spitzen polyedrischen Kegels ist im schlechtesten Fall in $\mathcal{O}(m^{\lfloor d/2 \rfloor})$ (diese obere Grenze wird auch erreicht), wobei m die Anzahl der Zeilen von $\mathbf{A}$ und d die Dimension des Kegels ist [170, 11]. In vielen Beispielen von Kegeln ergibt sich allerdings eine im Gegensatz zu dieser oberen Schranke sehr geringe Zahl an Extremalstrahlen [11, 212].
Das Problem die Extremalstrahlen eines rationalen spitzen polyedrischen Kegels $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{0}$ zu berechnen ist als Extremalstrahl-Aufzählungs-Problem bekannt [160, 212, 100]. Dieses Problem wurde direkt oder indirekt in vielen Arbeiten behandelt [207, 160, 212, 100, 11, 46, 163]. Offensichtlich ist die Effizienz von Lösungsverfahren durch die im Allgemeinen exponentiell von der Größe von $\mathbf{A}$ abhängende Anzahl an Extremalstrahlen beschränkt. Ein wichtiges theoretisches Ergebnis zeigt, dass es ein Lösungsverfahren gibt, dessen Komplexität analog zur oberen Schranke für die Anzahl der Extremalstrahlen $\mathcal{O}(m^{\lfloor d/2 \rfloor})$ (für $d \geqslant 4$) beträgt [56, 11]. Allerdings ergibt sich in fast allen praktischen Beispielen eine gegenüber der oberen Schranke wesentlich geringere Anzahl an Extremalstrahlen. Dementsprechend ist es im Hinblick auf

praktische Anwendbarkeit wichtig, Verfahren zu entwickeln, welche in solchen Fällen eine entsprechend gutartige Laufzeit haben. Es wurde also versucht, Lösungsverfahren zu entwickeln, welche polynomiellen Zeitaufwand in Abhängigkeit von der Größe der Eingabematrix zusammen mit der Anzahl der Ausgabe-Extremalstrahlen benötigen. Dies ist allerdings nur für Spezialfälle gelungen. Ob es im Allgemeinen ein entsprechendes Lösungsverfahren gibt, ist ein bisher ungelöstes Problem.
Den wichtigsten Spezialfall stellen sog. nicht-degenerierte Probleme dar, bei denen es grob gesagt für jeden Extremalstrahl eindeutig $d-1$ Ungleichungen, eine sog. Basis von Ungleichungen, des Systems $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{0}$ gibt, so dass sich der Extremalstrahl ergibt, indem für diese Ungleichungen Gleichheit gefordert wird. Nicht-degenerierte Probleme lassen sich mit sog. Pivotisierungsalgorithmen in polynomieller (normalerweise linearer) Zeit in der Anzahl der Extremalstrahlen lösen [160, 212, 11, 100]. Pivotisierungsalgorithmen werden auch als Graph-Traversierungs-Algorithmen bezeichnet. Im Allgemeinen suchen Graph-Traversierungs-Algorithmen grob gesagt zuerst einen Knoten eines Graphen und versuchen dann, durch geeignete Traversierung alle Knoten eines Graphen zu berechnen. In unserer Situation wird ähnlich vorgegangen, um die Extremalstrahlen eines Kegels aufzuzählen. Wie gesagt, können die Extremalstrahlen eines Kegels durch Basen von Ungleichungen identifiziert werden. Ausgehend von einer Anfangsbasis werden mit einer Pivotisierungsoperation, ähnlich wie sie auch im Simplex-Algorithmus vorkommt, sukzessive weitere Basen durch einen Austausch der Ungleichungen der Basen berechnet. Auf diese Weise lassen sich alle Extremalstrahlen auffinden. In die Klasse der Pivotisierungsalgorithmen fallen der „Geschenkverpacken-Algorithmus“ von Chand und Kapur, Seidels Algorithmus und der bekannte „Rückwärtssuch-Algorithmus“ von Avis und Fukuda. Im allgemeinen Fall degenerierter Probleme, welcher bei den meisten praktischen Anwendungen auftritt, weisen all diese Verfahren eine exponentielle Laufzeit in Abhängigkeit von der Größe der Eingabematrix und der Anzahl der Ausgabe-Extremalstrahlen auf.
Neben den Pivotisierungsalgorithmen gibt es zur Lösung des Extremalstrahl-Aufzählungs-Problems noch sog. Einfügealgorithmen [160, 212, 11]. Diese Algorithmen berechnen inkrementell die Strahlen von Relaxationen des ursprünglichen Kegels. Die Grundidee ist, mit einer Relaxation in einer sehr einfachen Form zu beginnen. Für diese lassen sich unmittelbar Strahlen, welche ein endliches Erzeugendensystem festlegen, angeben. Dann werden zur Ausgangs-Relaxation sukzessive Restriktionen hinzugefügt bis sich der ursprüngliche Kegel ergibt. Beim Hinzufügen einer Restriktion wird jeweils ein endliches Erzeugendensystem der neuen Relaxation aus den betrachteten Strahlen der vorigen Relaxation berechnet. Hierbei wird ein Teil der Strahlen beibehalten, ein Teil der Strahlen fällt weg und es entstehen einige neue Strahlen. Theoretisch kann die Anzahl der Strahlen im schlechtesten Fall in einem einzigen solchen Schritt quadratisch zunehmen. Daher können die notwendigen Berechnungen von Einfügealgorithmen sehr aufwändig sein. Normalerweise wächst die Anzahl der Strahlen aber wesentlich langsamer als im schlechtesten Fall. Dadurch, dass ein Teil der Strahlen in jedem Schritt wegfällt, bewirken einige Schritte typischerweise auch eine Reduktion der Anzahl der Strahlen. Gerade aus diesem Grund ist die Komplexität von Einfügealgorithmen bisher nicht

gut verstanden [160, 212, 11, 100]. Es ist unklar, wie sich die Anzahl der Strahlen entwickelt. Einerseits kann die Anzahl an zwischenzeitlich berechneten Strahlen exponentiell größer sein als die endgültige Anzahl an Extremalstrahlen. Andererseits kann die Entwicklung der Strahlenanzahlen bei einer anderen Reihenfolge des Hinzufügens der Restriktionen wieder vollkommen anders verlaufen. Es gibt daher viele Möglichkeiten, durch entsprechende Heuristiken und „geschickte“ Implementierungen in den meisten Fällen eine gutartige Laufzeit in Abhängigkeit von der Ausgabe zu erreichen. Im schlechtesten Fall ergibt sich jedoch für alle bekannten Einfügealgorithmen eine exponentielle Laufzeit sowohl in Abhängigkeit von der Eingabematrix als auch von der Anzahl der Ausgabestrahlen. Das wichtigste und älteste Verfahren zur Lösung des Extremalstrahl-Aufzählungs-Problems ist der unter der Bezeichnung „doppelte Beschreibungsmethode“ bekannte Einfügealgorithmus von Motzkin et al. Weitere bedeutsame Einfügealgorithmen sind der Cernikova-Algorithmus, die „darunter und darüber-Methode“ von Seidel, der randomisierte Algorithmus von Clarkson und Shor, der derandomisierte Algorithmus von Chazelle und die Fourier-Motzkin-Elimination. Die ersten vier Verfahren sind jedoch sehr ähnlich zur „doppelten Beschreibungsmethode". Die Fourier-Motzkin-Elimination ist ein allgemeineres Verfahren als die „doppelte Beschreibungsmethode". Insbesondere zur „doppelten Beschreibungsmethode", aber auch zu anderen Verfahren, gibt es etliche Verbesserungen, Implementierungsstrategien und Varianten der Verfahren, die dann meist für spezielle Problemklassen sehr effizient sind.
Generell gilt, dass bei nicht-degenerierten Problemen oder Problemen mit einem geringen Grad an Degeneration Pivotisierungsalgorithmen sinnvoll sind. Ansonsten haben Einfügealgorithmen normalerweise eine bessere Performance [160]. Eine allgemeine Eigenschaft von Extremalstrahl-Aufzählungs-Verfahren ist, dass sie sehr sensitiv bzgl. Implementierungsdetails sind, in dem Sinne, dass kleine Variationen der Algorithmen einen sehr großen Laufzeiteffekt haben können.

Beispiel

BEISPIEL: *Da der polyedrische Kegel aus Abbildung 51 rechts spitz ist, bilden die vier Extremalstrahlen das eindeutige minimale endliche Erzeugendensystem des Kegels. Jeder Extremalstrahl ergibt sich als Schnitt zweier eindeutiger Seitenflächen des Kegels. Daher ist der Kegel nicht-degeneriert. Zur Berechnung der Extremalstrahlen aus dem gegebenen Ungleichungssystem wäre in diesem Fall also ein Pivotisierungsalgorithmus die beste Wahl.*
Abbildung 52 zeigt, wie sich der Kegel aus Abbildung 51 rechts durch das Hinzufügen der Restriktion $-x_1 + 2x_2 \geqslant 0$ verändert. Zwei Extremalstrahlen des ursprünglichen Kegels sind auch wieder Extremalstrahlen des neuen Kegels, zwei der alten Extremalstrahlen fallen weg (grau) und es entstehen zwei neue Extremalstrahlen. Die neuen Extremalstrahlen liegen jeweils auf dem Schnitt einer von zwei alten Extremalstrahlen aufgespannten Ebene mit der durch die neue Restriktion definierten Ebene. Diese Illustration zeigt das Vorgehen von Einfügealgorithmen.

Eine zweite in dieser Arbeit wichtige Fragestellung ist die Berechnung eines rationalen Vektors, welcher Element eines gegebenen rationalen Polyeders ist. Dieses Problem ist als lineares Zulässigkeitsproblem bekannt [207, 214, 148]. Ein Lösungsverfahren für das lineare Zulässigkeitsproblem muss entscheiden, ob ein rationales Polyeder $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ nicht-leer ist, und im positiven Fall einen sog. zulässigen rationalen Vektor $\bar{\mathbf{x}}$ mit $\mathbf{A} \cdot \bar{\mathbf{x}} \geqslant \mathbf{b}$ berechnen.

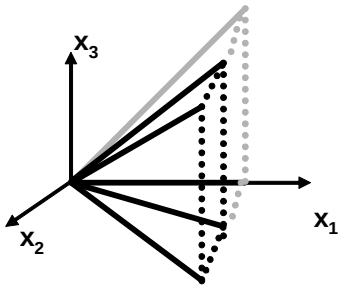

Abbildung 52: Illustration zu Einfügealgorithmen.

Bei dem klassischen linearen Optimierungsproblem, welches in vielen Lehrbüchern und Artikeln, z.B. [207, 214, 148], ausführlich dargestellt ist, wird ein rationales Polyeder $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ und eine lineare Zielfunktion $\min! \mathbf{c}^t \cdot \mathbf{x}$ betrachtet. Bei diesem Problem wird ein zulässiger Vektor $\bar{\mathbf{x}}$ gesucht, welcher die Zielfunktion minimiert, d.h. $\mathbf{c}^t \cdot \bar{\mathbf{x}} \leqslant \mathbf{c}^t \cdot \bar{\mathbf{y}}$ für alle zulässigen Vektoren $\bar{\mathbf{y}}$. Es gibt hierbei drei Möglichkeiten:

- Es gibt keinen zulässigen Vektor, d.h. das Polyeder ist leer.
- Es gibt zulässige Vektoren, jedoch kann die Zielfunktion für diese beliebig kleine Werte annehmen (dies ist nur bei einem unbeschränkten Polyeder möglich).
- Es gibt optimale zulässige Vektoren, die dann alle auf einer gemeinsamen Seitenfläche des Polyeders liegen (daher beschränkt sich die Suche nach optimalen zulässigen Vektoren normalerweise auf die Ecken des Polyeders).

Das lineare Zulässigkeitsproblem und das lineare Optimierungsproblem sind polynomiell äquivalent. Das lineare Optimierungsproblem für $\mathbf{c} = \mathbf{0}$ entspricht dem Zulässigkeitsproblem, da in diesem Fall alle zulässigen Vektoren optimal sind. Umgekehrt lässt sich das lineare Optimierungsproblem mit einem speziellen binären Suchverfahren polynomiell auf das lineare Zulässigkeitsproblem reduzieren.

Beispiel

BEISPIEL: *Betrachten wir das lineare Zulässigkeitsproblem für das Polyeder aus Abbildung 51 links, so sind beispielsweise die vier Eckpunkte zulässig, genauso aber auch alle anderen Punkte des Polyeders wie der Vektor* $(0, 0.5, 0.5)$. *Ergänzen wir das Problem durch die Zielfunktion* $\min! -x_1$ *zu einem linearen Optimierungsproblem, so ist nur der Vektor* $(1, 0, 0)$ *eine Lösung. Im Falle der Zielfunktion* $\min! x_1$ *ergeben sich die Vektoren* $(0, 1, 0)$ *und* $(0, 0, 1)$ *sowie alle Punkte auf der Verbindungsstrecke dieser zwei Punkte wie der Punkt* $(0, 0.5, 0.5)$ *als Lösungen. Betrachten wir den Kegel aus Abbildung 51 rechts zusammen mit der Zielfunktion* $\min! -x_1$, *so gibt es keinen optimalen zulässigen Vektor.*

Das in der Praxis bedeutendste Verfahren, um lineare Optimierungsprobleme und damit auch lineare Zulässigkeitsprobleme zu lösen, ist der von Dantzig entwickelte Simplexalgorithmus [61, 62]. Die geometrische Grundidee des Algorithmus besteht darin, von einer beliebigen Ecke des Polyeders entlang seiner Kanten zu einer optimalen Ecke zu gelangen. Eine wesentliche Schwierigkeit hierbei stellen sog. degenerierte Ecken dar, welche mehr Ungleichungen des Systems $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{0}$ mit Gleichheit erfüllen als die Stelligkeit von $\mathbf{x}$ beträgt. Der Simplexalgorithmus setzt sich aus zwei Phasen zusammen. Phase I bestimmt einen

zulässigen Startvektor oder stellt fest, dass das Polyeder leer ist. Phase I wird durch Konstruktion eines Hilfsproblems, welches selbst ein lineares Optimierungsproblem ist, bei dem aber schon ein zulässiger Startvektor bekannt ist, gelöst. Phase II verbessert einen bestehenden zulässigen Vektor, bis keine Verbesserung der Zielfunktion mehr möglich ist oder Unbeschränktheit des Problems festgestellt wird. Zur Lösung eines Zulässigkeitsproblems wird offensichtlich nur Phase I benötigt.
Der Simplexalgorithmus hat im schlechtesten Fall exponentielle Laufzeit. Dies trifft auch schon auf Phase I des Algorithmus zu. Der Grund hierfür liegt darin, dass ein Polyeder exponentiell viele Ecken in der Größe der definierenden Matrix besitzen kann. Es lässt sich zeigen, dass es Beispiele gibt, bei denen der Simplex-Algorithmus alle exponentiell vielen Ecken eines Polyeders abläuft. In den meisten Fällen werden allerdings nur wenige Ecken besucht. In der Praxis hängt die Laufzeit des Simplexalgorithmus oft linear von der Größe der Matrix $\mathbf{A}$ ab [207]. Es lässt sich auch formal nachweisen, dass Probleme, bei denen der Simplexalgorithmus exponentielle Laufzeit benötigt, extrem selten sind und dass die durchschnittliche Laufzeit des Simplexalgorithmus sehr gut ist [42]. Es wurden viele Verbesserungen und Varianten des Simplexalgorithmus entwickelt, jedoch noch keine, welche polynomielle Laufzeit für alle Probleminstanzen aufweist. Einige Varianten benötigen allerdings für bestimmte Klassen von linearen Optimierungsproblemen nur polynomielle Laufzeit. Eine wichtige Eigenschaft des Simplexalgorithmus ist, dass er bei leichter Veränderung des Problems, beispielsweise dem Hinzufügen einer zusätzlichen Ungleichung, einen „Warmstart“ von einer zuvor berechneten Lösung durchführen kann und dann meist nur wenige Iterationen zur erneuten Lösung benötigt. Derartige inkrementelle Simplexverfahren können in der Praxis häufig angewendet werden.
Khachiyan konnte zeigen, dass das lineare Optimierungsproblem allgemein in polynomieller Zeit lösbar ist [207, 214, 62]. Er entwickelte mit der Ellipsoidmethode einen polynomiellen Algorithmus zur Lösung des linearen Zulässigkeitsproblems (Komplexität $\mathcal{O}(n^6)$). Bei diesem Verfahren werden iterativ Ellipsoide konstruiert, welche das gegebene Polyeder enthalten. In jedem Schritt wird geprüft, ob das Zentrum des Ellipsoides im Polyeder liegt und damit ein zulässiger Vektor ist. Wird auf diese Weise nach einer bestimmten, polynomiell von der Eingabe abhängigen, Anzahl von Iterationen kein zulässiger Vektor gefunden, so lässt sich folgern, dass es keinen zulässigen Vektor gibt. Probabilistische Resultate und experimentelle Tests haben jedoch gezeigt, dass der durchschnittliche Rechenaufwand der Ellipsoidmethode im Vergleich zum Simplexalgorithmus sehr hoch ist. Daher ist die Ellipsoidmethode in der Praxis nicht brauchbar.
Der erste polynomielle Algorithmus zur Lösung linearer Optimierungsprobleme, der auch in der Praxis schnelle Laufzeiten aufweist, ist das von Karmarkar entwickelte Innere-Punkte-Verfahren [137] (Komplexität $\mathcal{O}(n^{3,5})$). Wie bei dem Simplexalgorithmus bewegt sich das Innere-Punkte-Verfahren von zulässigem Vektor zu zulässigem Vektor bis ein optimaler zulässiger Vektor erreicht ist. Im Gegensatz zum Simplexalgorithmus wird aber kein Pfad über die Ecken des Polyeders gesucht, sondern es wird das Innere des Polyeders durchquert. Einige entscheidende Erweiterungen und Verbesserungen des ursprünglichen Innere-Punkte-Verfahrens machten es zu einer echten Konkurrenz zum

Simplexalgorithmus. Dennoch bleibt der Simplexalgorithmus für die meisten praktischen Anwendungen die erste Wahl.

BEISPIEL: *Abbildung 53 zeigt Illustrationen der verschiedenen Lösungsverfahren für lineare Optimierungsprobleme (diese sind jeweils übernommen aus* `www.wikipedia.de`*).* *Beispiel*
Es ist zu beachten, dass bei dem Polyeder aus Abbildung 51 links alle Ecken nicht-degeneriert sind, da sie jeweils nur drei Ungleichungen mit Gleichheit erfüllen. Die Ecke $(0,0,0)$ *des Kegels rechts in Abbildung 51 erfüllt dahingegen vier Ungleichungen mit Gleichheit und ist daher degeneriert.*

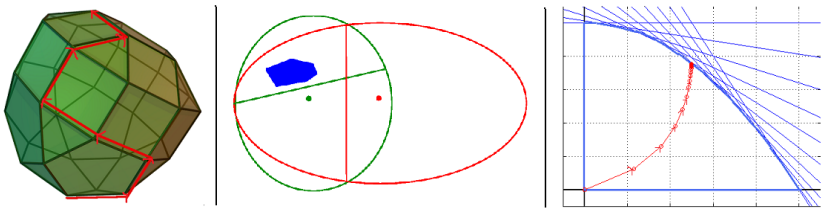

Abbildung 53: Illustration des Simplex-Verfahrens (dreidimensional), der Ellipsoidmethode (zweidimensional) und des Algorithmus von Karmarkar (zweidimensional).

Eigentlich sind wir in dieser Arbeit meist an einer ganzzahligen Lösung eines linearen Ungleichungssystems $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ interessiert. Die Fragestellung, ob ein gegebenes rationales lineares Ungleichungssystem eine ganzzahlige Lösung besitzt, zusammen mit einer anschließenden Berechnung einer entsprechenden Lösung, wird als ganzzahliges Zulässigkeitsproblem bezeichnet [207, 148, 62]. Erfüllt das gegebene Ungleichungssystem die Eigenschaft $\mathbf{b} \geqslant \mathbf{0}$, so genügt es, nach einem rationalen Lösungsvektor zu suchen. In diesem Fall führt nämlich die Multiplikation eines rationalen Lösungsvektors mit einer rationalen Zahl größer als Eins wieder zu einem rationalen Lösungsvektor. Somit ergibt in dieser Situation die Multiplikation eines rationalen Lösungsvektors mit dem Hauptnenner der Einträge des Lösungsvektors einen ganzzahligen Lösungsvektor des Ungleichungssystems. Gibt es keinen rationalen Lösungsvektor, so gibt es natürlich auch keinen ganzzahligen Lösungsvektor. Daher können wir das ganzzahlige Zulässigkeitsproblem in dem beschriebenen Fall mit den oben vorgestellten effizienten Verfahren zur Lösung des normalen Zulässigkeitsproblems lösen.
Erfüllt ein Ungleichungssystem $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ allerdings die Eigenschaft $\mathbf{b} \geqslant \mathbf{0}$ nicht, so ist eine effiziente Berechnung eines ganzzahligen Lösungsvektors nur in Spezialfällen möglich. Im Allgemeinen lässt sich zeigen, dass das ganzzahlige Zulässigkeitsproblem NP-vollständig ist [207, 148]. Zur Lösung des Problems lassen sich Verfahren zur ganzzahligen linearen Optimierung verwenden [207, 148, 62], wobei sich das Zulässigkeitsproblem wie im rationalen Fall als Spezialfall des Optimierungsproblems mit der Nullfunktion als Zielfunktion ergibt.
Bei der ganzzahligen linearen Optimierung wird nach einem ganzzahligen Punkt eines rationalen Polyeders $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ gesucht, welcher eine gegebene lineare Zielfunktion $\min! \mathbf{c}^t \cdot \mathbf{x}$ optimiert. Natürlich ist auch dieses Problem NP-vollständig. Es gibt allerdings einige spezielle Klassen von ganzzahligen linearen Optimierungsproblemen, hauptsächlich basierend auf geeigneten Unimodularitätsbedingungen an $\mathbf{A}$, welche

sich effizient lösen lassen. Zur allgemeinen Lösung ganzzahliger linearer Optimierungsprobleme gibt es einerseits exakte Lösungsverfahren, die normalerweise auf der Lösung vieler ähnlicher rationaler linearer Optimierungsprobleme basieren, und andererseits eine Vielzahl von Heuristiken [207, 148, 62].
Die wichtigsten exakten Verfahren sind Schnittebenenverfahren und Branch-and-Bound-Verfahren. Das ursprüngliche Schnittebenenverfahren von Gomory stellt den ersten Algorithmus zur Lösung ganzzahliger Optimierungsprobleme dar [148]. Branch-and-Bound-Verfahren wurden wenig später von Land und Doig eingeführt [149]. Beide Vorgehensweisen lösen zunächst die sog. LP-Relaxierung des Optimierungsproblems, bei der die Ganzzahligkeitsbedingung weggelassen wird. Dies liefert eine duale Schranke für den Optimalwert. Beim Schnittebenenverfahren wird diese Schranke durch schrittweises Hinzufügen sog. Schnittebenen, die zusätzliche Ungleichungen definieren, verschärft. Durch iteratives Lösen der entsprechenden LP-Relaxierungen lässt sich das ganzzahlige Optimierungsproblem lösen. Bei Branch-and-Bound-Verfahren wird das Problem derart in zwei oder mehr Teilprobleme zerlegt, dass jede zulässige Lösung in einem dieser Teilprobleme enthalten ist. Auf diese Art wird ein Verzweigungsbaum mit der LP-Relaxierung als Wurzelknoten aufgebaut. Durch diesen Baum lassen sich alle möglichen Lösungen aufzählen. Mit Hilfe der dualen Schranken, die durch die Lösungen der LP-Relaxierungen der Probleme an den Knoten gegeben sind, können aber Teilbäume abgeschnitten werden. Branch-and-Bound-Verfahren bilden die Grundlage für die gebräuchlichsten Verfahren zur Lösung ganzzahliger linearer Optimierungsprobleme. Besonders bedeutsame Erweiterungen von Branch-and-Bound-Verfahren sind Branch-and-Cut-Verfahren und Branch-and-Price-Verfahren. Beide Verfahren versuchen, mit zusätzlichen Strategien Verbesserungen der dualen Schranken zu erreichen. Branch-and-Cut-Verfahren kombinieren dazu Branch-and-Bound-Verfahren mit Schnittebenenverfahren, Branch-and-Price-Verfahren verwenden eine Pricing-Strategie. Oft wird mit einem exakten Verfahren nur versucht, eine näherungsweise optimale ganzzahlige Lösung effizient zu bestimmen. Insbesondere spielt bei den vorgestellten exakten Verfahren die Berechnung von ganzzahligen noch nicht notwendigerweise optimalen Punkten eine Rolle. Solche Punkte sind aber schon Lösungen des ganzzahligen Zulässigkeitsproblems, d.h. dieses Problem lässt sich häufig deutlich schneller lösen als das ganzzahlige Optimierungsproblem. Heuristische Strategien zur näherungsweisen Lösung des linearen ganzzahligen Optimierungsproblems umfassen beispielsweise das Runden von Lösungen von LP-Relaxierungen, lokale Suchverfahren, Tabu-Suche, genetische Algorithmen, Simulated-Annealing, variable Nachbarschaftssuche und Ameisenalgorithmen.

Beispiel

BEISPIEL: *Das Polyeder aus Abbildung 51 links hat beispielsweise nur drei ganzzahlige Punkte, nämlich* $(1,0,0)$, $(0,1,0)$ *und* $(0,0,1)$. *Als Lösung des ganzzahligen Zulässigkeitsproblems sowie zur Lösung ganzzahliger Optimierungsprobleme kommen also nur diese drei Punkte in Frage.*

Die Komplexität aller in diesem Abschnitt betrachteter Berechnungsverfahren hängt von der Anzahl der Ungleichungen des betrachteten linearen Ungleichungssystems $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ ab. Bei linearen Ungleichungssystemen können redundante Ungleichungen auftreten. Lässt man eine solche Ungleichung weg, dann ändert sich die Lösungsmenge

des Systems und damit das von dem System definierte Polyeder nicht. Das sukzessive Entfernen redundanter Restriktionen führt somit zu einer irredundanten Darstellung des betrachteten Polyeders. Eine derartige Vereinfachung beeinflusst daher die Ergebnisse aller in diesem Abschnitt vorgestellter Verfahren nicht. Dementsprechend kann es sinnvoll sein, zuerst redundante Ungleichungen zu entfernen und erst dann ein entsprechendes Verfahren auf dem dann kleineren Ungleichungssystem durchzuführen. Der Reduktion des Rechenaufwandes für das entsprechende Verfahren aufgrund der kleineren Anzahl an Ungleichungen steht allerdings der Aufwand eines Algorithmus zum Auffinden redundanter Ungleichungen gegenüber.
Es gibt keinen einfachen Algorithmus, um alle redundanten Ungleichungen eines Ungleichungssystems zu finden. Genauer gesagt lässt sich zeigen, dass die Identifikation einer redundanten Ungleichung zu einer linearen Optimierungsaufgabe äquivalent ist [88]. Allerdings gibt es etliche sehr effiziente Verfahren, um bestimmte Arten von redundanten Ungleichungen aufzufinden [88], z.B. identische Ungleichungen oder Ungleichungen, welche ein Vielfaches einer anderen Ungleichung sind.
Wenn vom Auffinden aller redundanten Ungleichungen die Rede ist, ist noch zu beachten, dass es im Allgemeinen verschiedene Mengen von Ungleichungen gibt, so dass deren Weglassen zu einer irredundanten Darstellung des ursprünglichen Polyeders führt. Es geht also darum, durch Weglassen von Ungleichungen ein irredundantes Ungleichungssystem, welches das ursprüngliche Polyeder definiert, zu erhalten. Die Menge der wegzulassenden Ungleichungen ist hierbei aber nicht eindeutig, sogar die Mächtigkeit der Menge ist nicht eindeutig. Entsprechend gibt es im Allgemeinen keine eindeutige irredundante Darstellung eines Polyeders durch ein Ungleichungssystem und nicht einmal die Größe einer irredundanten Darstellung muss eindeutig sein.

Beispiel

BEISPIEL: *Ergänzen wir zu dem Ungleichungssystem aus Abbildung 51 links die Ungleichung* $-x_1 \geqslant -1$*, so ist diese Ungleichung redundant, d.h. das entstehende Ungleichungssystem definiert noch immer dasselbe Polyeder. Um eine irredundante Darstellung des Polyeders zu erhalten, muss diese Ungleichung also wieder entfernt werden. Die betrachtete redundante Ungleichung verursacht insbesondere auch das Problem, dass die Ecke* $(1,0,0)$ *in der redundanten Darstellung des Polyeders degeneriert ist, da diese Ecke die neue Ungleichung mit Gleichheit erfüllt.*
Eine andere Möglichkeit das Ungleichungssystem aus Abbildung 51 links um eine redundante Ungleichung zu erweitern, ist die Ungleichung $2x_1 + 2x_2 + 2x_3 \geqslant 2$*, welche gerade das doppelte der ersten dargestellten Ungleichung ist. In dem resultierenden Ungleichungssystem sind dann diese beiden Ungleichungen redundant. Dennoch würde ein Weglassen beider Ungleichungen das Polyeder natürlich verändern. Um eine irredundante Darstellung des Polyeders zu erhalten, ist es aber egal, welche der beiden Ungleichungen weggelassen wird. Dieses Beispiel zeigt vor allem, warum beim Weglassen redundanter Ungleichungen ein sukzessives Vorgehen sinnvoll ist.*

Es existieren neben dem Weglassen redundanter Ungleichungen auch noch etliche weitere Möglichkeiten, um die beschriebenen Verfahren in bestimmten Fällen effizienter zu gestalten [207, 148, 214, 62]. Wir haben dabei Methoden zum Eliminieren nicht benötigter Variablen, zum Lösen dualer Probleme und zur Parallelisierung der Verfahren in Betracht gezogen. Für unsere Zwecke gab es jedoch keine Hinweise

darauf, dass diese Methoden signifikante Verbesserungen erbringen könnten. Dies hat sich auch in einigen experimentellen Tests bestätigt, so dass derartige Methoden im Weiteren keine Rolle spielen.

4

VERFAHREN ZUR SYNTHESE VON STELLEN/TRANSITIONS-NETZEN AUS ENDLICHEN PARTIELLEN SPRACHEN

In diesem Kapitel präsentieren wir Verfahren zur Synthese von Petrinetzen aus halbgeordneten Abläufen. Wir betrachten hierzu die klassische Fragestellung der exakten Synthese eines Stellen/Transitions-Netzes aus einer endlichen partiellen Sprachen.
Innerhalb unseres Synthesebaukastens aus der Einleitung wird also eine konkrete Problemstellung entsprechend der Bausteine der Problemdimension ausgewählt, nämlich die Netzklasse der S/T-Netze, der Sprachtyp der partiellen Sprache, eine endliche Sprache als Sprachklasse und klassische exakte Synthese als Variante der Synthesefragestellung. Für diese Problemstellung werden die einzelnen Bausteine der Lösungsdimension des Synthesebaukastens umfassend diskutiert. Wir stellen für jeden Baustein der Lösungsdimension mehrere geeignete Lösungsverfahren vor und zeigen, wie sich diese zu Lösungsverfahren für das gesamte betrachtete Syntheseproblem kombinieren lassen. Damit ergibt sich ein umfassender, bzgl. des heutigen Forschungsstands vollständiger Überblick von möglichen Lösungsverfahren. Wir diskutieren und vergleichen die Lösungsverfahren sowohl theoretisch als auch mithilfe von Implementierungen und experimentellen Resultaten.
Wir beginnen das Kapitel mit einer Diskussion der betrachteten Problemstellung gefolgt von einer überblicksartigen Erklärung der Vorgehensweise zur Lösung des Problems im Rahmen des vorgestellten Synthesebaukastens. Dann werden die drei Bausteine der Lösungsdimension des Synthesebaukastens, nämlich Regionendefinitionen, Berechnungsverfahren und Übereinstimmungstests, im Detail behandelt. Anschließend werden Implementierungen der resultierenden Syntheseverfahren vorgestellt. Zuletzt werden die Verfahren bzgl. verschiedener Kriterien verglichen.

4.1 PROBLEMSTELLUNG

Ausgangspunkt des hier betrachteten Standard-Syntheseproblems bildet eine Ablauf-Spezifikation in der Form einer endlichen partiellen Sprache. Wir sind an Algorithmen interessiert, welche zuerst entscheiden, ob ein Systemmodell in der Form eines markierten S/T-Netzes existiert, dessen Ablaufsemantik der Ablauf-Spezifikation entspricht. Im positiven Fall soll ein solches Lösungsnetz berechnet werden. Gegeben ist also eine endliche partielle Sprache und gesucht ist ein Lösungsnetz mit einer entsprechenden Ablaufsemantik.
Dabei ist entscheidend, dass bei dem von uns verwendeten Ablaufbegriff eine Ordnungsbeziehung zwischen zwei Knoten einer BPO eine mögliche Abhängigkeit zwischen den entsprechenden Schaltereignis-

sen definiert: Falls zwei Ereignisse einer BPO geordnet sind, so bedeutet dies, dass die Ereignisse in dem betrachteten Ablauf in der gegebenen Reihenfolge schalten. Dabei spielt es keine Rolle, ob die Ereignisse kausal voneinander abhängig sind, d.h. das Schalten des ersten Ereignisses aktiviert das zweite Ereignis, oder ob die Ereignisse nebenläufig zueinander sind. Im Gegensatz zur Prozessablaufsemantik wird also keine kausale Abhängigkeit der Ereignisse gefordert. Ungeordnete Ereignisse müssen dabei in jedem Fall nebenläufig zueinander schalten.
Aus diesen Überlegungen lässt sich folgern, dass die Ablaufsemantik eines S/T-Netzes immer präfix- und sequentialisierungsabgeschlossen ist. Würden wir nun im Rahmen der Synthese eines Petrinetzes fordern, dass die Ablaufsemantik eines Lösungsnetzes mit der Eingabesprache übereinstimmt, so gäbe es nur für präfix- und sequentialisierungsabgeschlossene partielle Sprachen ein Lösungsnetz. Für einen Anwender, welcher einen Beispielablauf spezifiziert, ist es aber unnatürlich, auch noch alle Präfixe und Sequentialisierungen des halbgeordneten Ablaufs anzugeben. Ohnehin impliziert das hier zugrunde gelegte Ablaufverständnis, dass in einem System, in dem ein halbgeordneter Ablauf möglich ist, auch alle Präfixe und Sequentialisierungen möglich sind. Das durch eine partielle Sprache spezifizierte Ablaufverhalten entspricht daher in natürlicher Weise dem Präfix- und Sequentialisierungsabschluss der Sprache. Dementsprechend ist es nicht nur aus Anwendungssicht sinnvoll, nach einem Lösungsnetz zu fragen, dessen Ablaufsemantik dem Präfix- und Sequentialisierungsabschluss der spezifizierten partiellen Sprache entspricht. Typischerweise wird ein Anwender sogar nur die Repräsentationsmenge der gewünschten Ablaufsemantik spezifizieren; dies wollen wir aber nicht zwingend voraussetzen, da eine Berechnung der Repräsentationsmenge u.U. aufwändig sein kann. Entsprechend dieser Überlegungen formulieren wir nun das Syntheseproblem, welches in diesem Kapitel untersucht wird. Wir suchen einen möglichst effizienten korrekten Algorithmus, welcher folgender Problemspezifikation genügt.

Problemspezifikation 4.1.1 (Standard-Syntheseproblem)

PROBLEMSPEZIFIKATION 4.1.1 (STANDARD-SYNTHESEPROBLEM)
Eingabe: *Eine endliche partielle Sprache* $\mathcal{L}$.
Ausgabe: *Ein markiertes S/T-Netz* (N, m_0) *mit* $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$, *falls ein solches Lösungsnetz existiert, und eine „*`notexists`*"-Meldung sonst.*

Algorithmisch macht es nur aus Komplexitätsüberlegungen heraus einen Unterschied, ob wir $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$ oder $\mathfrak{Bpo}(N, m_0) = \mathcal{L}$ fordern, da sich $PS(\mathcal{L})$ aus $\mathcal{L}$ leicht berechnen lässt, aber $PS(\mathcal{L})$ häufig exponentiell größer ist als $\mathcal{L}$. Es wird sich zeigen, dass, obwohl wir hier die Forderung an das zu synthetisierende Netz für $PS(\mathcal{L})$ formuliert haben, es für die meisten Verfahren und Definitionen dieses Kapitels nicht nötig ist, $PS(\mathcal{L})$ explizit zu betrachten und zu berechnen, d.h. die meisten Berechnungen können direkt mit der wesentlich kleineren Menge an BPOs aus $\mathcal{L}$ durchgeführt werden. Im Gegenteil macht es meist keinen Unterschied, ob $\mathcal{L}$, $PS(\mathcal{L})$, $Rep(\mathcal{L})$ oder irgendeine andere Sprache $\mathcal{L}'$ mit $Rep(\mathcal{L}) \subseteq \mathcal{L}' \subseteq PS(\mathcal{L})$ als Eingabe verwendet wird.

Beispiel

BEISPIEL: *In diesem Kapitel werden die Beispiele des vorherigen Kapitels aufgegriffen. Dort wurde diskutiert, inwiefern die aus den zwei BPOs* bpo_1 *und* bpo_2 *bestehende partielle Sprache aus Abbildung 4 zwei Abläufe modelliert. Im Kontext einer Synthesefragestellung spezifizieren die zwei Abläufe das Verhalten eines gesuchten Systems. Allerdings weist ein System, welches diese Abläufe bzgl. des von uns verwendeten Ablaufbegriffs der aktivierten BPOs*

besitzt, in jedem Fall auch alle durch die BPOs $\mathrm{PS}(\{\mathrm{bpo}_1, \mathrm{bpo}_2\})$ *aus Abbildung 47 gegebenen Abläufe auf. Daher ergibt sich in natürlicherweise die Frage nach einem System, welches genau diese Abläufe erlaubt. Gerade diese Frage haben wir in Problemspezifikation 4.1.1 formuliert. Für die diskutierte Eingabe* $\{\mathrm{bpo}_1, \mathrm{bpo}_2\}$ *ist dementsprechend das Netz aus Abbildung 3 eine mögliche Ausgabe. Dieses Netz löst das Syntheseproblem, da dessen Ablaufsemantik, wie in Kapitel 3 dargestellt, der partiellen Sprache* $\mathrm{PS}(\{\mathrm{bpo}_1, \mathrm{bpo}_2\})$ *entspricht.*

Entsprechend der betrachteten Problemspezifikation muss ein Algorithmus zur Lösung des Syntheseproblems insbesondere das Entscheidungsproblem lösen, ob es überhaupt ein markiertes S/T-Netz (N, m_0) gibt, welches $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$ erfüllt. Dies ist nicht immer der Fall. In dem Falle, dass ein Lösungsnetz (N, m_0) mit $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$ existiert, gibt es immer unendlich viele verschiedene solche Netze. Wiederum entsprechend der Problemspezifikation muss ein Synthesealgorithmus in diesem Fall eines dieser Lösungsnetze als Ausgabe liefern. Bei dem in diesem Kapitel betrachteten Syntheseproblem spielt es theoretisch erst einmal keine Rolle, welches Lösungsnetz ausgegeben wird.

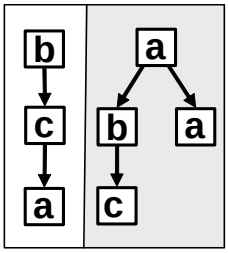

Abbildung 54: Eine partielle Sprache.

Beispiel

Beispiel: *Ein ganz einfaches Beispiel für ein Verhalten, welches nicht durch ein S/T-Netz reproduziert werden kann, sind die zwei Schaltfolgen* abca *und* bca *bzw. deren assoziierte BPOs. Das Schalten von* abc *und* bca *in einem Netz führt in jedem Fall zur selben Markierung (da dieselben Transitionen geschaltet werden), so dass es nicht möglich ist, dass* a *nur nach einem der beiden Schaltvorgänge aktiviert ist, jedoch nicht nach dem anderen. Somit muss ein Lösungsalgorithmus unseres Standard-Syntheseproblems für die partielle Sprache* $\mathcal{L}'$ *aus Abbildung 54, wobei die zweite BPO der BPO* bpo_2 *aus Abbildung 4 entspricht und die erste BPO die BPO* bpo_1 *aus Abbildung 4 um ein* a*-Ereignis fortsetzt, eine „*notexists*“-Meldung ausgeben. Dies liegt daran, dass in einem Lösungsnetz* abca *und* bca*, nicht jedoch* bcaa *aktiviert sein müssten.*
Nun betrachten wir wiederum die partielle Sprache $\{\mathrm{bpo}_1, \mathrm{bpo}_2\}$ *aus Abbildung 4. Wie erläutert, löst das Netz aus Abbildung 3 das Syntheseproblem für diese Sprache positiv. Diese Lösung ist allerdings nicht eindeutig. Beispielsweise ist das Netz aus Abbildung 42 ein etwas größeres Lösungsnetz und Abbildung 55 zeigt schließlich noch ein zum ersten Netz deutlich verschiedenes mögliches Lösungsnetz. Auch ein Netz, welches die Stellen all dieser drei Netze enthält, wäre ein Lösungsnetz. Generell gibt es unendlich viele verschiedene Lösungsnetze, da beispielsweise alle Vielfachen einer Stelle eines Lösungsnetzes zu dem Netz hinzugefügt werden können bzw. die entsprechende Stelle ersetzen können, ohne dass sich das Verhalten des Netzes verändert. Das Bilden eines Vielfachen bedeutet, dass alle Gewichte von mit*

der Stelle verbundenen Kanten und die Anfangsmarkierung der Stelle mit einer positiven Zahl multipliziert werden, z.B. ist die Stelle $2p_4$ *in Abbildung 42 das Doppelte der Stelle* p_4*. Abbildung 56 illustriert alle Vielfachen der Stelle* p_2 *aus Abbildung 3.*

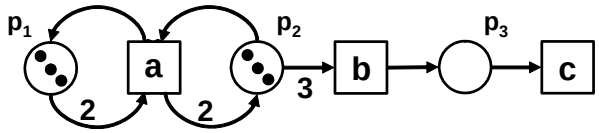

Abbildung 55: Ein markiertes S/T-Netz mit demselben Verhalten wie das Netz aus Abbildung 3.

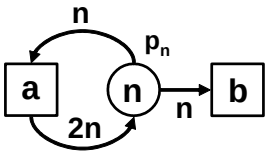

Abbildung 56: Vielfache der Stelle p_2 aus Abbildung 3.

Die hier formulierte Problemspezifikation überträgt die klassischen Fragestellungen aus dem Bereich der Theorie der Petrinetzsynthese (siehe z.B. [18]) auf partielle Sprachen und S/T-Netze. Die formulierte Problemspezifikation lässt sich also als die in unserem Kontext klassische theoretische Synthesefragestellung verstehen.
Allerdings stellt insbesondere die Betrachtung endlichen Verhaltens eine wesentliche Einschränkung dar. Wie wir im letzten Kapitel gesehen haben, kommen in der Realität zwar teilweise Spezifikationen endlichen Verhaltens vor, dennoch ist diese Einschränkung aus Anwendungssicht problematisch. Wir betrachten gemäß Problemspezifikation 4.1.1 exakte Syntheseverfahren, welche kein unendliches Verhalten generieren, wenn ein solches nicht spezifiziert ist. Reale Systeme weisen aber häufig Verhalten auf, welches sich unendlich oft wiederholen lässt. Auf dieses Problem werden wir allerdings erst im nächsten Kapitel in Abschnitt 5.1 eingehen.
Im Hinblick auf Anwendungen ergeben sich darüber hinaus typischerweise auch einige spezielle Anforderungen an Synthesealgorithmen. In Kapitel 2 haben wir schon kurz diskutiert, dass für Anwendungen die Performance eines Syntheseverfahrens und die Größe der synthetisierten Netze eine entscheidende Rolle spielen. Einen möglichst effizienten Synthesealgorithmus zu finden, ist dabei ohnehin auch ein zentraler Aspekt der theoretischen Ausführungen dieses Kapitels. Die Größe der resultierenden Netze spielt in den theoretischen Ausführungen dahingegen eine geringere Rolle. Insbesondere wird in der obig formulierten Problemspezifikation nicht explizit nach einem möglichst kompakten Netz gefragt, sondern es können beliebige Lösungsnetze synthetisiert werden. Dennoch diskutieren wir in diesem Kapitel auch verschiedene Möglichkeiten, um möglichst kompakte Netze zu erzeugen. Auch bei dem Vergleich der verschiedenen Algorithmen dieses Kapitels in Abschnitt 4.7 werden wir die Größe der von einem Synthesealgorithmus synthetisierten Netze als wichtiges Vergleichskriterium in Betracht ziehen.

Eine weitere typische Anforderung aus Anwendungssicht ist schließlich, dass normalerweise ein Interesse an einem sinnvollen Netzmodell des durch Abläufe spezifizierten Systems besteht, auch wenn ein exaktes Lösungsnetz nicht existiert. In der Praxis lässt sich eine Spezifikation häufig nicht exakt durch ein Petrinetz repräsentieren. In diesem Fall ist es oft sinnvoll, dass ein Synthesealgorithmus ein Netz berechnet, welches die Spezifikation möglichst gut widerspiegelt. Dies begründet sich insbesondere auch darin, dass eine reale System-Spezifikation vielfach mehr als möglicherweise noch unvollständiger Design-Vorschlag und nicht notwendigerweise als exakte Implementierungsanforderung zu verstehen ist. Generell ist aus Anwendungssicht die Berechnung eines in jedem Falle sinnvollen Systemmodells aus einer Spezifikation oft wichtiger als zu entscheiden, ob überhaupt ein Lösungsnetz existiert. Ist im Falle der Existenz eines Lösungsnetzes garantiert, dass ein solches berechnet wird, und im anderen Falle sichergestellt, dass eine möglichst gute Annäherung an die Spezifikation berechnet wird, so ist es häufig nicht mehr wichtig zu entscheiden, ob das Syntheseproblem positiv lösbar ist. Es wird zwar in der in diesem Abschnitt betrachteten Problemspezifikation nicht gefordert, dass ein Netz berechnet wird, wenn es kein exaktes Lösungsnetz gibt. Dennoch erzeugen die Synthesealgorithmen dieses Kapitels auch in diesem Fall ein Netz, welches eine gute Approximation an die Spezifikation darstellt. Alle Algorithmen gehen dabei so vor, dass zuerst ein Netz berechnet wird, welches ein Lösungsnetz ist, falls es ein solches gibt. Erst danach wird entschieden, ob es sich tatsächlich um ein Lösungsnetz handelt und damit auch ob es überhaupt ein Lösungsnetz gibt. Wenn wir in der betrachteten Problemspezifikation von der Lösung des Entscheidungsproblems, ob es ein Lösungsnetz gibt, abstrahieren, führt dies also zu einer wesentlichen Vereinfachung der Synthesealgorithmen. Ein ansonsten meist notwendiger Test, ob ein konstruiertes Netz das spezifizierte Verhalten aufweist oder nicht, ist dann überflüssig. Während ein solches Vorgehen in praktischer Hinsicht oft sinnvoll ist, so ist es für die Theorie der Synthese von Petrinetzen allerdings wichtig, auch das Entscheidungsproblems, ob es ein Lösungsnetz gibt, zu behandeln.

Beispiel

BEISPIEL: *Wir haben in dem letzten Beispiel diskutiert, dass es verschiedene Lösungsnetze für eine partielle Sprache gibt. Die Transitionen der Lösungsnetze sind identisch, nicht aber die Stellen. Im Hinblick auf den Anwendungsaspekt der Kompaktheit eines Netzes sollte daher möglichst ein solches Netz synthetisiert werden, welches zumindest einmal keine impliziten Stellen enthält. Das Netz aus Abbildung 42 wäre also unter diesem Gesichtspunkt kein geeignetes Lösungsnetz. Allerdings beinhaltet weder das Netz aus Abbildung 3 noch dasjenige aus Abbildung 55 implizite Stellen. Für diese Netze ist es nicht eindeutig zu beantworten, welches Netz kompakter ist. Während das eine weniger Stellen besitzt, hat das andere geringere Kantengewichte und eine kleinere Anfangsmarkierung. In einem solchen Fall können je nach Anwendungskontext geeignete Maßzahlen für die Größe eines Netzes zu Rate gezogen werden (siehe z.B. [194]).*
In dem letzten Beispiel wurde auch die partielle Sprache $\mathcal{L}'$ aus Abbildung 54 diskutiert, zu der es kein Lösungsnetz gibt. Aus Anwendungssicht ist hier die Berechnung einer Näherungslösung sinnvoll. Beispielsweise wäre das Netz aus Abbildung 3 schon eine gute Näherung, da dessen Ablaufverhalten sich nur geringfügig von $\mathrm{PS}(\mathcal{L}')$ unterscheidet. Allerdings ist die erste BPO aus Abbildung 54 bzgl. dieses Netzes nicht aktiviert. Manchmal ist es eher sinnvoll, eine Näherungslösung zu suchen, deren Ablaufverhalten $\mathrm{PS}(\mathcal{L}')$

enthält und möglichst wenig zusätzliche Abläufe erlaubt. Ein solches Netz ist in Abbildung 57 dargestellt.

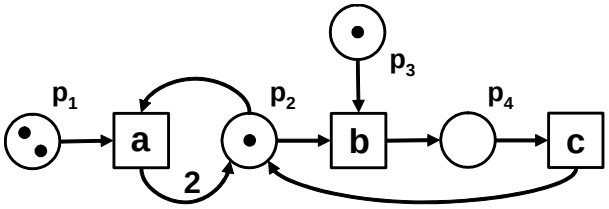

Abbildung 57: Näherungs-Netz für $\mathcal{L}'$ aus Abbildung 54.

Die in den letzten Absätzen diskutierten Anwendungsaspekte für Synthesealgorithmen werden in den folgenden theoretischen Überlegungen des Kapitels nur eine untergeordnete Rolle spielen (abgesehen von der auch aus theoretischer Sicht bedeutsamen Effizienz der Verfahren). Wir werden uns zumeist auf die in diesem Abschnitt formulierte theoretische Problemspezifikation konzentrieren. Allerdings werden wir im nächsten Kapitel in Abschnitt 5.4 auf entsprechende Modifikationen der Problemspezifikation eingehen, welche derartige Anwendungsaspekte berücksichtigen.

4.2 LÖSUNGSANSATZ

Zur Entwicklung von Lösungsverfahren für das im letzten Abschnitt formulierte Syntheseproblem greifen wir auf die Konzepte der sog. Regionentheorie zurück (siehe z.B. [18]). Der Begriff der Regionentheorie wurde für eine Vorgehensweise, welche in ähnlicher Form allen bekannten Verfahren zur Synthese von Petrinetzen aus Verhaltens-Spezifikationen zugrunde liegt, geprägt. Regionen wurden ursprünglich in den grundlegenden Arbeiten [89, 90] eingeführt. Sie stellten die zentrale Definition zur Lösung des Problems der Synthese eines elementaren Netzes aus einem Transitionssystem dar. Die Idee hinter dieser Definition wurde in den weiteren Arbeiten zur Synthese von Petrinetzen verschiedenster Klassen aus unterschiedlichen Verhaltens-Spezifikationen wiederverwendet. Diese schon in der Einleitung kurz skizzierte Idee lässt sich informal folgendermaßen beschreiben:

- Es wird nicht zuerst das Entscheidungsproblem, ob es ein Lösungsnetz für das Syntheseproblem gibt, gelöst und dann im positiven Fall ein Lösungsnetz berechnet, sondern es wird direkt ein Kandidat für ein Lösungsnetz konstruiert.
- Die Konstruktion beginnt mit den Transitionen des Netzes. Diese lassen sich direkt aus den Schaltereignissen der Verhaltensspezifikation ablesen.
- Um das Schaltverhalten der Transitionen einzuschränken, müssen Stellen zusammen mit deren (gewichteten) Verbindungskanten zu Transitionen und deren Anfangsmarkierungen hinzugefügt werden. Eine Stelle legt eine Abhängigkeit zwischen den Transitionen im Vorbereich und denen im Nachbereich der Stelle fest.

- Die Verhaltenseinschränkung einer Stelle kann das spezifizierte Verhalten zulassen oder verhindern. Im ersteren Fall heißt eine Stelle zulässig.
- Da das zu synthetisierende Netz das spezifizierte Verhalten ermöglichen soll, werden nur zulässige Stellen zu dem Netz hinzugefügt. Die eigentliche Idee ist es alle zulässigen Stellen hinzuzufügen. Auf diese Weise wird das Verhalten des Netzes maximal eingeschränkt unter der Bedingung, dass das spezifizierte Verhalten noch beinhaltet ist. Das resultierende Netz heißt gesättigt zulässiges Netz.
- Das gesättigt zulässige Netz ermöglicht das spezifizierte Verhalten und hat minimales zusätzliches Verhalten. Folglich stimmt das Verhalten des Netzes entweder mit der gegebenen Verhaltensspezifikation überein oder es gibt kein Netz mit dem spezifizierten Verhalten.
- Ein Problem hierbei ist, dass es normalerweise unendlich viele zulässige Stellen gibt. Zumindest ist die Anzahl zulässiger Stellen sehr hoch. Es reichen aber meist endliche, wesentlich kleinere Repräsentationsmengen an zulässigen Stellen aus, um dieselbe Verhaltenseinschränkung wie mit der Menge aller zulässiger Stellen zu erzielen oder zumindest alle relevanten Abhängigkeiten zu berücksichtigen.
- Die entscheidende Fragestellung ist nun, wie sich derartige Repräsentationsstellenmengen des gesättigt zulässigen Netzes identifizieren lassen. Hierzu wird die Eigenschaft der Zulässigkeit in entsprechende Anforderungen, welche auf der Struktur der gegebenen Verhaltensspezifikation basieren, übersetzt. Dies führt zum mathematischen Objekt der Region. Jede Region definiert eine zulässige Stelle und jede zulässige Stelle ist durch eine Region definiert.
- Mit geeigneten Regionendefinitionen lassen sich in vielen Fällen Repräsentationsstellenmengen des gesättigt zulässigen Netzes algorithmisch konstruieren. Durch Hinzufügen der Stellen einer solchen Repräsentationsmenge zur betrachteten Transitionsmenge entsteht dann das zu synthetisierende Kandidatennetz.
- Das Kandidatennetz hat wie das gesättigt zulässige Netz die Eigenschaft, dass es entweder das Syntheseproblem in einem positiven Sinne löst oder das Syntheseproblem eine negative Antwort hat. Eine negative Antwort ergibt sich genau dann, wenn in dem synthetisierten Netz zusätzliches Verhalten gegenüber der Spezifikation möglich ist. Um dies zu prüfen, wird in einem sog. Übereinstimmungstest das Verhalten des Netzes mit der Spezifikation verglichen.

Entsprechend dieser allgemeinen Überlegungen lässt sich die Entwicklung eines Verfahrens zur Synthese eines S/T-Netzes aus einer partiellen Sprache in die drei Bausteine der Lösungsdimension des betrachteten Synthesebaukastens aufgliedern. Zuerst muss ein geeigneter Regionenbegriff gefunden werden, mit dem sich die zulässigen Stellen einer partiellen Sprache charakterisieren lassen. Mithilfe einer Regionendefinition muss dann ein Verfahren zur Berechnung einer Repräsentationsstel-

lenmenge des gesättigt zulässigen Netzes einer partiellen Sprache entwickelt werden. Schließlich ist noch ein Übereinstimmungstest-Verfahren, welches die Ablaufsemantik eines synthetisierten Kandidatennetzes auf Übereinstimmung mit der spezifizierten partiellen Sprache prüft, notwendig. Algorithmus 4.2.1 stellt das entsprechende Synthesevorgehen auf einer abstrakten Ebene dar.

Algorithmus 4.2.1 (Grundlegendes Syntheseverfahren)

ALGORITHMUS 4.2.1 (GRUNDLEGENDES SYNTHESEVERFAHREN)

```
1   Eingabe: Partielle Sprache ℒ
2   Ausgabe: Markiertes S/T–Netz (N, m0) mit
      𝔅𝔭𝔬(N, m0) = PS(ℒ), falls ein solches (N, m0)
      existiert, und eine „notexists“–Meldung sonst
3   BEGIN
4       regionenTypFestlegen();
5       (N, m0):=berechnungRepraesentationsNetz(ℒ);
6       IF uebereinstimmungstest((N, m0), ℒ) THEN
7           RETURN (N, m0);
8       ELSE
9           RETURN „notexists“;
10      ENDIF
11  END
```

In den drei folgenden Abschnitten werden wir die drei Bausteine der Lösungsdimension des Synthesebaukastens diskutieren. Wir beginnen mit der Einführung verschiedener Regionendefinitionen für partielle Sprachen, stellen anschließend einige darauf basierende Verfahren zur Berechnung geeigneter Netze vor und gehen schließlich noch auf mögliche Verfahren für einen Übereinstimmungstest ein.

4.3 REGIONENDEFINITIONEN

Wie im letzten Abschnitt erklärt, bildet das Konzept der zulässigen Stellen die Grundlage für Definitionen von Regionen. Dementsprechend wird in diesem Abschnitt zuerst der Begriff der zulässigen Stelle und des gesättigt zulässigen Netzes für die in diesem Kapitel betrachtete Problemstellung der Synthese eines S/T-Netzes aus einer partiellen Sprache motiviert und formal definiert. In den anschließend folgenden Unterabschnitten werden dann entsprechende Regionendefinitionen eingeführt. Die meisten Definitionen und Ergebnisse dieses Abschnittes sind für unendliche partielle Sprachen formuliert. Die Beschränkung auf endliche partielle Sprachen in Problemspezifikation 4.1.1 wird in diesem Abschnitt nur für linear algebraische Charakterisierungen der Regionenbegriffe verwendet. Allerdings betrachten wir insgesamt ausschließlich partielle Sprachen mit einer endlichen Beschriftungsmenge. Das Ziel ist nun, zunächst die Idee der zulässigen Stellen zu formalisieren. Die endliche Menge der Transitionen eines zu synthetisierenden S/T-Netzes (N, m_0) ist unmittelbar durch die Beschriftungsmenge T der gegebenen partiellen Sprache $\mathcal{L}$ bestimmt. Diese Transitionen können also schon einmal konstruiert werden. In dem entstehenden Netz, welches noch keine Stellen beinhaltet, ist dann aber jede BPO über der gegebenen Transitionsmenge aktiviert. Der entscheidende Syntheseschritt ist das Hinzufügen geeigneter Stellen zu dieser Transitionsmenge, so dass nur noch die BPOs aus $\mathcal{L}$ aktiviert sind.

BEISPIEL: *Abbildung 58 zeigt ein Netz, welches ausschließlich aus den durch die partielle Sprache aus Abbildung 4 gegebenen Transitionen besteht. Jede BPO, welche nur* a*-,* b*- und* c*-Ereignisse enthält, ist bzgl. dieses Netzes aktiviert. Somit sind auch viele BPOs aktiviert, welche nicht durch die betrachtete partielle Sprache spezifiziert sind. Die Ablaufsemantik eines Netzes wird durch das Hinzufügen von Stellen eingeschränkt. Jede der vier Stellen aus Abbildung 3 verhindert die Ausführbarkeit einer bestimmten jeweils verschiedenen Menge an BPOs. Beispielsweise verhindert die Stelle* p_1 *die Ausführbarkeit aller BPOs mit drei oder mehr* a*-Ereignissen. Nach dem Hinzufügen aller vier Stellen zur ursprünglichen Transitionsmenge sind in dem dann resultierenden Netz nur noch die gewünschten BPOs aktiviert.* *Beispiel*

Abbildung 58: Netz ohne Stellen.

Wir haben bei den eben durchgeführten Überlegungen lapidar vom Hinzufügen von Stellen zu dem zu synthetisierenden Netz gesprochen, meinen aber eigentlich Stellen zusammen mit, im Falle von S/T-Netzen, deren gewichteten Verbindungskanten zu Transitionen und deren Anfangsmarkierungen. Die Benennung der Stellen spielt dabei keine Rolle. Daher betrachten wir formal sog. Stellentripel über T, welche aus einer Komponente für die Anfangsmarkierung der hinzuzufügenden Stelle sowie je einer Komponente für den Multivorbereich und den Multinachbereich der Stelle bestehen. Das Festlegen des Multivorbereichs und des Multinachbereichs einer Stelle entspricht dem Festlegen der gewichteten Verbindungskanten der Stelle von und zu jeder Transition. Das Hinzufügen eines Stellentripels zur gegebenen Transitionsmenge bzw. allgemeiner zu einem Netz mit der gegebenen Transitionsmenge bedeutet also, dass eine Stelle mit einer beliebigen neuen Benennung hinzugefügt wird, wobei die Anfangsmarkierung bzgl. dieser Stelle gemäß der entsprechenden Komponente des Stellentripels festgelegt wird und die Gewichtsfunktion bzgl. dieser Stelle gemäß den Komponenten des Stellentripels für den Multivorbereich und den Multinachbereich festgelegt wird. Ein Stellentripel über T definiert formal also ein Netz über der Transitionsmenge T mit nur einer Stelle, ein sog. atomares Netz. Hinzufügen mehrerer Stellentripel zur gegebenen Transitionsmenge entspricht dem Verschmelzen der übereinstimmenden Transitionen der durch die Stellentripel definierten atomaren Netze. Dies bewirkt formal, dass die durch die hinzugefügten Stellentripel definierten atomaren Netze genau die atomaren Teilnetze des synthetisierten S/T-Netzes (N, m_0) sind.
Wichtig ist hier, noch einmal zu bemerken, dass wir in diesem Kontext häufig eine intuitive informale Ausdrucksweise verwenden. So sprechen wir der Einfachheit halber häufig von einer Stelle, obwohl wir formal ein Stellentripel bzw. ein entsprechendes atomares Teilnetz bezeichnen. In den formalen Definitionen wird allerdings immer die formal korrekte Bezeichnung verwendet.

Definition 4.3.1 (Stellentripel)

Definition 4.3.1 (Stellentripel)
Ein Stellentripel über einer endlichen Menge von Transitionen T *ist ein Tripel* $st = (st_0, {}^\circ st, st^\circ)$, *wobei* $st_0 \in \mathbb{N}$ *und* ${}^\circ st, st^\circ \in \mathbb{N}^T$.
Ein markiertes S/T-Netz (N, m_0), $N = (P, T, W)$, *heißt atomar, falls* $|P| = 1$.
Ein Stellentripel st *über* T *definiert ein atomares Netz* (N_{st}, m_{st}), $N_{st} = (\{p_{st}\}, T, W_{st})$, *wobei* $m_{st}(p_{st}) = st_0$, $W_{st}(t, p_{st}) = {}^\circ st(t)$ *für alle* $t \in T$ *und* $W_{st}(p_{st}, t) = st^\circ(t)$ *für alle* $t \in T$.
Umgekehrt ist zu jedem atomaren Netz (N, m_0), $N = (\{p\}, T, W)$, *ein Stellentripel* $st = (st_0, {}^\circ st, st^\circ)$ *durch* $st_0 = m_0(p)$, ${}^\circ st = {}^\circ p$ *und* $st^\circ = p^\circ$ *assoziiert.*

Das zu einem atomaren Netz assoziierte Stellentripel definiert ein atomares Netz, welches sich von dem ersten atomaren Netz nur durch die Benennung der Stelle unterscheidet. Alle atomaren Netze sind also bis auf Umbenennung durch Stellentripel gegeben.
In obiger Definition wurde die Benennung der Stelle eines von einem Stellentripel definierten atomaren Netzes der Einfachheit halber fest durch ein p mit Indizierung des Tripels gewählt. Wir gehen davon aus, dass dies im Weiteren die Standardbenennung der entsprechenden Stelle beim Hinzufügen eines Stellentripels zu einem Netz ist. In den Beispielen werden wir die Stellen aber meist wie bisher durchnummeriert benennen.
Es sei angemerkt, dass durch die Benennungskonventionen des letzten Absatzes im Weiteren ein synthetisiertes Netz formal nie mehrere verschiedene atomare Teilnetze mit demselben assoziierten Stellentripel enthält. Wie das folgende Lemma zeigt, spielt dies für die in diesem Kapitel betrachtete Ablaufsemantik eines Netzes auch keine Rolle, wohl aber für die Prozessablaufsemantik. Algorithmisch erfordert die Vermeidung solcher atomarer Teilnetze in einigen später auftretenden Fällen eigentlich einen Test auf Übereinstimmung. Auf diese Implementierungsproblematik wollen wir aber nicht weiter eingehen, da solche Teilnetze ohnehin das Ablaufverhalten nicht ändern.

Beispiel

Beispiel: *Jede der vier Stellen des Netzes aus Abbildung 3 definiert eines der vier atomaren Teilnetze des Netzes. Beispielsweise zeigt Abbildung 59 links das zu* p_2 *gehörige Teilnetz, allerdings ohne die Stellenbenennung. Zu diesem atomaren Netz gehört das Stellentripel* $(1, 2a, a + b)$. *Das Hinzufügen dieses Stellentripels zur Transitionsmenge* $\{a, b, c\}$ *führt also zu einer entsprechenden Stelle, deren Standardbenennung dann* $p_{(1,2a,a+b)}$ *wäre. Abbildung 60 illustriert allgemein, inwiefern eine Stelle zusammen mit ihren gewichteten Verbindungskanten zu Transitionen und ihrer Anfangsmarkierung durch ein Stellentripel definiert wird.*

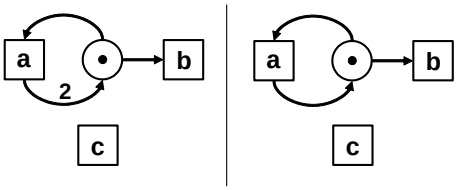

Abbildung 59: Links: Zulässige Stelle. Rechts: Nicht-zulässige Stelle.

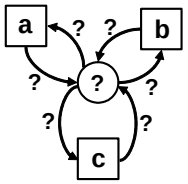

Abbildung 60: Hinzufügen einer Stelle.

Die bisherigen Überlegungen geben noch keine Antwort auf die Frage, welche Stellentripel zu einem zu synthetisierenden Netz hinzugefügt werden sollen. Die entscheidende Eigenschaft von S/T-Netzen (sowie

auch anderen Petrinetzklassen), welche im Rahmen dieser Fragestellung bei der regionenbasierten Synthese aus Sprachen ausgenutzt wird, ist nun die folgende:

LEMMA 4.3.1 *Lemma 4.3.1*
Die Ablaufsemantik eines markierten S/T-Netzes entspricht dem Schnitt der Ablaufsemantiken seiner atomaren Teilnetze.

BEWEIS: Die Aktiviertheit einer BPO ist über die Aktiviertheit von Schrittfolgen definiert. Ein Transitionsschritt τ ist in einem markierten S/T-Netz genau dann aktiviert, wenn er in allen atomaren Teilnetzen aktiviert ist, da jede Komponente der Aktiviertheitsbedingung $m \geqslant {}^\circ\tau$ gerade einem atomaren Teilnetz entspricht. Damit folgt die Behauptung. □

Aus diesem Lemma lässt sich folgern, dass die gewünschte Eigenschaft $\mathcal{L} \subseteq \mathfrak{Bpo}(N, m_0)$ bzw. äquivalent $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$ für ein synthetisiertes Netz (N, m_0) dann und nur dann Bestand hat, wenn nur sog. zulässige Stellentripel über T zu der Ausgangs-Transitionsmenge hinzugefügt werden. Dabei bezeichnen wir ein Stellentripel genau dann als zulässig, wenn die Ablaufsemantik des durch das Stellentripel definierten atomaren Netzes die partielle Sprache $\mathcal{L}$ enthält. Informal gesprochen verhindern zulässige Stellentripel das spezifizierte Verhalten nicht. Daher schränkt sukzessives Hinzufügen solcher Stellentripel das Verhalten des resultierenden Netzes nicht zu stark ein. Wird jedoch ein nicht-zulässiges Stellentripel hinzugefügt, so ist $\mathcal{L} \subseteq \mathfrak{Bpo}(N, m_0)$ und damit auch $PS(\mathcal{L}) = \mathfrak{Bpo}(N, m_0)$ nicht möglich.

DEFINITION 4.3.2 (ZULÄSSIGES STELLENTRIPEL) *Definition 4.3.2 (Zulässiges Stellentripel)*
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Ein Stellentripel st über T heißt zulässig bzgl. $\mathcal{L}$, falls $\mathcal{L} \subseteq \mathfrak{Bpo}(N_{st}, m_{st})$.

BEISPIEL: *Abbildung 59 zeigt das atomare Netz des bzgl. der partiellen Sprache aus Abbildung 4 zulässigen Stellentripels $(1, 2a, a + b)$ und des nicht-zulässigen Stellentripels $(1, a, a + b)$. Die Stelle auf der linken Seite ist zulässig, da sie nach dem Schalten von a mit zwei Marken belegt ist. In dieser Markierung kann der Schritt $a + b$, wie in bpo_2 spezifiziert, schalten. Das Schaltverhalten von c wird durch diese Stelle nicht beeinflusst. Die Stelle auf der rechten Seite ist nicht zulässig, da sie nach einem Schalten von a wiederum nur mit einer Marke belegt ist. In dieser Markierung kann der Schritt $a + b$ nicht schalten. Die BPO bpo_2 ist somit bzgl. des abgebildeten atomaren Netzes nicht aktiviert.*
Insgesamt ist es im Rahmen der Synthese eines Netzes somit sinnvoll, die linke Stelle zu dem Netz hinzuzufügen, nicht jedoch die rechte. Das Hinzufügen der vier zulässigen Stellentripel $(2, 0, a)$ $(1, 2a, a + b)$, $(1, 0, b)$ und $(0, b, c)$ zur Transitionsmenge $\{a, b, c\}$ würde bis auf die Stellenbenennungen zu dem Netz aus Abbildung 3 führen, welches das Syntheseproblem löst. *Beispiel*

Aufgrund der Präfix- und Sequentialisierungsabgeschlossenheit der Ablaufsemantik gilt das folgende Lemma, welches sicherstellt, dass es bzgl. der Zulässigkeit keinen Unterschied macht, ob $\mathcal{L}$ oder $PS(\mathcal{L})$ betrachtet wird.

LEMMA 4.3.2 *Lemma 4.3.2*
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Ein Stellentripel st über T ist genau dann zulässig bzgl. $\mathcal{L}$, wenn es zulässig bzgl. $PS(\mathcal{L})$ ist.

BEWEIS: Die '"wenn"-Implikation folgt aus $\mathcal{L} \subseteq PS(\mathcal{L})$. Die '"genau"-Implikation folgt aus $PS(\mathfrak{Bpo}(N_{st}, m_{st})) = \mathfrak{Bpo}(N_{st}, m_{st})$. □

Wir folgern weiter, dass es, wie in Abschnitt 4.1 diskutiert, im Rahmen der Zulässigkeit sogar genügt, $Rep(\mathcal{L})$ zu betrachten.

Korollar 4.3.1

KOROLLAR 4.3.1
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Ein Stellentripel st über T ist genau dann zulässig bzgl. $\mathcal{L}$, wenn es zulässig bzgl. $Rep(\mathcal{L})$ ist.

BEWEIS: Die Behauptung folgt aus Lemma 4.3.2, da $PS(\mathcal{L}) = PS(Rep(\mathcal{L}))$. □

Grundsätzlich wird die Ablaufsemantik eines Netzes durch das Hinzufügen von Stellentripeln reduziert. Daher lässt sich folgern, dass das Hinzufügen aller zulässiger Stellentripel zur durch $\mathcal{L}$ gegebenen Transitionsmenge zu einem Netz führt, dessen Ablaufsemantik $\mathcal{L}$ und damit auch $PS(\mathcal{L})$ enthält und minimal mit dieser Eigenschaft ist. Das resultierende Netz heißt gesättigt zulässiges Netz. Im Allgemeinen stimmt die Ablaufsemantik des gesättigt zulässigen Netzes allerdings nicht notwendigerweise mit $PS(\mathcal{L})$ überein. In diesem Fall gibt es aber dann auch kein anderes markiertes S/T-Netz, dessen Ablaufsemantik mit $PS(\mathcal{L})$ übereinstimmt. Also hat das Syntheseproblem aus Problemspezifikation 4.1.1 dann und nur dann eine positive Lösung, wenn die Ablaufsemantik des gesättigt zulässigen Netzes der partiellen Sprache $PS(\mathcal{L})$ entspricht.

Definition 4.3.3 (Gesättigt zulässiges Netz)

DEFINITION 4.3.3 (GESÄTTIGT ZULÄSSIGES NETZ)
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Das markierte S/T-Netz (N, m_0), dessen Menge von atomaren Teilnetzen der Menge aller von zulässigen Stellentripeln über T definierten atomaren Netzen entspricht, heißt gesättigt zulässiges Netz bzgl. $\mathcal{L}$.

Lemma 4.3.3

LEMMA 4.3.3
Sei (N, m_0) das gesättigt zulässige Netz bzgl. $\mathcal{L}$. Dann gilt $\mathcal{L} \subseteq \mathfrak{Bpo}(N, m_0)$ und $\mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m_0')$ für alle markierten S/T-Netze (N', m_0') mit $\mathcal{L} \subseteq \mathfrak{Bpo}(N', m_0')$.

BEWEIS: Aus Lemma 4.3.1 folgt direkt $\mathcal{L} \subseteq \mathfrak{Bpo}(N, m_0)$.
Sei (N', m_0') ein markierten S/T-Netz über T mit $\mathcal{L} \subseteq \mathfrak{Bpo}(N', m_0')$. Ebenfalls aus Lemma 4.3.1 folgt, dass für jedes atomare Teilnetz von (N', m_0') das assoziierte Stellentripel zulässig ist. Es ist allerdings möglich, dass es verschiedene atomare Teilnetze mit demselben assoziierten Stellentripel gibt. Aus der Definition der Aktiviertheit von BPOs lässt sich leicht ersehen, dass für solche Teilnetze alle bis auf eines weggelassen werden können, ohne die Ablaufsemantik zu verändern (man beachte, dass dies im Allgemeinen nicht für die Prozessablaufsemantik gilt). Weglassen muss hier formal als Gegenstück zu dem oben diskutierten Begriff des Hinzufügens verstanden werden, d.h. die Menge der atomaren Teilnetze wird genau um das betreffende atomare Netz reduziert. Auch das Umbenennen von Stellen verändert die Ablaufsemantik nicht. Insgesamt lässt sich also durch geeignetes Umbenennen von Stellen und Weglassen von atomaren Teilnetzen das Netz (N', m_0') in ein markiertes S/T-Netz mit derselben Ablaufsemantik überführen, so dass alle atomaren Teilnetze von zulässigen Stellentripeln definiert werden. Somit ist dann die Menge der atomaren Teilnetze eine Teilmenge der

atomaren Teilnetze von (N, m_0). Wiederum mit Lemma 4.3.1 lässt sich nun $\mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m_0')$ folgern.
Entspricht die Menge der Transitionen von (N', m_0') nicht T, so muss sie T echt enthalten. Werden die zusätzlichen Transitionen zusammen mit ihren gewichteten Verbindungskanten zu Stellen weggelassen (ähnlich wie wir Stellen weggelassen haben), so werden aus $\mathfrak{Bpo}(N', m_0')$ alle Abläufe, welche solche Transitionen beinhalten wegfallen. Es entsteht also ein Netz mit einer kleineren Ablaufsemantik, welche aber immer noch $\mathcal{L}$ enthält. Nun können wir mit diesem Netz wie im vorigen Abschnitt argumentieren und erhalten die Behauptung. □

Aus diesem Lemma lässt sich nun formal ableiten, dass entweder das gesättigt zulässige Netz das betrachtete Syntheseproblem positiv löst oder das Problem eine negative Antwort hat.

KOROLLAR 4.3.2 *Korollar 4.3.2*
Sei (N, m_0) das gesättigt zulässige Netz bzgl. $\mathcal{L}$. Falls $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$, so gilt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N', m_0')$ für alle markierten S/T-Netze (N', m_0').

BEWEIS: Sei $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$. Angenommen es gibt ein markiertes S/T-Netz (N', m_0') mit $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$. Dann gilt insbesondere $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N', m_0')$ und damit auch $\mathcal{L} \subseteq \mathfrak{Bpo}(N', m_0')$. Aus Lemma 4.3.3 folgt dann $\mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m_0') = PS(\mathcal{L})$. Ebenfalls nach Lemma 4.3.3 gilt ohnehin $\mathcal{L} \subseteq \mathfrak{Bpo}(N, m_0)$ und damit aufgrund der Präfix- und Sequentialisierungsabgeschlossenheit von $\mathfrak{Bpo}(N, m_0)$ auch $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$. Insgesamt ergibt sich $PS(\mathcal{L}) = \mathfrak{Bpo}(N, m_0)$ und somit ein Widerspruch. □

Das gesättigt zulässige Netz ist also ein geeignetes Kandidatennetz für Syntheseverfahren. Ein Problem ist allerdings, dass es immer unendlich ist, d.h. die Anzahl der zulässigen Stellentripel einer partiellen Sprache ist immer unendlich, da beispielsweise jedes Vielfache eines zulässigen Stellentripels ebenfalls zulässig ist. Damit eignet sich das gesättigt zulässige Netz nicht direkt für algorithmische Berechnungen. Stattdessen sind wir an geeigneten endlichen Repräsentationsmengen der Menge aller zulässigen Stellentripel einer partiellen Sprache interessiert.

BEISPIEL: *Wir haben gezeigt, dass es im Rahmen der Synthese eines Netzes aus der partiellen Sprache aus Abbildung 4 sinnvoll ist, nur bzgl. dieser Sprache zulässige Stellen in Betracht zu ziehen. Weiter haben wir gezeigt, dass wir durch das Hinzufügen aller zulässiger Stellen zur gegebenen Transitionsmenge ein Netz erhalten, welches das Syntheseproblem löst, falls es überhaupt eine Lösung gibt. Ein zentrales Problem bei diesem Vorgehen ist, dass es unendlich viele zulässige Stellen gibt. So sind beispielsweise alle in Abbildung 56 illustrierten Stellen p_n zulässig. Somit ergibt sich neben der Frage, wie zulässige Stellen berechnet werden können, auch die Frage, wie eine geeignete endliche Auswahl zulässiger Stellen, in unserem Fall beispielsweise die vier Stellen aus Abbildung 3, gefunden werden kann.* *Beispiel*

Um algorithmisch zuerst einmal überhaupt zulässige Stellentripel identifizieren zu können und dann auch noch geeignete Repräsentationsmengen berechnen zu können, werden wir im Folgenden verschiedene Regionendefinitionen diskutieren. Regionen erlauben es, die Menge der zulässigen Stellentripel auf der Ebene der partiellen Sprache zu definieren. Wir führen also in den folgenden Unterabschnitten jeweils eine Regionendefinition für partielle Sprachen ein und zeigen, dass sich aus dieser die Menge der zulässigen Stellentripel ableiten lässt.

4.3.1 *Schritt-Transitions-Regionen*

In diesem Unterabschnitt stellen wir die naheliegendste Möglichkeit zur Definition von Regionen für partielle Sprachen vor. Die Idee hierbei ist, dass eine partielle Sprache eine Menge von Schrittfolgen erzeugt. Auf diese Menge von Schrittfolgen lassen sich dann klassische Regionenkonzepte übertragen.
Die Basis für die Ausführungen dieses Unterabschnittes stellt die ursprüngliche Aktiviertheitsdefinition Definition 3.3.8 dar. In dieser Definition wird die Aktiviertheit einer BPO mithilfe der Menge der zu Schrittlinearisierungen der BPO assoziierten Schrittfolgen eingeführt. Daher lässt sich der Begriff des zulässigen Stellentripels einer partiellen Sprache auf einen entsprechenden Begriff für Mengen von Schrittfolgen zurückführen.

Definition 4.3.4 (Zulässiges Stellentripel für Schrittfolgen)

Definition 4.3.4 (Zulässiges Stellentripel für Schrittfolgen)
Sei $\mathcal{S}$ eine Mengen von Schrittfolgen, wobei die Menge T der Transitionen, welche in $\mathcal{S}$ vorkommen, endlich ist. Ein Stellentripel st über T heißt zulässig bzgl. $\mathcal{S}$, falls $\mathcal{S} \subseteq \mathfrak{Step}(N_{st}, m_{st})$.

Lemma 4.3.4

Lemma 4.3.4
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Ein Stellentripel über T ist genau dann zulässig bzgl. $\mathcal{L}$, wenn es zulässig bzgl. $\{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$ ist.

Beweis: Sei st ein Stellentripel über T. Nach Definition 3.3.8 ist eine BPO bpo genau dann aktiviert bzgl. (N_{st}, m_{st}), wenn alle Schrittfolgen der Menge $\{\sigma(bpo') \mid bpo' \in Slin(bpo)\}$ bzgl. (N_{st}, m_{st}) aktiviert sind. Damit sind alle BPOs der partiellen Sprache $\mathcal{L}$ genau dann bzgl. (N_{st}, m_{st}) aktiviert, wenn alle Schrittfolgen der Menge $\bigcup_{bpo \in \mathcal{L}} \{\sigma(bpo') \mid bpo' \in Slin(bpo)\} = \{\sigma(bpo') \mid bpo' \in Slin(\mathcal{L})\}$ bzgl. (N_{st}, m_{st}) aktiviert sind. Es folgt die Behauptung. □

Das vorausgehende Lemma zeigt, dass zur Identifikation zulässiger Stellen einer partiellen Sprache $\mathcal{L}$ auch die Menge von Schrittfolgen $\{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$ betrachtet werden kann. Damit können wir eine Regionendefinition für partielle Sprachen direkt aus einer geeigneten Regionendefinition für Mengen von Schrittfolgen gewinnen. Wir entwickeln also im Weiteren zuerst einen Regionenbegriff für Mengen von Schrittfolgen und lehnen daran eine Regionendefinition für partielle Sprachen an.

Beispiel

Beispiel: *Zur BPO bpo_1 aus Abbildung 4 gehört nur die Schrittfolge bc. Zu bpo_2 gehören, wie schon diskutiert, die Schrittfolgen $aabc$, $abac$, $abca$, $aba + c$ und $aa + bc$. Die insgesamt sechs Schrittfolgen sind in einem Netz genau dann aktiviert, wenn die zwei BPOs aus Abbildung 4 aktiviert sind. Somit lässt sich die Zulässigkeit eines Stellentripels bzgl. der partiellen Sprache aus Abbildung 4 auf die Zulässigkeit bzgl. der Menge der betrachteten Schrittfolgen zurückführen.*

Zur Definition von Regionen auf Mengen von Schrittfolgen greifen wir auf wohlbekannte Regionenkonzepte zurück. Wir kombinieren die wesentlichen Ideen und Prinzipien von drei Ansätzen. In [120] wurden Regionen für Trace-Sprachen vorgeschlagen, ohne effektive Syntheseverfahren abzuleiten. In [13, 18, 65, 63] wurden Regionen für Sprachen von Schaltfolgen vorgestellt. In [173, 17, 18] wurden Regionen

für Schritt-Transitionssysteme eingeführt. Aus den beiden letzteren Regionendefinitionen wurden effektive polynomielle Syntheseverfahren entwickelt. Die drei genannten Regionendefinitionen sind sehr ähnlich. In allen drei Fällen ist eine Region ein Vektor von nicht-negativen ganzen Zahlen, dessen Einträge direkt die Komponenten eines Stellentripels festlegen. Über die Schaltregel von Petrinetzen werden geeignete Forderungen an diese Vektoren gestellt, welche garantieren, dass das entsprechende Stellentripel zulässig ist.
Zur Einführung einer Regionendefinition für Mengen von Schrittfolgen wollen wir dementsprechend auch hier Stellentripel durch Vektoren darstellen.

DEFINITION 4.3.5 (KORRESPONDENZ ZWISCHEN STELLENTRIPELN UND VEKTOREN)
Sei $T = \{t_1, \ldots, t_m\}$ *eine geordnete Menge von Transitionen. Jeder Vektor* $\mathbf{x} = (x_0, \ldots, x_{2m}) \in \mathbb{N}^{2m+1}$ *definiert bzgl.* $T = \{t_1, \ldots, t_m\}$ *ein korrespondierendes Stellentripel* $\overrightarrow{\mathbf{x}}$ *durch* $\overrightarrow{\mathbf{x}}_0 = x_0$, $^\circ\overrightarrow{\mathbf{x}}(t_i) := x_i$ *und* $\overrightarrow{\mathbf{x}}^\circ(t_i) := x_{m+i}$ *für* $1 \leqslant i \leqslant m$. *Für ein Stellentripel* st *heißt umgekehrt der eindeutige Vektor* $\mathbf{x} \in \mathbb{N}^{2m+1}$ *mit* $\overrightarrow{\mathbf{x}} = \mathrm{st}$ *der zu* st *bzgl.* $T = \{t_1, \ldots, t_m\}$ *korrespondierende Vektor.*

Definition 4.3.5 (Korrespondenz zwischen Stellentripeln und Vektoren)

Entsprechend dieser Definition lassen sich Stellen eines Netzes dadurch repräsentieren, dass die Anfangsmarkierung der Stelle sowie die Beziehungen der Stelle zu den Transitionen des Netzes in einem Vektor kodiert werden. Daher wird eine Regionendefinition, welche auf dieser Repräsentation von Stellen basiert, als Transitions-Regionen-Definition bezeichnet.

BEISPIEL: *Zu dem in Abbildung 59 links illustrierten Stellentripel* $(1, 2a, a + b)$ *korrespondiert bzgl. der Transitionsreihenfolge* a, b, c *der Vektor* $(1, 2, 0, 0, 1, 1, 0)$. *Zu dem rechts illustrierten Stellentripel* $(1, a, a + b)$ *korrespondiert der Vektor* $(1, 1, 0, 0, 1, 1, 0)$.

Beispiel

Eine Transitions-Region einer Menge von Schrittfolgen ist entsprechend obigen Überlegungen ein Vektor von nicht-negativen ganzen Zahlen, dessen Einträge die Anfangsmarkierung einer Stelle sowie die gewichteten Verbindungskanten der Stelle von und zu einer gegebenen Menge von Transitionen repräsentieren. Ähnlich wie in den drei obig zitierten Regionenkonzepten müssen noch geeignete Anforderungen an eine Transitions-Region gestellt werden. Hierzu formulieren wir Ungleichungen, welche sicherstellen, dass durch eine zu einer Transitions-Region gehörigen Stelle die Aktiviertheit keiner der gegebenen Schrittfolgen verhindert wird. Es ergibt sich somit für jeden Schritt einer der gegebenen Schrittfolgen eine Ungleichung, welche die Aktiviertheitsbedingung für den Schritt innerhalb der Schrittfolge widerspiegelt.

DEFINITION 4.3.6 (TRANSITIONS-REGION FÜR SCHRITTFOLGEN)
Sei S *eine Menge von Schrittfolgen und* $T = \{t_1, \ldots, t_m\}$ *die Menge der Transitionen, welche in* S *vorkommen, in geordneter Form. Eine Transitions-Region von* S *(bzgl. der gewählten Ordnung der Transitionsmenge* $T = \{t_1, \ldots, t_m\}$*) ist ein Vektor* $\mathbf{r} = (r_0, \ldots, r_{2m}) \in \mathbb{N}^{2m+1}$, *welcher für jede Schrittfolge* $\sigma = \tau_1 \ldots \tau_n \in S$ *und jedes* $j \in \{1, \ldots, n\}$ *die folgende Ungleichung erfüllt:*

Definition 4.3.6 (Transitions-Region für Schrittfolgen)

$$(ST) \quad r_0 + \sum_{i=1}^{m} ((\tau_1 + \ldots + \tau_{j-1})(t_i) \cdot r_i - (\tau_1 + \ldots + \tau_j)(t_i) \cdot r_{m+i}) \geqslant 0.$$

Jede Transitions-Region $\mathbf{r}$ *von* $\mathcal{S}$ *bzgl.* $T = \{t_1, \ldots, t_m\}$ *definiert ein korrespondierendes Stellentripel* $\overrightarrow{\mathbf{r}}$.

In folgendem Lemma wird gezeigt, dass diese Regionendefinition zur Synthese von S/T-Netzen aus Mengen von Schrittfolgen in dem Sinne geeignet ist, dass die zu Regionen korrespondierenden Stellentripel genau die bzgl. der Menge von Schrittfolgen zulässigen Stellentripel sind.

Lemma 4.3.5

LEMMA 4.3.5
Sei $\mathcal{S}$ *eine Menge von Schrittfolgen und* $T = \{t_1, \ldots, t_m\}$ *die Menge der Transitionen, welche in* $\mathcal{S}$ *vorkommen, in geordneter Form. Ein Stellentripel über* T *ist genau dann zulässig bzgl.* $\mathcal{S}$*, wenn es zu einer Transitions-Region von* $\mathcal{S}$ *bzgl.* $T = \{t_1, \ldots, t_m\}$ *korrespondiert.*

BEWEIS: Sei $\sigma = \tau_1 \ldots \tau_n \in \mathcal{S}$ und $j \in \{1, \ldots, n\}$. Sei weiter $\mathbf{r} \in \mathbb{N}^{2m+1}$. Wir formen (ST) in eine Eigenschaft von $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ um. Zuerst werden die Einträge von $\mathbf{r}$ durch die entsprechende Anfangsmarkierung und die entsprechenden Kantengewichte ausgedrückt. Dann werden die betrachteten Summen geeignet aufgeteilt und im nächsten Schritt in entsprechende Multivorbereiche und Multinachbereiche übersetzt. Zuletzt wird verwendet, dass $p_{\overrightarrow{\mathbf{r}}}$ die einzige Stelle von $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ ist. Es gilt also: (ST) $\iff m_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}) + \sum_{i=1}^{m}((\tau_1 + \ldots + \tau_{j-1})(t_i) \cdot W_{\overrightarrow{\mathbf{r}}}(t_i, p_{\overrightarrow{\mathbf{r}}}) - (\tau_1 + \ldots + \tau_j)(t_i) \cdot W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, t_i)) \geqslant 0 \iff m_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}) + \sum_{i=1}^{m}((\tau_1 + \ldots + \tau_{j-1})(t_i) \cdot W_{\overrightarrow{\mathbf{r}}}(t_i, p_{\overrightarrow{\mathbf{r}}})) - \sum_{i=1}^{m}((\tau_1 + \ldots + \tau_{j-1})(t_i) \cdot W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, t_i)) \geqslant \sum_{i=1}^{m}(\tau_j(t_i) \cdot W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, t_i)) \iff m_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}) + (\tau_1 + \ldots + \tau_{j-1})^\circ(p_{\overrightarrow{\mathbf{r}}}) - {}^\circ(\tau_1 + \ldots + \tau_{j-1})(p_{\overrightarrow{\mathbf{r}}}) \geqslant {}^\circ\tau_j(p_{\overrightarrow{\mathbf{r}}}) \iff m_{\overrightarrow{\mathbf{r}}} + (\tau_1 + \ldots + \tau_{j-1})^\circ - {}^\circ(\tau_1 + \ldots + \tau_{j-1}) \geqslant {}^\circ\tau_j$.
Damit lässt sich das Lemma nun leicht nachweisen. Sei $\overrightarrow{\mathbf{r}}$ das zu einer Transitions-Region $\mathbf{r}$ von $\mathcal{S}$ bzgl. $T = \{t_1, \ldots, t_m\}$ korrespondierende Stellentripel und $\sigma = \tau_1 \ldots \tau_n \in \mathcal{S}$. Wir betrachten das atomare Netz $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ und setzen $m_j = m_{\overrightarrow{\mathbf{r}}} + (\tau_1 + \ldots + \tau_j)^\circ - {}^\circ(\tau_1 + \ldots + \tau_j)$ für $j \in \{1, \ldots, n\}$. Aus obiger Umformung von (ST) ergibt sich dann direkt $m_{\overrightarrow{\mathbf{r}}} \xrightarrow{\tau_1} m_1 \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} m_n$, d.h. σ ist entsprechend Definition 3.2.4 bzgl. $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ aktiviert. Also ist $\overrightarrow{\mathbf{r}}$ zulässig.
Sei st ein zulässiges Stellentripel bzgl. $\mathcal{S}$ und $\mathbf{r} \in \mathbb{N}^{2m+1}$ der eindeutige zu st bzgl. $T = \{t_1, \ldots, t_m\}$ korrespondierende Vektor, i.e. $\overrightarrow{\mathbf{r}} = st$. Sei $\sigma = \tau_1 \ldots \tau_n \in \mathcal{S}$ und $j \in \{1, \ldots, n\}$. Da σ bzgl. $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ aktiviert ist, folgt $m_{\overrightarrow{\mathbf{r}}} + (\tau_1 + \ldots + \tau_{j-1})^\circ - {}^\circ(\tau_1 + \ldots + \tau_{j-1}) \geqslant {}^\circ\tau_j$ und damit nach obiger Umformung auch (ST). Also ist $\mathbf{r}$ eine Transitions-Region von $\mathcal{S}$ bzgl. $T = \{t_1, \ldots, t_m\}$ und st korrespondiert zu dieser Region. □

Aufgrund der eins-zu-eins-Korrespondenz zwischen Stellentripeln und Vektoren aus Definition 4.3.5 ergibt sich aus letzterem Lemma sogar eine eins-zu-eins-Korrespondenz zwischen der Menge der zulässigen Stellentripel bzgl. $\mathcal{S}$ und der Menge der Transitions-Regionen von $\mathcal{S}$ bzgl. $T = \{t_1, \ldots, t_m\}$.

Beispiel

BEISPIEL: *Es lässt sich nachprüfen, dass der im letzten Beispiel zuerst betrachtete Vektor* $(1, 2, 0, 0, 1, 1, 0)$ *eine Transitions-Region der Menge* $\{bc, aabc, abac, abca, aba + c, aa + bc\}$ *bzgl.* $\{a, b, c\}$ *ist. Hierzu muss für jeden Schritt einer der gegebenen Schrittfolgen eine entsprechende* (ST)*-Ungleichung betrachtet werden. Für eine Schrittfolge der Länge* n *werden also* n *Ungleichungen untersucht. Insgesamt müssen somit* 20 *Ungleichungen geprüft werden. Beispielsweise ergibt sich durch Einsetzen in die* (ST)*-Ungleichung von* $\sigma = aa + bc$ *und* $j = 2$ *die gültige Beziehung:* $1 + 2 - 2 +$

$0-1+0-0 \geqslant 0$. *Das in Abbildung 59 links dargestellte zu* $(1,2,0,0,1,1,0)$ *korrespondierende Stellentripel ist zulässig.*
Der im letzten Beispiel betrachtete Vektor $(1,1,0,0,1,1,0)$ *ist keine Transitions-Region, da sich durch Einsetzen in die* (ST)*-Ungleichung von* $\sigma = aa + bc$ *und* $j = 2$ *die folgende ungültige Beziehung ergibt:* $1+1-2+0-1+0-0 \geqslant 0$. *Das in Abbildung 59 rechts dargestellte korrespondierende Stellentripel ist dementsprechend nicht zulässig.*

Entsprechend den anfänglichen Überlegungen dieses Abschnittes leiten wir nun aus der Definition von Transitions-Regionen für Mengen von Schrittfolgen die folgende Transitions-Regionen-Definition für partielle Sprachen ab.

DEFINITION 4.3.7 (SCHRITT-TRANSITIONS-REGION) *Definition 4.3.7 (Schritt-Transitions-Region)*
Sei $\mathcal{L}$ *eine partielle Sprache mit Beschriftungsmenge* $T = \{t_1, \ldots, t_m\}$. *Eine Transitions-Region* $\mathbf{r}$ *von* $\{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$ *(bzgl. der gewählten Ordnung der Transitionsmenge* $T = \{t_1, \ldots, t_m\}$*) heißt Schritt-Transitions-Region von* $\mathcal{L}$ *(bzgl.* $T = \{t_1, \ldots, t_m\}$*).*
Jede Schritt-Transitions-Region $\mathbf{r}$ *von* $\mathcal{L}$ *bzgl.* $T = \{t_1, \ldots, t_m\}$ *definiert ein korrespondierendes Stellentripel* $\overrightarrow{\mathbf{r}}$.

Es bleibt noch, formal nachzuweisen, dass sich mit dieser Regionendefinition tatsächlich die Menge der zulässigen Stellentripel charakterisieren lässt.

SATZ 4.3.1 *Satz 4.3.1*
Sei $\mathcal{L}$ *eine partielle Sprache mit endlicher Beschriftungsmenge* $T = \{t_1, \ldots, t_m\}$. *Ein Stellentripel über* T *ist genau dann zulässig bzgl.* $\mathcal{L}$*, wenn es zu einer Schritt-Transitions-Region von* $\mathcal{L}$ *bzgl.* $T = \{t_1, \ldots, t_m\}$ *korrespondiert.*

BEWEIS: Nach Lemma 4.3.4 ist ein Stellentripel über T genau dann zulässig bzgl. $\mathcal{L}$, wenn es zulässig bzgl. $\{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$ ist. Nach Lemma 4.3.5 ist ein Stellentripel über T genau dann zulässig bzgl. $\{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$, wenn es zu einer Transitions-Region von $\{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$ bzgl. $T = \{t_1, \ldots, t_m\}$ und damit zu einer Schritt-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ korrespondiert. □

Wiederum ergibt sich aufgrund von Definition 4.3.5 sogar eine eins-zu-eins-Korrespondenz zwischen der Menge der zulässigen Stellentripel bzgl. $\mathcal{L}$ und der Menge der Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$.

BEISPIEL: *Die für die Definition einer Schritt-Transitions-Region der partiellen Sprache aus Abbildung 4 relevante Menge von Schrittfolgen ist die Menge* $\{bc, aabc, abac, abca, aba+c, aa+bc\}$. *Entsprechend dem letzten Beispiel ist damit* $(1,2,0,0,1,1,0)$ *eine Schritt-Transitions-Region, während* $(1,1,0,0,1,1,0)$ *keine Schritt-Transitions-Region ist. Wie in Abbildung 59 dargestellt, ist auch nur das zu ersterem Vektor korrespondierende Stellentripel zulässig bzgl. der partiellen Sprache.* *Beispiel*

Zulässige Stellentripel bzgl. $\mathcal{L}$ lassen sich nach den bisherigen Ausführungen also aus Schritt-Transitions-Regionen gewinnen. Zur Berechnung von Schritt-Transitions-Regionen können die Ungleichungen (ST) zu einem homogenen linearen Ungleichungssystem zusammengefasst werden. Falls $\mathcal{L}$ endlich ist, besteht das Ungleichungssystem aus endlich vielen Ungleichungen und lässt sich somit durch eine Matrix repräsentieren. Dies führt zur folgenden linear algebraischen Charakterisierung der Menge aller Schritt-Transitions-Regionen von $\mathcal{L}$.

Lemma 4.3.6

Lemma 4.3.6
Sei $\mathcal{L}$ eine endliche partielle Sprache mit Beschriftungsmenge $T = \{t_1, \ldots, t_m\}$. Die Menge der Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ entspricht der Menge der ganzzahligen Lösungen des homogenen linearen Ungleichungssystems

$$\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}.$$

Die Matrix $\mathbf{A}_{\mathcal{L}}^{ST}$ setzt sich aus einem Zeilenvektor $\mathbf{a}_\sigma^j = (a_{\sigma,0}^j, \ldots, a_{\sigma,2m}^j)$ für jedes $\sigma = \tau_1 \ldots \tau_n \in \{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$ und $j \in \{1, \ldots, n\}$ zusammen. Der Zeilenvektor $\mathbf{a}_\sigma^j$ ist gegeben durch

$$a_{\sigma,i}^j = \begin{cases} 1 & \textit{für } i = 0, \\ (\tau_1 + \ldots + \tau_{j-1})(t_i) & \textit{für } i = 1, \ldots, m \\ -(\tau_1 + \ldots + \tau_j)(t_{i-m}) & \textit{für } i = m+1, \ldots, 2m. \end{cases}$$

Beweis: Der Zeilenvektor $\mathbf{a}_\sigma^j$ ist derart definiert, dass $\mathbf{a}_\sigma^j \cdot \mathbf{x} \geqslant \mathbf{0} \Leftrightarrow$ (ST) bzgl. σ und j. Gemäß Definition 4.3.6 entspricht daher die Menge der nicht-negativen ganzzahligen Lösungen von $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}$ der Menge der Transitions-Regionen der betrachteten Menge von Schrittfolgen $\{\sigma(bpo) \mid bpo \in Slin(\mathcal{L})\}$ bzgl. $T = \{t_1, \ldots, t_m\}$. Entsprechend Definition 4.3.7 ist diese Menge von Transitions-Regionen gerade die Menge der Schritt-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$. Unter Beachtung, dass der Anteil $\mathbf{x} \geqslant \mathbf{0}$ des betrachteten Ungleichungssystems einer Forderung nach nicht-negativen Lösungen entspricht, ergibt sich die Behauptung. □

$$\begin{array}{c} \text{b} \\ \text{bc} \\ \text{a} \\ \text{aa} \\ \text{aab} \\ \text{aabc} \\ \text{a} \\ \text{ab} \\ \text{aba} \\ \text{abac} \\ \text{a} \\ \text{ab} \\ \text{abc} \\ \text{abca} \\ \text{a} \\ \text{ab} \\ \text{aba+c} \\ \text{a} \\ \text{aa+b} \\ \text{aa+bc} \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -2 & 0 & 0 \\ 1 & 2 & 0 & 0 & -2 & -1 & 0 \\ 1 & 2 & 1 & 0 & -2 & -1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 \\ 1 & 1 & 1 & 0 & -2 & -1 & 0 \\ 1 & 2 & 1 & 0 & -2 & -1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 \\ 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -2 & -1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 \\ 1 & 1 & 1 & 0 & -2 & -1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -2 & -1 & 0 \\ 1 & 2 & 1 & 0 & -2 & -1 & -1 \end{pmatrix} \bullet \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Abbildung 61: Ungleichungssystem $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}$.

BEISPIEL: *Abbildung 61 zeigt das Ungleichungssystem* $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}$ *für die partielle Sprache aus Abbildung 4. Die Matrix* $\mathbf{A}_{\mathcal{L}}^{ST}$ *hat 20 Zeilen entsprechend den 20 schon angesprochenen* (ST)*-Ungleichungen (die jeweils zugehörigen Schrittfolgen* $\tau_1 \dots \tau_j$ *sind neben der Matrix angegeben) und* $7 = 2|\{a, b, c\}| + 1$ *Spalten. Die zuvor betrachtete Schritt-Transitions-Region* $(1, 2, 0, 0, 1, 1, 0)$ *ist ein Beispiel für eine Lösung des Ungleichungssystems.* *Beispiel*

Zur algorithmischen Verwendung der Charakterisierung aus Lemma 4.3.6 ist die Größe der Matrix $\mathbf{A}_{\mathcal{L}}^{ST}$ wichtig. Während die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{ST}$ durch $2|T| + 1$ gegeben ist, kann die Anzahl der Zeilen sehr groß werden. Das Problem dieses Ansatzes ist, dass die Anzahl der Schrittlinearisierungen und auch der entsprechenden Schrittfolgen einer BPO exponentiell in der Größe der BPO, d.h. in der Anzahl der Knoten und Kanten der BPO, sein kann. Damit kann die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{ST}$ exponentiell in der Größe der Eingabe $\mathcal{L}$ sein.

BEISPIEL: *BPOs mit wenigen Abhängigkeitsbeziehungen haben häufig eine exponentielle Anzahl an Schrittfolgen. Dies gilt insbesondere auch für BPOs, bei denen alle Ereignisse nebenläufig sind. Ein interessantes Beispiel für eine BPO mit einer exponentiellen Anzahl an Schrittfolgen zeigt Abbildung 62. Die BPO hat* $2n$ *Ereignisse. Hier kann beispielsweise eine beliebige Teilmenge der* n a*-Ereignisse als erster Schritt gewählt werden. Dies sind schon* 2^n *Möglichkeiten.* *Beispiel*

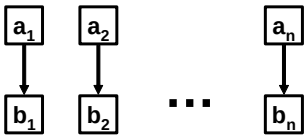

Abbildung 62: BPO mit $2n$ Ereignissen.

Wir schlagen in Bemerkung 4.3.1 vier Optimierungsverfahren vor, um die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{ST}$ zu reduzieren. Dazu benötigen wir noch das folgende Lemma.

LEMMA 4.3.7 *Lemma 4.3.7*
Sei $\mathcal{L}$ *eine partielle Sprache mit endlicher Beschriftungsmenge* T*. Ein Stellentripel über* T *ist genau dann zulässig bzgl.* $\mathcal{L}$*, wenn es zulässig bzgl.* $\{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Rep}(\text{Slin}(\mathcal{L}))\}$ *ist.*

BEWEIS: Nach Lemma 4.3.4 ist ein Stellentripel über T genau dann zulässig bzgl. $\mathcal{L}$, wenn es zulässig bzgl. $\{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\mathcal{L})\}$ ist. Aufgrund der Äquivalenz der Aktiviertheit von schrittweise linearen BPOs und deren assoziierten Schrittfolgen ist dies genau dann der Fall, wenn es zulässig bzgl. $\text{Slin}(\mathcal{L})$ ist. Nach Korollar 4.3.1 ist dies äquivalent zur Zulässigkeit bzgl. $\text{Rep}(\text{Slin}(\mathcal{L}))$ und dementsprechend zur Zulässigkeit bzgl. $\{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Rep}(\text{Slin}(\mathcal{L}))\}$. □

BEMERKUNG 4.3.1 (OPTIMIERUNGSVORSCHLÄGE) *Bemerkung 4.3.1 (Optimierungsvorschläge)*
Die folgenden Optimierungen des Schritt-Transitions-Regionen-Begriffs aus Definition 4.3.7 und der entsprechenden linear algebraischen Charakterisierung aus Lemma 4.3.6 sind möglich.

- *Die einfachste Möglichkeit der Verkleinerung von* $\mathbf{A}_{\mathcal{L}}^{ST}$ *ergibt sich durch folgende Überlegung. Bei der Betrachtung einer Schrittfolge* $\sigma = \tau_1 \dots \tau_n \in S$ *zusammen mit* $j \in \{1, \dots, n\}$ *und einer anderen Schrittfolge*

$\sigma' = \tau'_1 \dots \tau'_l$ *zusammen mit* $k \in \{1, \dots, l\}$ *ergeben sich häufig identische Schrittfolgen* $\tau_1 \dots \tau_j$ *und* $\tau'_1 \dots \tau'_k$*. In diesem Fall sind natürlich auch die entsprechenden* (ST)*-Ungleichungen identisch. Daher ist es sinnvoll, eine solche Schrittfolge nur einmal zu berücksichtigen und somit entsprechende doppelte Zeilenvektoren in* $\mathbf{A}_{\mathcal{L}}^{ST}$ *zu vermeiden.*

- *Falls in der im letzten Absatz dargestellten Situation die zwei Schrittfolgen* $\tau_1 \dots \tau_j$ *und* $\tau'_1 \dots \tau'_k$ *zwar nicht identisch sind, aber* $\tau_j = \tau'_k$ *und* $\sum_{i=1}^{j-1} \tau_i = \sum_{i=1}^{k-1} \tau'_i$ *gilt, so erzeugen sie ebenfalls zwei identische* (ST)*-Ungleichungen. Auch in diesem Fall können also Zeilenvektoren aus* $\mathbf{A}_{\mathcal{L}}^{ST}$ *weggelassen werden (ähnlich wie in [63, 18, 65]).*

- *Die Überlegungen des letzten Absatzes lassen sich noch verfeinern, um weitere Ungleichungen entfernen zu können. Beispielsweise kann in obiger Situation die zu* $\tau'_1 \dots \tau'_k$ *gehörige* (ST)*-Ungleichung auch ohne Weiteres weggelassen werden, wenn anstelle von* $\tau_j = \tau'_k$ *die Beziehung* $\tau_j \geqslant \tau'_k$ *gilt. Für derartige Vorgehensweisen eignet sich jedoch die Regionendefinition des nächsten Abschnittes besser, so dass wir hier nicht mehr näher auf solche fortschrittlichen Optimierungsmöglichkeiten auf der Ebene der Schrittfolgen eingehen möchten.*

- *Generell ist es entsprechend Korollar 4.3.1 bei allen in diesem Abschnitt betrachteten Regionendefinitionen möglich, nur die BPOs aus der Repräsentationsmenge* $\mathrm{Rep}(\mathcal{L})$ *anstelle aller BPOs aus* $\mathcal{L}$ *zu betrachten. Dadurch lassen sich bei der Menge der Schrittlinearisierungen, welche die entscheidende Rolle beim Begriff der Schritt-Transitions-Region spielt, aber bestenfalls Präfixe vermeiden. Entsprechend ergibt sich durch dieses Vorgehen nur eine geringe bzw. im Falle, dass der meist sinnvolle erste Optimierungsvorschlag verwendet wird, gar keine Reduzierung der Zeilen von* $\mathbf{A}_{\mathcal{L}}^{ST}$*. Allerdings kann eine Reduzierung des Aufwandes zur Berechnung der Schrittlinearisierungen ausgehend von der kleineren Menge* $\mathrm{Rep}(\mathcal{L})$ *erzielt werden. Demgegenüber steht aber der zusätzliche Aufwand zur Berechnung von* $\mathrm{Rep}(\mathcal{L})$ *aus* $\mathcal{L}$*. Außerdem ist* $\mathrm{Rep}(\mathcal{L})$ *aus Anwendungssicht typischerweise nicht sehr viel kleiner als* $\mathcal{L}$*, da es für einen Anwender ohnehin sinnvoll und intuitiv ist, möglichst nur* $\mathrm{Rep}(\mathcal{L})$ *als Eingabe für ein Syntheseproblem zu spezifizieren.*

- *Um tatsächlich die Anzahl der Zeilen von* $\mathbf{A}_{\mathcal{L}}^{ST}$ *wesentlich zu reduzieren, stellt dagegen das Betrachten von* $\mathrm{Rep}(\mathrm{Slin}(\mathcal{L}))$*, also der Repräsentationsmenge von* $\mathrm{Slin}(\mathcal{L})$*, eine vielversprechende Optimierung dar, d.h. es sollten auf der Ebene der Schrittlinearisierungen Präfixe und Sequentialisierungen weggelassen werden. Dies bedeutet formal, dass nur Zeilenvektoren* $\mathbf{a}_{\sigma}^{j}$ *für alle Schrittfolgen* σ *der Menge* $\{\sigma(\mathrm{bpo}) \mid \mathrm{bpo} \in \mathrm{Rep}(\mathrm{Slin}(\mathcal{L}))\}$ *anstelle der größeren Menge* $\{\sigma(\mathrm{bpo}) \mid \mathrm{bpo} \in \mathrm{Slin}(\mathcal{L})\}$ *betrachtet werden. Entsprechend Lemma 4.3.7 ist die Betrachtung dieser Zeilenvektoren auch ausreichend. Die Berechnung der kleineren Menge* $\{\sigma(\mathrm{bpo}) \mid \mathrm{bpo} \in \mathrm{Rep}(\mathrm{Slin}(\mathcal{L}))\}$ *an Schrittfolgen erfordert allerdings zusätzlichen Aufwand. Natürlich können anstatt* $\{\sigma(\mathrm{bpo}) \mid \mathrm{bpo} \in \mathrm{Rep}(\mathrm{Slin}(\mathcal{L}))\}$ *auch alle Mengen* $\mathcal{S}$ *an Schrittfolgen mit der Eigenschaft* $\{\sigma(\mathrm{bpo}) \mid \mathrm{bpo} \in \mathrm{Rep}(\mathrm{Slin}(\mathcal{L}))\} \subseteq \mathcal{S} \subseteq \{\sigma(\mathrm{bpo}) \mid \mathrm{bpo} \in \mathrm{Slin}(\mathcal{L})\}$ *verwendet werden. Dies ermöglicht das Anwenden von Heuristiken, welche darauf abzielen, eine möglichst kleine, aber nicht notwendigerweise minimale, solche Menge* $\mathcal{S}$ *zu erzeugen.*

- *Aus einem linear algebraischen Blickwinkel heraus bedeuten die bisherigen Optimierungsvorschläge das Weglassen bestimmter redundan-*

ter Ungleichungen des Ungleichungssystems $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$*, da Ungleichungen weggelassen werden und sich dabei die Lösungsmenge des Ungleichungssystems nicht verändert (letzteres lässt sich aus der eins-zu-eins Korrespondenz zwischen Stellentripeln und Vektoren aus Definition 4.3.5 folgern). Im Allgemeinen lassen sich mit den bisherigen Optimierungsvorschlägen aber nicht alle redundanten Ungleichungen des Ungleichungssystems* $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ *entfernen. Deswegen ist es unabhängig von obigen Optimierungsvorschlägen sinnvoll, nach redundanten Ungleichungen des Ungleichungssystem zu suchen und die entsprechenden Zeilen der Matrix* $\mathbf{A}_{\mathcal{L}}^{ST}$ *zu entfernen. Natürlich bleibt dadurch eine korrekte linear algebraische Charakterisierung der Menge der Schritt-Transitions-Regionen erhalten. Entsprechend Abschnitt 3.4 gibt es verschiedene Verfahren mit verschieden hohem rechnerischen Aufwand zum Auffinden redundanter Ungleichungen. Wie im Rahmen des dritten Optimierungsvorschlages beschrieben, sind auch weitere Verfahren auf der Ebene der Schrittfolgen denkbar, um redundante Ungleichungen zu entfernen.*

Beispiel

BEISPIEL: *Wir betrachten die partielle Sprache aus Abbildung 4 und das entsprechende Ungleichungssystem über den nicht-negativen ganzen Zahlen aus Abbildung 61. Es fällt sofort auf, dass einige der links dargestellten Schrittfolgen mehrfach vorkommen. So findet sich die Schrittfolge* a *in den Zeilen* 3, 7, 11, 15 *und* 18 *und die Schrittfolge* ab *in den Zeilen* 8, 12 *und* 16*. Die entsprechenden Zeilen von* $\mathbf{A}_{\mathcal{L}}^{ST}$ *sind also identisch. Entsprechend dem ersten Optimierungsvorschlag können beispielsweise die Zeilen* 3, 7, 11 *und* 18 *sowie* 8 *und* 12 *von der Matrix entfernt werden.*
Die Situation des zweiten Optimierungsvorschlages findet sich in den Zeilen 6, 10 *und* 20*. Die entsprechenden Schrittfolgen* $aabc$, $abac$ *und* $aa + bc$ *erzeugen ebenfalls identische Ungleichungen.*
Auch die im dritten Optimierungsvorschlag geschilderte Situation lässt sich mehrfach finden, beispielsweise für die Schrittfolgen ab *aus Zeile* 16 *und* $aa + b$ *aus Zeile* 19*.*
Im Hinblick auf den vierten Optimierungsvorschlag gilt in unserem Beispiel $\mathrm{Rep}(\mathcal{L}) = \mathcal{L}$*, so dass dieser hier nicht anwendbar ist.*
Nun betrachten wir den fünften Optimierungsvorschlag näher. Im Allgemeinen lässt sich dieser folgendermaßen veranschaulichen. Das Verwenden der Repräsentationsmenge $\mathrm{Rep}(\mathrm{Slin}(\mathcal{L}))$ *von* $\mathrm{Slin}(\mathcal{L})$ *bedeutet auf der Ebene der Schrittfolgen, dass, wenn eine Schrittfolge* $\sigma = \tau_1 \dots \tau_n$ *betrachtet wird, dann „Präfixschrittfolgen" der Form* $\tau_1 \dots \tau_{k-1} \tau_k'$ *mit* $k \leqslant n$ *und* $\tau_k' \leqslant \tau_k$ *und „Sequentialisierungsschrittfolgen" der Form* $\tau_1^1 \dots \tau_1^{l_1} \dots \tau_n^1 \dots \tau_n^{l_n}$ *mit* $\tau_k = \sum_{i=1}^{l_k} \tau_k^i$ *nicht berücksichtigt werden, da deren Aktiviertheit aus der Aktiviertheit von* $\sigma = \tau_1 \dots \tau_n$ *folgt. Mithilfe der Definition der* (ST)*-Ungleichungen lässt sich nachprüfen, dass ein Vektor über den nicht-negativen ganzen Zahlen, welcher alle aus einer Schrittfolge* $\sigma = \tau_1 \dots \tau_n$ *resultierenden* (ST)*-Ungleichungen erfüllt, auch tatsächlich allen aus einer „Präfixschrittfolge" oder einer „Sequentialisierungsschrittfolge" resultierenden* (ST)*-Ungleichungen genügt. Somit können von der ursprünglich zu betrachtenden Menge an Schrittfolgen* $\{bc, aabc, abac, abca, aba + c, aa + bc\}$ *die Schrittfolgen* $aabc$, $abac$ *und* $abca$ *weggelassen werden. In dem Ungleichungssystem können daher die Zeilen* 3 *bis* 14 *entfernt werden.*
Wenn zuerst der fünfte Optimierungsvorschlag angewendet wird und anschließend der erste, so enthält das resultierende Ungleichungssystem welches aus den Zeilen 1, 2, 15, 16, 17, 19 *und* 20 *besteht, noch immer redundante Ungleichungen. Beispielsweise lässt sich noch die ursprüngliche Zeile* 16 *entsprechend dem dritten Optimierungsvorschlag entfernen. Auch danach ist*

noch immer die ursprüngliche Zeile 20 redundant. Diese kann beispielsweise durch eine allgemeine Suche nach redundanten Ungleichungen entsprechend des sechsten Optimierungsvorschlages entdeckt und entfernt werden. Auf diese Weise entsteht das Ungleichungssystem aus Abbildung 63 mit fünf Ungleichungen, welches keine Redundanz mehr aufweist. Zu beachten ist bei diesem Beispiel allerdings, dass es nicht unbedingt praktikabel ist, all diese Optimierungen zu verwenden, da sie teilweise einen hohen Aufwand erfordern. Es wird sich zeigen, dass etliche dieser Optimierungen in natürlicher Weise im Rahmen der Regionendefinition des nächsten Unterabschnittes inherent und ohne weiteren Aufwand berücksichtigt werden.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 1 & 0 & -2 & -1 & -1 \\ 1 & 1 & 0 & 0 & -2 & -1 & 0 \end{pmatrix} \bullet \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Abbildung 63: Reduziertes Ungleichungssystem.

Im schlechtesten Fall bleibt auch bei der Verwendung aller vorgestellter Optimierungen die exponentielle Abhängigkeit der Größe der linearen Ungleichungssysteme von der Größe der Eingabe bestehen. Da dies in der exponentiellen Anzahl von Schrittlinearisierungen in der Größe einer BPO begründet ist, werden in den folgenden zwei Unterabschnitten Regionendefinitionen für partielle Sprachen vorgestellt, welche den Zwischenschritt der Schrittlinearisierungen vermeiden. Vielmehr beruhen diese Regionendefinitionen direkt auf der Struktur der BPOs der gegebenen partiellen Sprache.

4.3.2 *BPO-Transitions-Regionen*

Die Regionendefinition dieses Unterabschnittes ist der des letzten Unterabschnittes sehr ähnlich (sie basiert also auch auf Überlegungen aus [120, 173, 13, 18, 65, 63]). Im Gegensatz zu den an den klassischen Regionendefinitionen angelehnten Schritt-Transitions-Regionen wird allerdings eine fortschrittlichere direkt auf der Struktur der partiellen Sprache basierende Definition einer Transitions-Region eingeführt. Diese Definition verwendet die Charakterisierung der Aktiviertheit von BPOs aus Lemma 3.3.1. Entsprechend dieses Lemmas spielen die Schnitte der BPOs einer partiellen Sprache die zentrale Rolle bei der Regionendefinition dieses Unterabschnittes.
Im Rahmen der linear algebraischen Charakterisierung einer Regionendefinition werden durch die Betrachtung von Schnitten im Gegensatz zu dem im letzten Unterabschnitt vorherrschenden Fokus auf Schrittfolgen viele redundante Ungleichungen automatisch vermieden. Als Motivation für die folgenden Definitionen illustrieren wir dies kurz anhand des zweiten und fünften Optimierungsvorschlages aus Bemerkung 4.3.1. Der zweite Vorschlag zeigt, dass es zur Berücksichtigung einer (ST)-Ungleichung für $\sigma = \tau_1 \dots \tau_n \in \mathcal{S}$ zusammen mit $j \in \{1, \dots, n\}$

genügt, nur das gesamte Vorkommen der Transitionen $\pi = \sum_{i=1}^{j-1} \tau_i$ in $\tau_1 \dots \tau_{j-1}$ und den Schritt τ_j zu betrachten. Die Multimengen π nennen wir dann einen Präfixschritt, τ_j heißt auf π folgender Schritt. Wie im letzten Beispiel ausgeführt, wird durch den fünften Optimierungsvorschlag die Berücksichtigung von „Präfixschrittfolgen“ und „Sequentialisierungsschrittfolgen“ vermieden. Aus den entsprechenden Überlegungen lässt sich für die gerade eingeführten auf Präfixschritte folgenden Schritte folgern, dass es für Präfixschritte π und π' sowie auf π und π' folgende Schritte τ und τ' genügt, nur π zusammen mit τ zu betrachten und π' zusammen mit τ' zu ignorieren, falls $\pi' \geqslant \pi$ und $\tau'_j + (\pi' - \pi) \leqslant \tau_j$. Auch wenn wir diese allgemeine Beziehung erst bei den Optimierungsvorschlägen am Ende des Unterabschnittes im Detail verwenden werden, so wird hier durch den Spezialfall $\pi = \pi'$ deutlich, dass nur maximale auf einen Präfixschritt folgende Schritte τ_j berücksichtigt werden müssen. Da die Schnitte einer BPO die maximalen Mengen an nebenläufigen Ereignissen der BPO repräsentieren, ergibt sich auf der Ebene der partiellen Sprache die folgende Überlegung. Es genügt, nur die Schritte τ_j, welche dem Parikhvektor eines Schnittes einer BPO der Sprache entsprechen, zusammen mit dem Präfixschritt π, welcher dem Parikhvektor des Präfixes des verwendeten Schnittes entspricht, zu betrachten. Der Parikhvektor einer Menge von Ereignissen ist dabei die Multimenge von Transitionen, welche die Anzahlen des Vorkommens der Transitionen in der Menge wiedergibt. Durch diesen Ansatz werden viele redundante (ST)-Ungleichungen, welche entsprechend dem letzten Beispiel bei der Betrachtung von Schrittfolgen entstehen können, vermieden.

Beispiel

BEISPIEL: *Wir betrachten die Schrittfolge* $aa + bc$ *von* bpo_2 *aus Abbildung 4. Für* $j = 3$ *ergibt sich der Präfixschritt* $2a + b$ *und der darauf folgende Schritt* c. *Dies gilt genauso für die Schrittfolge* $abac$ *von* bpo_2 *und* $j = 4$. *In beiden Fällen entspricht hier der Präfixschritt* $2a + b$ *demselben Präfix von* bpo_2. *Für* $j = 2$ *ergeben sich für die beiden Schrittfolgen* $aa + bc$ *und* $abac$ *zum einen der Präfixschritt* a *und der darauf folgende Schritt* $a + b$ *und zum anderen der Präfixschritt* a *zusammen mit dem darauffolgenden Schritt* b. *Entsprechend des diskutierten Zusammenhanges* $\pi' \geqslant \pi$ *und* $\tau'_j + (\pi' - \pi) \leqslant \tau_j$ *ist das zweite Paar von Präfixschritt und darauffolgendem Schritt gegenüber dem ersten unnötig. Während der erste Schritt* $a + b$ *zu einem Schnitt von* bpo_2 *korrespondiert, ist dies für den zweiten Schritt* b *nicht der Fall.*

Entsprechend obiger Überlegungen bilden die folgenden Ausführungen die formale Grundlage für die einzuführende Regionendefinition.

DEFINITION 4.3.8 (PRÄFIXSCHRITT, RICHTIGE FORTSETZUNG, SCHNITTFORTSETZUNG)

Definition 4.3.8 (Präfixschritt, richtige Fortsetzung, Schnittfortsetzung)

Sei $\mathcal{L}$ *eine partielle Sprache mit endlicher Beschriftungsmenge* T. *Die Menge* $\Pi_{\mathcal{L}} = \{\pi \in \mathbb{N}^T \mid \exists bpo \in \mathcal{L}, bpo' = (V', <', l') \in Pref(bpo) : \pi = V'_{l'}\}$ *beinhaltet alle Parikhvektoren von Präfixen von BPOs aus* $\mathcal{L}$. *Ein Element* $\pi \in \Pi_{\mathcal{L}}$ *heißt Präfixschritt von* $\mathcal{L}$.
Für einen Präfixschritt π, *heißt eine Multimenge* $\tau \in \mathbb{N}^T$ *ein auf* π *folgender Schritt, falls es eine BPO* $bpo = (V, <, l) \in \mathcal{L}$, *eine Co-Menge* $C \subseteq V$ *und ein Präfix* $(V', <', l')$ *von* C *gibt, so dass* $\tau = C_l$ *und* $\pi = V'_{l'}$. *Falls* C *ein Schnitt ist, so heißt* τ *ein auf* π *folgender Schnitt.*
Wir bezeichnen $\mathcal{L}^{\Pi}_{co} = \{(\pi, \tau) \mid \pi \in \Pi_{\mathcal{L}}, \tau$ *ein auf* π *folgender Schritt*$\}$. *Ein Element* $(\pi, \tau) \in \mathcal{L}^{\Pi}_{co}$ *heißt richtige Fortsetzung von* $\mathcal{L}$. *Wir bezeichnen*

weiter $\mathcal{L}^{\Pi} = \{(\pi, \tau) \mid \pi \in \Pi_{\mathcal{L}}, \tau$ *ein auf* π *folgender Schnitt}. Ein Element* $(\pi, \tau) \in \mathcal{L}^{\Pi}$ *heißt Schnittfortsetzung von* $\mathcal{L}$.

In dem letzten Unterabschnitt haben wir durch eine geeignete Regionendefinition sichergestellt, dass alle Schrittfolgen der BPOs von $\mathcal{L}$ bzgl. einer zu einer Region korrespondierenden Stelle schalten können. In diesem Unterabschnitt ist nun entscheidend, dass alle richtigen Fortsetzungen von $\mathcal{L}$ schalten können. Daher definieren wir formal, unter welcher Bedingung ein Paar von Transitionsschritten (π, τ) in einem Netz aktiviert ist.

Definition 4.3.9 (Aktivierte Paare von Schritten)

Definition 4.3.9 (Aktivierte Paare von Schritten)
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes S/T-Netz. Ein Paar von Transitionsschritten* $(\pi, \tau) \in \mathbb{N}^T \times \mathbb{N}^T$ *heißt aktiviert in* (N, m_0), *falls* $m_0 - {}^\circ\pi + \pi^\circ \geqslant {}^\circ\tau$.

Beispiel

Beispiel: *Die Menge der Präfixschritte der partiellen Sprache aus Abbildung 4 ist* $\{0, b, b+c, a, a+b, 2a, 2a+b, a+b+c, 2a+b+c\}$. *Die ersten drei aufgeführten Schritte werden dabei von* bpo_1 *erzeugt. Die Menge der richtigen Fortsetzungen* (π, τ) *mit* $\tau \neq 0$ *ist* $\{(0, b), (0, a), (b, c), (a, b), (a, a), (a, a+b), (a+b, a), (a+b, c), (a+b, a+c), (2a, b), (2a+b, c), (a+b+c, a)\}$. *Die Menge der Schnittfortsetzungen ist die Teilmenge* $\{(0, b), (0, a), (b, c), (a, a+b), (a+b, a+c)\}$. *Die fünf Schnittfortsetzungen entsprechen den zwei Schnitten von* bpo_1 *und den drei Schnitten von* bpo_2.
Alle aufgezählten richtigen Fortsetzungen sind in dem Netz aus Abbildung 3 aktiviert. Nicht aktiviert ist beispielsweise das Paar von Transitionsschritten $(b+c, a)$, *da nach dem Schalten von* b *und* c *die Stelle* p_2 *keine Marke enthält. Dieses Paar ist allerdings in dem Netz aus Abbildung 57 aktiviert.*

Die bisherigen Überlegungen und Definitionen werden nun für eine entsprechende Transitions-Regionen-Definition verwendet. Es sollen also wie im letzten Unterabschnitt zulässige Stellentripel durch geeignete Vektoren, dessen Einträge die Anfangsmarkierung einer Stelle sowie die gewichteten Verbindungskanten der Stelle von und zu jeder Transitionen darstellen, repräsentiert werden. Um die Zulässigkeit eines Stellentripels zu gewährleisten, muss entsprechend Lemma 3.3.1 grob gesagt der Parikhvektor jedes Schnittes einer BPO einer partiellen Sprache in dem zugehörigen atomaren Netz nach dem Parikhvektor des Präfixes des Schnittes schalten können, d.h. jede Schnittfortsetzung muss in dem atomaren Netz aktiviert sein. Diese Anforderung lässt sich auf der Vektorebene durch entsprechende Ungleichungen repräsentieren. Dementsprechend ergibt sich die folgende Transitions-Regionen-Definition.

Definition 4.3.10 (BPO-Transitions-Region)

Definition 4.3.10 (BPO-Transitions-Region)
Sei $\mathcal{L}$ *eine partielle Sprache mit geordneter endlicher Beschriftungsmenge* $T = \{t_1, \ldots, t_m\}$. *Eine BPO-Transitions-Region von* $\mathcal{L}$ *(bzgl. der gewählten Ordnung der Transitionsmenge* $T = \{t_1, \ldots, t_m\}$*) ist ein Vektor* $\mathbf{r} = (r_0, \ldots, r_{2m}) \in \mathbb{N}^{2m+1}$, *welcher für jede Schnittfortsetzung* $(\pi, \tau) \in \mathcal{L}^{\Pi}$ *die folgende Ungleichung erfüllt:*

$$(\mathrm{BPOT}) \qquad r_0 + \sum_{i=1}^{m} (\pi(t_i) \cdot r_i - (\pi + \tau)(t_i) \cdot r_{m+i}) \geqslant 0.$$

Jede BPO-Transitions-Region $\mathbf{r}$ *von* $\mathcal{L}$ *bzgl.* $T = \{t_1, \ldots, t_m\}$ *definiert ein korrespondierendes Stellentripel* $\overrightarrow{\mathbf{r}}$.

SATZ 4.3.2 *Satz 4.3.2*
Sei $\mathcal{L}$ eine partielle Sprache mit Beschriftungsmenge $T = \{t_1, \ldots, t_m\}$. Ein Stellentripel über T ist genau dann zulässig bzgl. $\mathcal{L}$, wenn es zu einer BPO-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ korrespondiert.

BEWEIS: Sei $bpo = (V, <, l) \in \mathcal{L}$ und $C \subseteq V$ ein Schnitt von bpo. Sei weiter $(V', <', l')$ das Präfix von C. Entsprechend Definition 4.3.8 gilt dann $(\pi, \tau) \in \mathcal{L}^{\Pi}$ für $\tau = C_l$ und $\pi = V'_{l'}$. Sei $\mathbf{r} \in \mathbb{N}^{2m+1}$. Wir formen (BPOT) in eine Eigenschaft von $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ um. Zuerst werden die Einträge von $\mathbf{r}$ durch die entsprechende Anfangsmarkierung und die entsprechenden Kantengewichte ausgedrückt. Dann werden π und τ ersetzt, indem über die entsprechenden Knoten von C und V' summiert wird. Zuletzt werden diese Summen noch geeignet angeordnet und die Beziehung $V' = \{v \in V \mid v < C\}$ verwendet. Es gilt also: (BPOT) $\iff m_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}) + \sum_{i=1}^{m}(\pi(t_i) \cdot W_{\overrightarrow{\mathbf{r}}}(t_i, p_{\overrightarrow{\mathbf{r}}}) - (\pi + \tau)(t_i) \cdot W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, t_i)) \geqslant 0 \iff m_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}) + \sum_{v \in V'} W_{\overrightarrow{\mathbf{r}}}(l(v), p_{\overrightarrow{\mathbf{r}}}) - \sum_{v \in V'} W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, l(v)) - \sum_{v \in C} W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, l(v)) \geqslant 0 \iff m_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}) + \sum_{v \in V, v < C}(W_{\overrightarrow{\mathbf{r}}}(l(v), p_{\overrightarrow{\mathbf{r}}}) - W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, l(v))) \geqslant \sum_{v \in C} W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, l(v))$.
Damit lässt sich der Satz leicht nachweisen. Sei $\overrightarrow{\mathbf{r}}$ das zu einer BPO-Transitions-Region $\mathbf{r}$ von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ korrespondierende Stellentripel und $bpo = (V, <, l) \in \mathcal{L}$. Wir betrachten das atomare Netz $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$. Sei $C \subseteq V$ ein Schnitt von bpo. Obige Umformung von (BPOT) zeigt, dass entsprechend Lemma 3.3.1 die BPO bpo bzgl. $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ aktiviert ist. Also ist $\overrightarrow{\mathbf{r}}$ zulässig.
Sei st ein zulässiges Stellentripel bzgl. $\mathcal{L}$ und $\mathbf{r} \in \mathbb{N}^{2m+1}$ der eindeutige zu st bzgl. $T = \{t_1, \ldots, t_m\}$ korrespondierende Vektor, i.e. $\overrightarrow{\mathbf{r}} = st$. Sei $(\pi, \tau) \in \mathcal{L}^{\Pi}$. Dann existiert eine BPO $bpo = (V, <, l) \in \mathcal{L}$, ein Schnitt $C \subseteq V$ von bpo sowie ein eindeutiges Präfix $(V', <', l')$ von C, so dass $\tau = C_l$ und $\pi = V'_{l'}$. Da bpo bzgl. $(N_{\overrightarrow{\mathbf{r}}}, m_{\overrightarrow{\mathbf{r}}})$ aktiviert ist, ergibt sich $m_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}) + \sum_{v \in V, v < C}(W_{\overrightarrow{\mathbf{r}}}(l(v), p_{\overrightarrow{\mathbf{r}}}) - W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, l(v))) \geqslant \sum_{v \in C} W_{\overrightarrow{\mathbf{r}}}(p_{\overrightarrow{\mathbf{r}}}, l(v))$ und damit nach obiger Umformung auch (BPOT). Also ist $\mathbf{r}$ eine BPO-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ und st korrespondiert zu dieser Region. □

BEISPIEL: *Es lässt sich nachprüfen, dass der Vektor $(1, 2, 0, 0, 1, 1, 0)$ eine BPO-Transitions-Region der partiellen Sprache aus Abbildung 4 bzgl. $\{a, b, c\}$ ist. Hierzu muss für jede Schnittfortsetzung der Menge $\{(0, b), (0, a), (b, c), (a, a + b), (a + b, a + c)\}$ die entsprechende (BPOT)-Ungleichung betrachtet werden. Es müssen also 5 Ungleichungen geprüft werden. Beispielsweise ergibt sich durch Einsetzen in die (BPOT)-Ungleichung von $(a + b, a + c)$ die gültige Beziehung: $1 + 2 - 2 + 0 - 1 + 0 - 0 \geqslant 0$. Das in Abbildung 59 links dargestellte zu $(1, 2, 0, 0, 1, 1, 0)$ korrespondierende Stellentripel ist zulässig. Der Vektor $(1, 1, 0, 0, 1, 1, 0)$ ist keine BPO-Transitions-Region, da sich durch Einsetzen in die (BPOT)-Ungleichung von $(a + b, a + c)$ die folgende ungültige Beziehung ergibt: $1 + 1 - 2 + 0 - 1 + 0 - 0 \geqslant 0$. Das in Abbildung 59 rechts dargestellte korrespondierende Stellentripel ist dementsprechend nicht zulässig.* *Beispiel*

Wie bei Schritt-Transitions-Regionen ergibt sich aufgrund der eins-zu-eins-Korrespondenz zwischen Stellentripeln und Vektoren entsprechend Definition 4.3.5 aus Satz 4.3.2 sogar eine eins-zu-eins-Korrespondenz zwischen der Menge der BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ und der Menge der zulässigen Stellentripel bzgl. $\mathcal{L}$. Mit Definition 4.3.5 können wir darüber hinaus folgern, dass die Menge der BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ exakt der

Menge der Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ entspricht.
Trotz dieser Übereinstimmung stellen BPO-Transitions-Regionen einen neuen und eigenständigen Ansatz dar, welcher in einer neuen linear algebraischen Charakterisierung und damit in einer neuen Herangehensweise zur Berechnung von zulässigen Stellentripeln deutlich wird. Zur Berechnung von zulässigen Stellentripeln mithilfe von BPO-Transitions-Regionen werden hier nun die Ungleichungen (BPOT) zu einem homogenen linearen Ungleichungssystem zusammengefasst. Wir betrachten im Folgenden wiederum den Fall, dass $\mathcal{L}$ endlich ist. Dann ergibt sich die folgende linear algebraische Charakterisierung der Menge aller BPO-Transitions-Regionen von $\mathcal{L}$, welche sich von der linear algebraischen Charakterisierung der Schritt-Transitions-Regionen aus Lemma 4.3.6 unterscheidet.

Lemma 4.3.8

LEMMA 4.3.8
Sei $\mathcal{L}$ eine endliche partielle Sprache mit Beschriftungsmenge $T = \{t_1, \ldots, t_m\}$. Die Menge der BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ entspricht der Menge der ganzzahligen Lösungen des homogenen linearen Ungleichungssystems

$$\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}.$$

Die Matrix $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ setzt sich aus einem Zeilenvektor $\mathbf{a}_{(\pi,\tau)} = (a_{(\pi,\tau),0}, \ldots, a_{(\pi,\tau),2m})$ für jedes $(\pi,\tau) \in \mathcal{L}^{\Pi}$ zusammen. Der Zeilenvektor $\mathbf{a}_{(\pi,\tau)}$ ist gegeben durch

$$a_{(\pi,\tau),i} = \begin{cases} 1 & \text{für } i = 0, \\ \pi(t_i) & \text{für } i = 1, \ldots, m \\ -(\pi+\tau)(t_{i-m}) & \text{für } i = m+1, \ldots, 2m. \end{cases}$$

BEWEIS: Der Zeilenvektor $\mathbf{a}_{(\pi,\tau)}$ ist derart definiert, dass $\mathbf{a}_{(\pi,\tau)} \cdot \mathbf{x} \geqslant \mathbf{0} \Leftrightarrow$ (BPOT) bzgl. (π,τ). Gemäß Definition 4.3.10 entspricht daher die Menge der nicht-negativen ganzzahligen Lösungen von $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}} \cdot \mathbf{x} \geqslant \mathbf{0}$ der Menge der BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$. Unter Beachtung, dass der Anteil $\mathbf{x} \geqslant \mathbf{0}$ des betrachteten Ungleichungssystems einer Forderung nach nicht-negativen Lösungen entspricht, ergibt sich die Behauptung. □

Beispiel

BEISPIEL: *Das Ungleichungssystem $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}} \cdot \mathbf{x} \geqslant \mathbf{0}$ für die partielle Sprache aus Abbildung 4 ist in Abbildung 63 dargestellt. Ohne die Verwendung etwaiger Optimierungen ergibt sich also in natürlicher Weise dasjenige kleine Ungleichungssystem ohne Redundanz, welches wir im Falle von Schritt-Transitions-Regionen nur durch eine umfassende Anwendung von Reduktionsverfahren erhalten haben. Die fünf Zeilen der Matrix $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ entsprechen den fünf Schnittfortsetzungen $(0, b)$, (b, c), $(0, a)$, $(a + b, a + c)$ und $(a, a + b)$. Die Anzahl der Spalten ergibt sich durch $2|\{a, b, c\}| + 1 = 7$. Die BPO-Transitions-Region $(1, 2, 0, 0, 1, 1, 0)$ ist ein Beispiel für eine Lösung des Ungleichungssystems.*

Wie im Falle der Schritt-Transitions-Regionen ist die Größe der Matrix $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ zu algorithmischen Zwecken wichtig. Wiederum ist die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ durch $2|T| + 1$ gegeben, aber die Anzahl der Zeilen kann sehr groß werden. Das Problem ist hier, dass die Anzahl der Schnitte einer BPO exponentiell in der Größe der BPO, i.e. in der Anzahl der Knoten und Kanten der BPO, sein kann. Damit kann die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ exponentiell in der Größe der Eingabe $\mathcal{L}$ sein.

BEISPIEL: *Ein Schnitt der BPO aus Abbildung 62 ergibt sich dadurch, dass eine beliebige Teilmenge* $I \subseteq \{1, \ldots, n\}$ *gewählt wird. Diejenigen* a*-Ereignisse, deren Index aus* I *ist, bilden zusammen mit denjenigen* b*-Ereignissen, deren Index nicht aus* I *ist, einen Schnitt. Somit ist die Anzahl der Schnitte* 2^n*. Sie hängt also exponentiell von der Anzahl der Knoten der BPO, welche* $2n$ *beträgt, ab.* *Beispiel*

Wir schlagen in Bemerkung 4.3.2 drei Optimierungsverfahren vor, um die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ zu reduzieren. Dazu benötigen wir noch die folgenden Vorüberlegungen. In den Ausführungen im Vorfeld von Definition 4.3.8 haben wir hergeleitet, dass für $(\pi, \tau), (\pi', \tau') \in \mathbb{N}^T \times \mathbb{N}^T$ in bestimmten Fällen die Aktiviertheit von (π', τ') in einem Netz die Aktiviertheit von (π, τ) impliziert. In anderen Worten stellt in diesen Fällen die (BPOT)-Ungleichung von (π, τ) eine geringere Anforderung dar als die (BPOT)-Ungleichung von (π', τ'). Wir haben dabei die folgende binäre Relation $\ll$ auf $\mathbb{N}^T \times \mathbb{N}^T$ betrachtet:

$$(\pi', \tau') \ll (\pi, \tau) :\Longleftrightarrow (\pi', \tau') \neq (\pi, \tau) \wedge \pi \leqslant \pi' \wedge \tau' + \pi' \leqslant \tau + \pi.$$

Diese Relation soll hier nun formal untersucht werden.

LEMMA 4.3.9 *Lemma 4.3.9*
Es gelten die folgenden Aussagen über die Relation $\ll$*.*

- *Sei* $T = \{t_1, \ldots, t_m\}$ *und* $(\pi, \tau), (\pi', \tau') \in \mathbb{N}^T \times \mathbb{N}^T$ *mit* $(\pi', \tau') \ll (\pi, \tau)$*. Dann ist in dem Ungleichungssystem über den nicht-negativen ganzen Zahlen, welches aus den zwei* (BPOT)*-Ungleichungen von* (π', τ') *und* (π, τ) *besteht, die* (BPOT)*-Ungleichungen von* (π', τ') *redundant.*
- *Der gerichtete Graph* $(\mathbb{N}^T \times \mathbb{N}^T, \ll)$ *ist eine partielle Ordnung.*
- *Seien* $bpo = (V, <, l), bpo' = (V', <', l')$ *zwei BPOs mit* $bpo' \in PS(bpo)$*. Für* $(\pi', \tau') \in \{bpo'\}^{\Pi}$ *gilt* $(\pi', \tau') \in \{bpo\}^{\Pi}$ *oder es existiert* $(\pi, \tau) \in \{bpo\}^{\Pi}$ *mit* $(\pi', \tau') \ll (\pi, \tau)$*.*

BEWEIS: Die drei Aussagen des Lemmas werden nacheinander bewiesen.

- Nach Voraussetzung gilt $\pi \leqslant \pi'$ und $\tau' + \pi' \leqslant \tau + \pi$. Daraus können wir $r_0 + \sum_{i=1}^{m} (\pi'(t_i) \cdot r_i - (\pi' + \tau')(t_i) \cdot r_{m+i}) \geqslant r_0 + \sum_{i=1}^{m} (\pi(t_i) \cdot r_i - (\pi + \tau)(t_i) \cdot r_{m+i})$ für alle $\mathbf{r} = (r_0, \ldots, r_{2m}) \in \mathbb{N}^{2m+1}$ folgern. Damit ist die (BPOT)-Ungleichungen von (π', τ') redundant.
- Die Relation $\ll$ ist per Definition irreflexiv. Die Transitivität folgt aus der Transitivität der $\leqslant$-Relation für Multimengen, wobei die Ungleichheitsbedingung von $\ll$ zusätzlich zu beachten ist.
- Für $(\pi', \tau') \in \{bpo'\}^{\Pi}$ gibt es einen Schnitt $C' \subseteq V'$ und ein eindeutiges Präfix $bpo'' = (V'', <'', l'')$ von C', so dass $\tau' = C'_{l'}$, und $\pi' = V''_{l''}$. Die Menge C' ist dann eine Co-Menge von bpo und V'' definiert ein Präfix dieser Co-Menge von bpo. Wir definieren die Menge $C_1 = \{v \in V'' \mid (\nexists v' \in V'' : v < v') \wedge (v \operatorname{co}_< C')\}$ ($\operatorname{co}_<$ ist die Relation co für die Relation $<$). Die Menge $\tilde{C} = C' \cup C_1$ ist eine Co-Menge von bpo. Weiter definieren wir $C_2 = \{v \in V \setminus V'' \mid (\nexists v' \in V \setminus V'' : v' < v) \wedge (v \operatorname{co}_< \tilde{C})\}$. Die Menge $C = \tilde{C} \cup C_2$ ist ein Schnitt von bpo. Die Menge $V'' \setminus C_1$ definiert das Präfix

des Schnittes C. Also gilt für $\tau = C_l$ und $\pi = (V'' \setminus C_1)_l$, dass $(\pi, \tau) \in \{\text{bpo}\}^\Pi$. Es gilt $V'' \setminus C_1 \subseteq V''$ und $C' \cup V'' \subseteq (C' \cup C_1 \cup C_2) \cup (V'' \setminus C_1) = C \cup (V'' \setminus C_1)$. Damit gilt $\pi \leqslant \pi'$ und $\tau' + \pi' \leqslant \tau + \pi$. Es folgt $(\pi', \tau') \ll (\pi, \tau)$ oder $(\pi', \tau') = (\pi, \tau)$ und damit die Behauptung.

□

Auch wenn eine Umkehrung der ersten Aussage des Lemmas nicht gilt, d.h. selbst bei der Betrachtung von nur zwei (BPOT)-Ungleichungen gibt es noch Fälle von redundanten Ungleichungen, welche nicht mit der $\ll$-Relation gefunden werden können, so sind durch das Kriterium $(\pi', \tau') \ll (\pi, \tau)$ dennoch die wichtigsten Fälle von redundanten (BPOT)-Ungleichungen abgedeckt. Das Kriterium erlaubt es insbesondere, redundante (BPOT)-Ungleichungen auf der Ebene der Schnittfortsetzungen zu identifizieren.
Die Redundanz von (BPOT)-Ungleichungen entsprechend der ersten Aussage des Lemmas ist auch der eigentliche Grund dafür, dass wir schon bisher nur die Menge der Schnittfortsetzungen $\mathcal{L}^\Pi$ anstelle der Menge aller richtigen Fortsetzungen $\mathcal{L}^\Pi_{co}$ in der Definition von BPO-Transitions-Regionen betrachten konnten. Eine richtige Fortsetzung, die keiner Schnittfortsetzung entspricht, ist nämlich niemals maximal bzgl. $\ll$, d.h. die Menge der maximalen Elemente von $\mathcal{L}^\Pi_{co}$ bzgl. $\ll$ entspricht der Menge der maximalen Elemente von $\mathcal{L}^\Pi$ bzgl. $\ll$. Als natürliche Fortführung dieser Ideen stellt sich nun die Betrachtung der Menge der maximalen Schnittfortsetzungen $\mathcal{L}^\Pi_{max} = \{(\pi, \tau) \in \mathcal{L}^\Pi | \forall (\pi', \tau') \in \mathcal{L}^\Pi : (\pi, \tau) \not\ll (\pi', \tau')\} \subseteq \mathcal{L}^\Pi$ zur linear algebraischen Charakterisierung von BPO-Transitions-Regionen dar. Entsprechend der ersten Aussage des Lemmas ergibt sich die Menge der BPO-Transitions-Regionen auch als Menge der nicht-negativen ganzzahligen Lösungen des Ungleichungssystems, welches nur die (BPOT)-Ungleichungen für die maximalen Schnittfortsetzungen $\mathcal{L}^\Pi_{max}$ beinhaltet.
Die Bedeutung der zweiten Aussage des Lemmas stellt sich in diesem Kontext folgendermaßen dar. Die Argumentation im letzten Abschnitt setzt die Existenz einer geeigneten Menge bzgl. $\ll$ maximaler Schnittfortsetzungen bzw. maximaler richtiger Fortsetzungen voraus. Diese Existenz ist durch die Ordnungseigenschaft von $\ll$ in der zweiten Aussage des Lemmas gewährleistet.
Die ursprüngliche Motivation der Betrachtung von $\ll$ war es, redundante Ungleichungen, welche im Rahmen der Schritt-Transitions-Regionen aufgrund von „Präfixschrittfolgen“ und „Sequentialisierungsschrittfolgen“ auftraten, zu vermeiden. Entsprechend Korollar 4.3.1 (ähnliche Argumentation wie in Bemerkung 4.3.1) erzeugen auch im Falle von BPO-Transitions-Regionen Präfixe und Sequentialisierungen von BPOs redundante Ungleichungen. Die dritte Aussage des letzten Lemmas zeigt, dass diese redundanten Ungleichungen durch die Betrachtung maximaler Schnittfortsetzungen vollständig vermieden werden. Dabei ist aber wichtig, dass durch diese Herangehensweise auch noch viele weitere redundante Ungleichungen zusätzlich vermieden werden. In folgender Bemerkung sind nun die angekündigten Optimierungsvorschläge zur Reduktion der Anzahl der Zeilen von $\mathbf{A}^{BPOT}_{\mathcal{L}}$ beschrieben.

Bemerkung 4.3.2 (Optimierungsvorschläge)

BEMERKUNG 4.3.2 (OPTIMIERUNGSVORSCHLÄGE)
Die folgenden Optimierungen des Begriffs der BPO-Transitions-Regionen aus Definition 4.3.10 und der entsprechenden linear algebraischen Charakterisierung aus Lemma 4.3.8 sind möglich.

- *Wiederum ist es entsprechend Korollar 4.3.1 möglich, nur die BPOs aus der Repräsentationsmenge* $\mathrm{Rep}(\mathcal{L})$ *anstelle aller BPOs aus* $\mathcal{L}$ *zu betrachten. Hierdurch wird die Anzahl der Zeilen von* $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ *reduziert, da* $\mathrm{Rep}(\mathcal{L})^{\Pi} \subseteq \mathcal{L}^{\Pi}$. *Auch der Aufwand zur Berechnung der Menge der Schnittfortsetzungen ausgehend von der kleineren Menge* $\mathrm{Rep}(\mathcal{L})$ *ist geringer. Demgegenüber steht allerdings wiederum der zusätzliche Aufwand zur Berechnung von* $\mathrm{Rep}(\mathcal{L})$ *aus* $\mathcal{L}$. *Außerdem gilt auch wieder, dass* $\mathrm{Rep}(\mathcal{L})$ *aus Anwendungssicht typischerweise nicht sehr viel kleiner ist als* $\mathcal{L}$.

- *In der dritten Aussage aus Lemma 4.3.9 wurde gezeigt, dass der erste Optimierungsvorschlag dem Weglassen bestimmter bzgl.* $\ll$ *nicht maximaler Schnittfortsetzungen entspricht. Unabhängig von dieser Optimierung ist es vielversprechend, generell nur die bzgl.* $\ll$ *maximalen Schnittfortsetzungen* $\mathcal{L}_{\mathrm{max}}^{\Pi}$ *zu betrachten. Dies bedeutet, dass* $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ *nur aus den Zeilenvektoren* $\mathbf{a}_{(\pi,\tau)}$ *für alle Schnittfortsetzungen* (π,τ) *der Menge* $\mathcal{L}_{\mathrm{max}}^{\Pi}$ *anstelle der größeren Menge* $\mathcal{L}^{\Pi}$ *besteht. Entsprechend den ersten zwei Aussagen aus Lemma 4.3.9 ist die Betrachtung dieser Zeilenvektoren ausreichend. Die Berechnung der kleineren Menge* $\mathcal{L}_{\mathrm{max}}^{\Pi}$ *an Schnittfortsetzungen erfordert allerdings auch hier zusätzlichen Aufwand. Es können wiederum anstatt* $\mathcal{L}_{\mathrm{max}}^{\Pi}$ *auch Mengen* Π *an Schnittfortsetzungen mit* $\mathcal{L}_{\mathrm{max}}^{\Pi} \subseteq \Pi \subseteq \mathcal{L}^{\Pi}$ *verwendet werden. Dies ermöglicht das Anwenden von Heuristiken, welche darauf abzielen, eine möglichst kleine, aber nicht notwendigerweise minimale, solche Menge* Π *zu erzeugen.*

- *Entsprechend Lemma 4.3.9 bedeuten die ersten zwei Optimierungsvorschläge aus linear algebraischer Sicht das Weglassen bestimmter redundanter Ungleichungen des Ungleichungssystems* $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$. *Im Allgemeinen lassen sich mit diesen Optimierungsvorschlägen aber nicht alle redundanten Ungleichungen des Ungleichungssystems entfernen. Deswegen ist es unabhängig von obigen Optimierungsvorschlägen sinnvoll, nach redundanten Ungleichungen des Ungleichungssystem zu suchen und die entsprechenden Zeilen der Matrix* $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}}$ *zu entfernen. Natürlich bleibt dadurch eine korrekte linear algebraische Charakterisierung der Menge der BPO-Transitions-Regionen erhalten. Generelle Verfahren zum Auffinden redundanter Ungleichungen sind in Abschnitt 3.4 diskutiert. Es sind auch über den zweiten Optimierungsvorschlag hinausgehende Verfahren auf der Ebene der Schnittfortsetzungen denkbar, um redundante Ungleichungen zu entfernen.*

Beispiel

BEISPIEL: *Für die partielle Sprache aus Abbildung 4 gilt* $\mathrm{Rep}(\mathcal{L}) = \mathcal{L}$, *so dass der erste Optimierungsvorschlag keine Verbesserung erbringt. Außerdem gilt* $\mathcal{L}_{\mathrm{max}}^{\Pi} = \mathcal{L}^{\Pi}$, *d.h. jede Schnittfortsetzung ist in diesem Beispiel maximal. Somit ergibt sich auch mit dem zweiten Optimierungsvorschlag keine Reduktion der Zeilen der Matrix aus Abbildung 63.*
Anders liegt die Situation im Falle der partiellen Sprache aus Abbildung 64. Zuerst einmal kann entsprechend des ersten Optimierungsvorschlages die erste BPO weggelassen werden, da sie eine Sequentialisierung der zweiten BPO ist. Allerdings können entsprechend den bisherigen Überlegungen deren Schnittfortsetzung auch mithilfe des zweiten Optimierungsvorschlages weggelassen werden. Die Schnittfortsetzungen der ersten BPO sind $(0, a)$, (a, c) *und* $(a + c, b)$, *die Schnittfortsetzungen der zweiten BPO sind* $(0, a)$ *und* $(a, b + c)$ *und die Schnittfortsetzungen der dritten BPO sind* $(0, a + b)$ *und* $(a + b, c)$. *Damit kommt die erste Schnittfortsetzung der ersten BPO auch*

als Schnittfortsetzung der zweiten BPO vor und die weiteren zwei Schnittfortsetzungen sind nicht maximal bzgl. $\ll$, da sowohl $(a, c) \ll (a, b + c)$ als auch $(a + c, b) \ll (a, b + c)$ gilt. Alle drei Schnittfortsetzungen werden also tatsächlich bei Verwendung des zweiten Optimierungsvorschlages vernachlässigt. Mit dem zweiten Optimierungsvorschlag können aber auch in anderen Situationen redundante Ungleichungen vermieden werden. So ist auch die erste Schnittfortsetzungen der zweiten BPO und die zweite Schnittfortsetzung der dritten BPO nicht maximal, da $(0, a) \ll (0, a + b)$ und $(a + b, c) \ll (a, b + c)$ gilt. Obwohl die betrachtete partielle Sprache also 7 Schnitte und 6 verschiedene Schnittfortsetzungen beinhaltet, so sind nur die zwei Schnittfortsetzungen $(a, b + c)$ und $(0, a + b)$ maximal bzgl. $\ll$. Bei Verwendung des zweiten Optimierungsansatzes entsteht somit ein Ungleichungssystem mit nur zwei Ungleichungen.

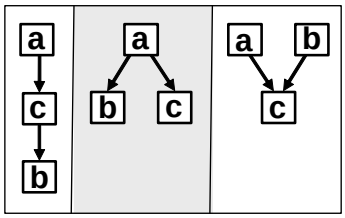

Abbildung 64: Eine partielle Sprache.

Insgesamt lässt sich das Konzept der BPO-Transitions-Regionen als eine grundlegende Weiterentwicklung des Konzeptes der Schritt-Transitions-Regionen verstehen. Zum einen wird die aufwändige Berechnung aller Schrittlinearisierungen der BPOs einer partiellen Sprache vermieden. Zum anderen haben die Ausführungen zu Beginn des Unterabschnittes gezeigt, dass in den linearen Ungleichungssystemen im Falle von BPO-Transitions-Regionen etliche redundante Ungleichungen, welche im Falle von Schritt-Transitions-Regionen auftreten, vermieden werden. Es lässt sich auch formal nachweisen, dass ein Ungleichungssystem $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}} \cdot \mathbf{x} \geqslant \mathbf{o}$ einerseits keine identischen Ungleichungen und andererseits nur Ungleichungen, welche auch in $\mathbf{A}_{\mathcal{L}}^{\mathrm{ST}} \cdot \mathbf{x} \geqslant \mathbf{o}$ vorkommen, enthält.

Lemma 4.3.10

LEMMA 4.3.10
Sei $\mathcal{L}$ eine partielle Sprache mit Beschriftungsmenge $T = \{t_1, \ldots, t_m\}$.

- *Für $(\pi, \tau), (\pi', \tau') \in \mathcal{L}^{\Pi}$ mit $(\pi, \tau) \neq (\pi', \tau')$ gilt $\mathbf{a}_{(\pi,\tau)} \neq \mathbf{a}_{(\pi',\tau')}$.*
- *Für $(\pi, \tau) \in \mathcal{L}^{\Pi}$ existiert $\sigma = \tau_1 \ldots \tau_n \in \{\sigma(\mathrm{bpo}) \mid \mathrm{bpo} \in \mathrm{Slin}(\mathcal{L})\}$ und $j \in \{1, \ldots, n\}$ mit $\mathbf{a}_{\sigma}^{j} = \mathbf{a}_{(\pi,\tau)}$.*

BEWEIS: Die zwei Aussagen des Lemmas werden nacheinander bewiesen.

- Die erste Möglichkeit ist, dass ein t_i existiert mit $\pi(t_i) \neq \pi'(t_i)$. Dann gilt $a_{(\pi,\tau),i} = \pi(t_i) \neq \pi'(t_i) = a_{(\pi',\tau'),i}$ und damit $\mathbf{a}_{(\pi,\tau)} \neq \mathbf{a}_{(\pi',\tau')}$. Die zweite Möglichkeit ist, dass $\pi = \pi'$ erfüllt ist aber ein t_i existiert mit $\tau(t_i) \neq \tau'(t_i)$. Dann gilt $a_{(\pi,\tau),i+m} = -(\pi + \tau)(t_i) = -\pi(t_i) - \tau(t_i) = -\pi'(t_i) - \tau(t_i) \neq -\pi'(t_i) - \tau'(t_i) = -(\pi' + \tau')(t_i) = a_{(\pi',\tau'),i+m}$ und damit $\mathbf{a}_{(\pi,\tau)} \neq \mathbf{a}_{(\pi',\tau')}$.
- Es existiert eine BPO $\mathrm{bpo} = (V, <, l) \in \mathcal{L}$, ein Schnitt $C \subseteq V$ von bpo sowie ein eindeutiges Präfix $\mathrm{bpo}' = (V', <', l')$ von C, so dass

$\tau = C_l$ und $\pi = V'_{l'}$. Außerdem betrachten wir für $V'' = V \setminus (V' \cup C)$, $<'' = <_{V'' \times V''}$ und $l'' = l|_{V''}$ die BPO $bpo'' = (V'', <'', l'')$. Sei $bpo_1 = (V', <_1, l') \in Slin(bpo')$ und $bpo_2 = (V'', <_2, l'') \in Slin(bpo'')$. Wir konstruieren weiter die BPO $bpo_s = (V, <_1 \cup <_2 \cup (V' \times C) \cup (V' \times V'') \cup (C \times V''), l)$. Es lässt sich nachprüfen, dass $bpo_s \in Slin(bpo) \subseteq Slin(\mathcal{L})$. Wir wählen j um eins größer als die Anzahl der Schritte der Schrittfolge $\sigma(bpo_1)$. Dann gilt für $\sigma = \sigma(bpo_s)$ und j die Beziehung $\mathbf{a}^j_\sigma = \mathbf{a}_{(\pi,\tau)}$.

□

Trotz all der genannten positiven Aspekte von BPO-Transitions-Regionen bleibt im schlechtesten Fall auch bei der Verwendung aller in Bemerkung 4.3.2 vorgestellten Optimierungen die exponentielle Abhängigkeit der Größe der entsprechenden linearen Ungleichungssysteme von der Größe der betrachteten partiellen Sprache bestehen. Insgesamt sieht es nicht danach aus, dass im Rahmen des Konzeptes der Transitions-Regionen eine polynomielle Abhängigkeit der Größe der linearen Ungleichungssysteme von der Größe der Eingabe realisierbar ist. Daher wird im nächsten Unterabschnitt eine sich von den bisherigen Definitionen konzeptuell deutlich unterscheidende Regionendefinition für partielle Sprachen vorgeschlagen.

4.3.3 *Markenfluss-Regionen*

In diesem Unterabschnitt führen wir sog. Markenfluss-Regionen für partielle Sprachen ein. Während bei den zwei Typen von Transitions-Regionen der letzten beiden Unterabschnitte direkt die Kantengewichte und die Anfangsmarkierung von zu synthetisierenden Stellen kodiert werden, werden bei Markenfluss-Regionen Flüsse von Marken entlang der Kanten von BPOs betrachtet. Ein Markenfluss zwischen zwei Ereignissen stellt dar, dass eine bestimmte Anzahl von Marken in der zu synthetisierenden Stelle von dem ersten Ereignis produziert und dann von dem zweiten Ereignis konsumiert wird.
Ein solcher Zusammenhang wurde in Prozessnetzen durch entsprechende Bedingungen repräsentiert. Daher gibt es einen sehr engen Zusammenhang zwischen Markenflüssen und Prozessnetzen. Markenflüsse abstrahieren von der Individualität von Bedingungen eines Prozessnetzes, indem sie die Flussrelation des Prozessnetzes durch nicht-negative ganze Zahlen kodieren. Hierzu wird die Prozess-BPO des Prozessnetzes betrachtet. Jeder Kante zwischen zwei Ereignissen der Prozess-BPO wird eine Zahl für jede Stelle des zugrunde liegenden Petrinetzes zugeordnet. Eine solche Zahl repräsentiert die Anzahl entsprechend beschrifteter Bedingungen, welche im Prozessnetz im Nachbereich des ersten Ereignisses und zugleich im Vorbereich des zweiten Ereignisses liegen. Mit anderen Worten wird also die Anzahl der Marken, welche vom ersten Ereignis in der Stelle produziert und dann vom zweiten Ereignis konsumiert wird, angegeben. Die minimalen und maximalen Bedingungen des Prozessnetzes können auf diese Weise nicht berücksichtigt werden, da sie kein Ereignis im Vorbereich oder im Nachbereich haben. Daher wird die Prozess-BPO noch um ein künstliches minimales und maximales Ereignis erweitert, dem sog. Quellenknoten und dem sog. Senkenknoten. Der Quellenknoten wird als Ereignis, welches die Marken der Anfangsmarkierung des Petrinetzes produziert, und der Senkenknoten als Ereignis, welches die

Endmarkierung des Petrinetzes nach dem Schalten der BPO konsumiert, interpretiert. Damit lassen sich dann auch die minimalen und maximalen Bedingungen durch entsprechende Markenflüsse darstellen. Insgesamt repräsentieren Markenflüsse also die Bedingungen eines Prozessnetzes auf der Ebene von BPOs. Mit dieser Idee lässt sich eine kompaktere Repräsentation des von Prozessnetzen modellierten Verhaltens in der Form sog. Markenfluss-Prozesse gewinnen.

Beispiel

BEISPIEL: *Abbildung 65 zeigt den Markenfluss-Prozess, welcher dem zweiten Prozessnetz aus Abbildung 48 entspricht. Der i-te Eintrag der an den Kanten der dargestellten BPO annotieren Viertupel stellt den Markenfluss bzgl. der Stelle p_i dar.*

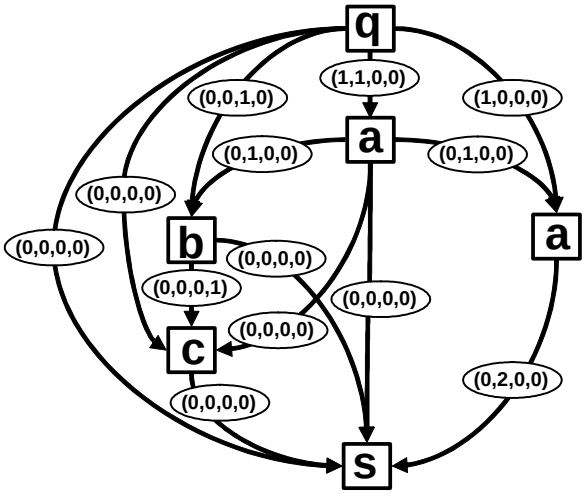

Abbildung 65: Zu dem zweiten Prozessnetz aus Abbildung 48 korrespondierender Markenfluss-Prozess.

Im folgenden wird zuerst der Begriff des Markenfluss-Prozesses formal hergeleitet und dann die Möglichkeit der Repräsentation des von Prozessnetzen modellierten Verhaltens durch Markenfluss-Prozesse im Detail beschrieben. Darauf aufbauend wird eine geeignete Definition von Markenfluss-Regionen für partielle Sprachen entwickelt. Hierzu wird die Charakterisierung der Aktiviertheit von BPOs durch Prozessnetze aus Satz 3.3.1 wesentlich verwendet. Damit basiert jede der drei bisher vorgestellten Regionendefinitionen auf einer anderen der drei Charakterisierungen der Aktiviertheit von BPOs aus Unterabschnitt 3.3.2.
Es bleibt noch zu bemerken, dass das Konzept der Markenflüsse und die Idee, basierend auf Markenflüssen einen Regionenbegriff herzuleiten, schon in [131, 133, 132, 153] präsentiert wurden. Allerdings werden in der hier vorliegenden Arbeit ein klarer strukturierter Aufbau und intuitivere und einfachere Formalismen (ähnlich zu [134]) entwickelt und verwendet. Dadurch ergeben sich dann einfachere und kürzere Beweise für die wesentlichen Resultate zu Markenflüssen.
Genauer gesagt ist der wesentliche konzeptuelle Unterschied zu [132, 153], dass wir hier die grundsätzlichen Ideen zur Synthese von S/T-Netzen mit Hilfe von Regionen schon zu Beginn dieses Abschnittes noch unabhängig von dem konkreten technischen Begriff der Markenfluss-Region eingeführt haben. Ausgehend von diesen Ausführungen

lassen sich dann in natürlicherweise die verschiedenen Regionendefinitionen, also auch der Begriff der Markenfluss-Region, herleiten. Diese klare Trennung der grundsätzlichen Syntheseideen von den technischen Regionendefinitionen führt zu einer einfacheren Struktur der Definitionen, einer besseren Verständlichkeit und einfacheren Techniken und Beweisen.
Die in diesem Unterabschnitt verwendeten Formalismen unterscheiden sich dementsprechend in einigen Punkten stark von den entsprechenden Techniken in [153]. Hervorzuheben ist hierbei der neue Begriff des Markenfluss-Prozesses, durch den eine eigenständige, neue Möglichkeit zur Verhaltensbeschreibung von S/T-Netzen eingeführt wird. Markenfluss-Prozesse sind daher auch losgelöst von dem in dieser Arbeit betrachteten Syntheseproblem von Interesse. Dies wird als kleiner Exkurs in folgender Bemerkung diskutiert (ausführlich ist dies in den Arbeiten [34, 134] nachzulesen).

Bemerkung 4.3.3 (Markenfluss-Prozess)

BEMERKUNG 4.3.3 (MARKENFLUSS-PROZESS)
Bisher wurden in Unterabschnitt 3.3.2 die Konzepte der aktivierten BPO und des Prozessnetzes zur Repräsentation von halbgeordneten Abläufen von S/T-Netzen vorgestellt. Markenfluss-Prozesse stellen hierzu ein drittes sich von den anderen beiden wesentlich unterscheidendes Konzept dar. Somit definieren Markenfluss-Prozesse eine neue Semantik für S/T-Netze.
Im Gegensatz zu Prozessnetzen wird bei Markenfluss-Prozessen die Individualität von Marken mit derselben „Geschichte" aufgehoben. Das bedeutet, dass die durch das Schalten einer Transition in einer Stelle produzierten Marken nicht mehr durch individuelle Bedingungen unterschieden werden. Stattdessen wird nur unterschieden, wie viele der produzierten Marken von welchem nachfolgenden Schaltereignis konsumiert werden. Zum einen ermöglicht dies für einzelne halbgeordneter Abläufe eine kompaktere Darstellung mit einer einfacheren Struktur. Zum anderen erlaubt die einfachere Struktur von Markenfluss-Prozessen auch neue effiziente Möglichkeiten zur kompakten Zusammenfassung mehrerer bzw. aller Abläufe eines S/T-Netzes in einer Art Verzweigungsprozess. Hier bieten sich enorme Einsparungspotentiale sowohl in der Rechenzeit zur Konstruktion von entsprechenden Verzweigungsprozessen als auch in der Größe der resultierenden Verzweigungsprozesse. Dies ist insbesondere im Hinblick auf Model-Checking-Verfahren, welche auf Verzweigungsprozessen basieren, aber auch generell im Rahmen von Verifikations- und Validierungsansätzen für Petrinetze interessant.
Bei aktivierten BPOs wird im Vergleich zu Markenfluss-Prozessen der zu einem Ablauf eines S/T-Netzes gehörige Fluss der Marken im Netz vollständig ausgeblendet. Dadurch lassen sich die wahren kausalen Abhängigkeiten zwischen den Ereignissen, welche tatsächlich durch einen entsprechenden Fluss von Marken begründet sind, nicht repräsentieren. Zu einer aktivierten BPO gibt es typischerweise unterschiedliche mögliche Verteilungen des Marken-Abflusses eines Ereignisses auf Marken-Zuflüsse folgender Ereignisse. Ein Markenfluss-Prozess basiert auf genau einer solchen Verteilung. Dadurch werden in Markenfluss-Prozessen exakte Informationen über die kausalen Abhängigkeiten der Ereignisse dargestellt und der Fluss von entsprechenden Ressourcen modelliert. Außerdem ergeben sich durch die Flussinformationen auch explizit Zustandsinformationen. Aktivierte BPOs enthalten keine Zustandsinformationen und eignen sich daher auch sehr schlecht zur Zusammenfassung in entsprechenden Verzweigungsprozessen.

Wir beginnen die technischen Ausführungen zur Herleitung des neuen Begriffs des Markenfluss-Prozesses mit der Idee der Markenfluss-Funktion. Eine solche gibt für jede Kante einer BPO an, wie viele

Marken in einer bestimmten Stelle von dem ersten Ereignis produziert und von dem zweiten Ereignis konsumiert werden. Diese Anzahl an Marken kann natürlich auch null sein. Die Summe der Marken, die von einem Ereignis produziert bzw. konsumiert wird heißt Marken-Abfluss bzw. Marken-Zufluss des Ereignisses.

Definition 4.3.11 (Markenfluss-Funktion)

Definition 4.3.11 (Markenfluss-Funktion)
Sei $\mathrm{bpo} = (V, <, l)$ *eine BPO. Eine Funktion* $x :<\to \mathbb{N}$ *heißt Markenfluss-Funktion (auf* bpo*). Für einen Knoten* $v \in V$ *bezeichnen wir* $\mathrm{Zu}_x(v) = \sum_{v'<v} x(v', v)$ *als Marken-Zufluss von* v *und* $\mathrm{Ab}_x(v) = \sum_{v<v'} x(v, v')$ *als Marken-Abfluss von* v *(jeweils bzgl.* x*).*

Wir betrachten Markenfluss-Funktionen auf sog. $*$-BPOs mit einem eindeutigen minimalen Knoten, dem Quellenknoten, und einem eindeutigen maximalen Knoten, dem Senkenknoten. Eine $*$-BPO lässt sich aus jeder BPO durch Hinzufügen eines Quellenknotens und eines Senkenknotens gewinnen.

Definition 4.3.12 ($*$-BPO)

Definition 4.3.12 ($*$-BPO)
Eine BPO $\mathrm{bpo} = (V, <, l)$ *mit einem eindeutigen minimalen Knoten* $q_{\mathrm{bpo}} \in V$ *mit* $l(q_{\mathrm{bpo}}) = q$ *und einem eindeutigen maximalen Knoten* $s_{\mathrm{bpo}} \in V$ *mit* $l(s_{\mathrm{bpo}}) = s$ *heißt $*$-BPO. Sei* bpo $= (V, <, l)$ *eine BPO, dann bezeichnen wir die $*$-BPO* $\mathrm{bpo}^* = (V^*, <^*, l^*)$ *mit* $V = V^* \setminus \{q_{\mathrm{bpo}^*}, s_{\mathrm{bpo}^*}\}$, $<=<^* |_{V\times V}$ *und* $l = l^*|_V$ *als $*$-Erweiterung der BPO* bpo*. Umgekehrt bezeichnen wir die BPO* bpo *als $*$-Einschränkung der $*$-BPO* bpo^**.*

Beispiel

Beispiel: *Abbildung 66 zeigt die $*$-Erweiterung* bpo_2^* *von* bpo_2 *aus Abbildung 4 versehen mit einer Markenfluss-Funktion. Das* q*-Ereignis hat einen Marken-Abfluss von* 1*. Sowohl das erste als auch das zweite* a*-Ereignis haben einen Marken-Zufluss von* 1 *und einen Marken-Abfluss von* 2*. Das* b*-Ereignis hat einen Marken-Zufluss von* 1 *und einen Marken-Abfluss von* 0*. Das* c*-Ereignis hat einen Marken-Zufluss und einen Marken-Abfluss von* 0*. Das* s*-Ereignis hat einen Marken-Zufluss von* 2*.*

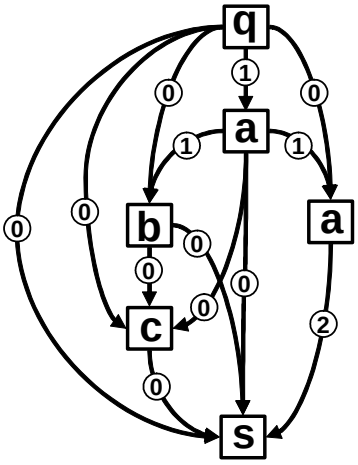

Abbildung 66: Markenfluss-Funktion auf der $*$-Erweiterung von bpo_2 aus Abbildung 4.

Ähnlich wie Kausalnetze die Grundlage zur Definition von Prozessnetzen bilden, stellen sog. partiell geordnete Markenflüsse die Grundlage zur Definition von halbgeordneten Abläufen von S/T-Netzen in der

Form von Markenfluss-Prozessen dar. Ein partiell geordneter Markenfluss setzt sich aus einer $*$-BPO sowie einer Menge von Markenfluss-Funktionen auf der $*$-BPO zusammen.

DEFINITION 4.3.13 (PARTIELL GEORDNETER MARKENFLUSS)
Ein partiell geordneter Markenfluss ist ein Paar $\mu = (\mathrm{bpo}, X)$, *wobei* $\mathrm{bpo} = (V, <, \iota)$ *eine* $*$*-BPO und* X *eine Menge von Markenfluss-Funktionen auf* bpo *ist. Die* $*$*-Einschränkung* bpo_μ *der* $*$*-BPO* bpo *heißt zu* μ *korrespondierende BPO.*

Definition 4.3.13 (Partiell geordneter Markenfluss)

Um einen Ablauf eines S/T-Netzes durch einen partiell geordneten Markenfluss zu beschreiben, betrachten wir die Ereignisse des partiell geordneten Markenflusses als Schaltereignisse des S/T-Netzes. Der Quellenknoten des partiell geordneten Markenflusses wird als ein spezielles Ereignis zum Produzieren der Anfangsmarkierung des S/T-Netzes interpretiert. Der Senkenknoten wird als ein spezielles Ereignis zum Konsumieren der Folgemarkierung des S/T-Netzes nach der Ausführung aller anderen Ereignisse des partiell geordneter Markenflusses interpretiert. Eine Markenfluss-Funktion des partiell geordneten Markenflusses repräsentiert einen möglichen Fluss von Marken zwischen den Ereignissen des partiell geordneten Markenflusses bzgl. einer bestimmten Stelle des Netzes. Die entsprechenden Marken-Abflüsse bzw. Marken-Zuflüsse der Ereignisse stellen die Anzahlen der von den Ereignissen in der betrachteten Stelle produzierten bzw. konsumierten Marken dar.
Dabei ist zu beachten, dass es für ein und dieselbe Stelle unterschiedliche mögliche Verteilungen des Flusses der Marken geben kann. Dies liegt daran, dass eine Stelle eines S/T-Netzes mehrere Transitionen in ihrem Vorbereich und ihrem Nachbereich haben kann. Damit können Marken, die eine Transition in einer Stelle konsumiert, von verschiedenen Transitionen produziert worden sein (bzw. aus der Anfangsmarkierung stammen). Umgekehrt können auch Marken, die von einer Transition in einer Stelle produziert werden (bzw. aus der Anfangsmarkierung stammen), von verschiedenen Transitionen konsumiert werden. Entsprechend gibt es bzgl. einer Stelle verschiedene Verteilungen des Marken-Abflusses eines Ereignisses auf Marken-Zuflüsse nachfolgender Ereignisse des partiell geordneten Markenflusses. Die Idee ist hier, dass eine Markenfluss-Funktion genau eines der möglichen Flussverhalten bzgl. einer Stelle darstellt.

BEISPIEL: *Wir betrachten den vom dritten Prozessnetz aus Abbildung 48 bzgl. der Stelle* p_2 *definierten Markenfluss auf der* $*$*-Erweiterung der zugehörigen Prozess-BPO. Eine zu diesem Markenfluss alternative Verteilung des Flusses bzgl.* p_2 *auf derselben BPO ist dann durch das zweite Prozessnetz gegeben. Die zwei Markenflüsse sind in diesem Fall zwar bzgl. derselben BPO definiert, die durch Flüsse von Marken gegebenen wahren Abhängigkeiten der Ereignisse unterscheiden sich allerdings. So hängt das* b*-Ereignis nur im ersten Fall tatsächlich von dem zweiten* a*-Ereignis ab.*

Beispiel

Für einen partiell geordneten Markenflusses soll nun gelten, dass er für jede Stelle des S/T-Netzes genau eine Markenfluss-Funktion enthält, welche einen Fluss von Marken bzgl. dieser Stelle repräsentiert. Dabei stellt entsprechend den bisherigen Überlegungen eine Markenfluss-Funktion genau dann einen möglichen Fluss von Marken bzgl. einer Stelle dar,

- wenn der Marken-Zufluss bzw. Marken-Abfluss jedes Schaltereignisses des partiell geordneten Markenflusses der (durch die Gewichtsfunktion des S/T-Netzes gegebenen) Anzahl der beim Schalten der entsprechenden Transition aus der Stelle konsumierten bzw. in der Stelle produzierten Marken entspricht
- und der Marken-Abfluss des Quellenknotens mit der Anfangsmarkierung des S/T-Netzes in der Stelle übereinstimmt.

Insgesamt lässt sich auf diese Weise mit einem partiell geordneten Markenfluss ein mögliches Flussverhalten des gesamten S/T-Netzes repräsentieren. Diese Überlegungen werden in dem Begriff des Markenfluss-Prozesses zusammengefasst.

Definition 4.3.14 (Markenfluss-Prozess)

Definition 4.3.14 (Markenfluss-Prozess)
Ein partiell geordneter Markenfluss $\mu = (bpo, X)$, $bpo = (V, <, l)$, *ist ein Markenfluss-Prozess eines markierten S/T-Netzes* (N, m_0), $N = (P, T, W)$, *falls* $l(V \setminus \{q_{bpo}, s_{bpo}\}) \subseteq T$ *und eine bijektive Funktion* $\phi : X \to P$ *existiert, so dass*

(i) $\forall x \in X : Ab_x(q_{bpo}) = m_0(\phi(x))$.

(ii) $\forall x \in X, \forall v \in V \setminus \{q_{bpo}, s_{bpo}\} : Ab_x(v) = W(l(v), \phi(x)) \wedge Zu_x(v) = W(\phi(x), l(v))$.

Beispiel

Beispiel: *Abbildung 65 illustriert einen partielle geordneten Markenfluss mit vier Markenfluss-Funktionen. Letztere werden durch die Komponenten der an den Kanten des partiell geordneten Markenflusses annotieren Viertupel dargestellt. Es lässt sich nachprüfen, dass der partiell geordnete Markenfluss ein Markenfluss-Prozess des Netzes aus Abbildung 3 ist. Der durch die* i*-te Komponente dargestellte Markenfluss ist dabei der Stelle* p_i *zugeordnet. Beispielsweise haben wir im Rahmen von Abbildung 66 die zu* p_2 *zugeordnete Markenflussfunktion diskutiert. Die berechneten Marken-Zuflüsse und Marken-Abflüsse der Ereignisse bzgl. dieses Markenflusses genügen tatsächlich den Anforderung der letzten Definition. So haben alle* a*-Ereignisse einen Marken-Zufluss von* 1 *und einen Marken-Abfluss von* 2*. Diese Werte entsprechen gerade den Kantengewichten der von* p_2 *zu* a *gerichteten Kante und der umgekehrt gerichteten Kante. Analoge Entsprechungen gelten auch für die* b*- und* c*-Ereignisse. Der Marken-Abfluss des* q*-Ereignisses ist* 1*. Dies ist in Übereinstimmung mit der Anfangsmarkierung von* p_2*.*

Wie in Abschnitt 3.3 ausgeführt, lassen sich halbgeordnete Abläufe in einer natürlichen Weise mit BPOs modellieren. Dadurch, dass wir BPOs hier um Markenflüsse ergänzen, lassen sich die wahren Abhängigkeiten der Ereignisse repräsentieren. Mit einer wahren Abhängigkeit meinen wir eine Abhängigkeit, zu welcher ein entsprechend positiver Fluss von Marken (bzgl. irgendeiner der Markenfluss-Funktionen) korrespondiert.
Wird von den Markenfluss-Funktionen sowie dem künstlichen Quellenknoten und Senkenknoten eines Markenfluss-Prozesses abstrahiert, so entsteht eine einfache BPO, welche nur noch die Reihenfolge der zu Transitionen zugeordneten Schaltereignisse des Markenfluss-Prozesses darstellt. Diese BPO repräsentiert somit den zu einem Markenfluss-Prozess zugehörigen Ablauf. Sie wird als zum Markenfluss-Prozess gehörige Markenfluss-BPO bezeichnet. Da auf der Ebene der Markenfluss-BPOs keine Markenfluss-Informationen mehr existieren, lässt sich

bei Markenfluss-BPOs nicht mehr von wahren Abhängigkeiten zwischen Ereignissen sprechen. Zu einer BPO kann es nämlich verschiedene Markenfluss-Verteilung geben, welche jeweils einen anderen Markenfluss-Prozess und somit andere wahre Abhängigkeiten festlegen.

DEFINITION 4.3.15 (MARKENFLUSS-BPO) *Definition 4.3.15 (Markenfluss-BPO)*
Sei $\mu = (\text{bpo}, X)$ *ein Markenfluss-Prozess eines markierten S/T-Netzes* (N, m_0). *Die zu* μ *korrespondierende BPO* bpo_μ *heißt dann Markenfluss-BPO von* (N, m_0) *(bzgl.* μ*).*

Eine erste Beobachtung zu dem Ablaufbegriff der Markenfluss-BPO ist, dass jede Sequentialisierung und jedes Präfix einer Markenfluss-BPO wieder eine Markenfluss-BPO ist.

LEMMA 4.3.11 *Lemma 4.3.11*
Sei $\mu = (\text{bpo}, X)$, $\text{bpo} = (V, <, l)$, *ein Markenfluss-Prozess eines markierten S/T-Netzes* (N, m_0) *und* bpo_μ *die zugehörige Markenfluss-BPO. Dann ist jede Sequentialisierung und jedes Präfix von* bpo_μ *auch eine Markenfluss-BPO von* (N, m_0).

BEWEIS: Sei $\text{bpo}' = (V, <', l)$ eine Sequentialisierung von bpo. Für eine Markenfluss-Funktion $x \in X$ betrachten wir die Markenfluss-Funktion x', welche x durch 0-Werte auf $<'$ fortsetzt, d.h. $x'|_< = x$ und $x'|_{<' \setminus <} = 0$. Da sich dadurch die Marken-Zuflüsse und die Marken-Abflüsse von allen Knoten V nicht ändern, ist $\mu' = (\text{bpo}', \{x' \mid x \in X\})$ ein Markenfluss-Prozess. Offensichtlich ist jede Sequentialisierung von bpo_μ eine zu einem solchen Markenfluss-Prozess gehörige Markenfluss-BPO.
Sei V'' eine abgeschlossene Knotenmenge von bpo_μ. Wir betrachten die BPO $\text{bpo}' = (V', <', l')$ mit $V' = V'' \cup \{q_{\text{bpo}}, s_{\text{bpo}}\} \subseteq V$, $<' = <|_{V' \times V'}$ und $l' = l|_{V'}$. Für eine Markenfluss-Funktion $x \in X$ betrachten wir die Markenfluss-Funktion x' auf $<'$, welche die x-Flüsse auf Kanten von Knoten V' zu Knoten $V \setminus V'$ den von den ersteren Knoten zu s_{bpo} führenden Kanten zuordnet und ansonsten mit x übereinstimmt, d.h. für $<'' = <|_{(V' \setminus \{s_{\text{bpo}}\}) \times (V' \setminus \{s_{\text{bpo}}\})}$ gilt $x'|_{<''} = x|_{<''}$ und für $v \in V' \setminus \{s_{\text{bpo}}\}$ gilt $x'(v, s_{\text{bpo}}) = x(v, s_{\text{bpo}}) + \sum_{v' \in V \setminus V', v <' v'} x(v, v')$. Da sich dadurch die Marken-Zuflüsse und die Marken-Abflüsse von allen Knoten $V' \setminus \{s_{\text{bpo}}\} \subseteq V$ nicht ändern, ist $\mu' = (\text{bpo}', \{x' \mid x \in X\})$ ein Markenfluss-Prozess. Offensichtlich ist jedes Präfix von bpo_μ eine zu einem solchen Markenfluss-Prozess gehörige Markenfluss-BPO. □

BEISPIEL: *Die Markenfluss-BPO des Markenfluss-Prozesses aus Abbildung 65 ist die BPO* bpo_2 *aus Abbildung 4.* *Beispiel*

Über den Begriff des Markenfluss-Prozesses haben wir eine neue Möglichkeit zur Beschreibung halbgeordneten Schaltverhaltens von S/T-Netzen eingeführt. Die schon angekündigte und motivierte Konsistenz dieses Begriffes zum Begriff des Prozessnetzes ergibt sich formal wie folgt. Jeder Markenfluss-Prozess $\mu = (\text{bpo}, X)$, $\text{bpo} = (V, <, l)$, eines markierten S/T-Netzes (N, m_0), $N = (P, T, W)$, definiert ein zugehöriges Prozessnetz $K_\mu = (O, \rho)$, $O = (B, E, G)$, von (N, m_0). Dieses hat die Eigenschaft, dass die zu μ gehörige Markenfluss-BPO die zu K_μ gehörige Prozess-BPO sequentialisiert. Die Ereignisse von K_μ und deren Beschriftungen werden direkt von bpo durch $E = V \setminus \{q_{\text{bpo}}, s_{\text{bpo}}\}$ und $\rho|_E = l|_E$ übernommen. Die Bedingungen und deren Verbindungskanten zu Ereignissen werden entsprechend der Markenfluss-Funktionen

X konstruiert. Die Mengen B und G sowie die Beschriftungen $\rho|_B$ sind durch folgende Eigenschaften festgelegt (die Benennung der Bedingungen spielt keine Rolle):

(i) $|\{b \in B | \rho(b) = p, b \in e^\bullet \cap {}^\bullet e'\}| = \phi^{-1}(p)(e, e')$ für $p \in P$ und $e, e' \in E$ mit $e < e'$,

(ii) $|\{b \in B | \rho(b) = p, b \in e^\bullet \cap {}^\bullet e'\}| = 0$ für $p \in P$ und $e, e' \in E$ mit $e \not< e'$,

(iii) $|\{b \in B | \rho(b) = p, b \in {}^\bullet e, {}^\bullet b = \emptyset\}| = \phi^{-1}(p)(q_{bpo}, e)$ für $p \in P$ und $e \in E$,

(iv) $|\{b \in B | \rho(b) = p, b \in e^\bullet, b^\bullet = \emptyset\}| = \phi^{-1}(p)(e, s_{bpo})$ für $p \in P$ und $e \in E$,

(v) $|\{b \in B | \rho(b) = p, {}^\bullet b = \emptyset, b^\bullet = \emptyset\}| = \phi^{-1}(p)(q_{bpo}, s_{bpo})$ für $p \in P$.

Lemma 4.3.12

LEMMA 4.3.12
Sei $\mu = (bpo, X)$, $bpo = (V, <, l)$, *ein Markenfluss-Prozess eines markierten S/T-Netzes* (N, m_0), $N = (P, T, W)$, *dann ist* $K_\mu = (O, \rho)$, $O = (B, E, G)$, *ein Prozessnetz von* (N, m_0). *Dabei gilt, dass die Markenfluss-BPO* bpo_μ *eine Sequentialisierung der Prozess-BPO* bpo_{K_μ} *ist.*

BEWEIS: Wir prüfen alle Eigenschaften eines Prozessnetzes:

- Bei (i) bis (v) wird schon vorausgesetzt, dass die Bedingungen unverzweigt sind.
- Mit (i) und (ii) und der Transitivität von $<$ lässt sich folgern, dass $G^+|_{E\times E} \subseteq <|_{E\times E}$. Da bpo azyklisch ist, ist somit auch O azyklisch.
- Die Definition von ρ erfüllt offensichtlich $\rho(B) \subseteq P$. Die Eigenschaft $\rho(E) \subseteq T$ ergibt sich aus der Eigenschaft $l(V \setminus \{q_{bpo}, s_{bpo}\}) \subseteq T$ von μ.
- Mit (i), (ii) und (iii) bzw. (i), (ii) und (iv) zusammen mit $\rho|_E = l|_E$ gilt für $e \in E$ und $p \in P$ die Beziehung $|\{b \in {}^\bullet e \mid \rho(b) = p\}| = \sum_{e' \in E, e' < e} \phi^{-1}(p)(e', e) + \phi^{-1}(p)(q_{bpo}, e) = Zu_{\phi^{-1}(p)}(e) = W(p, \rho(e))$ bzw. $|\{b \in e^\bullet \mid \rho(b) = p\}| = \sum_{e' \in E, e < e'} \phi^{-1}(p)(e, e') + \phi^{-1}(p)(e, s_{bpo}) = Ab_{\phi^{-1}(p)}(e) = W(\rho(e), p)$.
- Mit (iii) und (v) gilt für $p \in P$ die Beziehung $|\{b \in Min(O) \mid \rho(b) = p\}| = \sum_{e \in E} \phi^{-1}(p)(q_{bpo}, e) + \phi^{-1}(p)(q_{bpo}, s_{bpo}) = Ab_{\phi^{-1}(p)}(q_{bpo}) = m_0(p)$.

Der Zusatz $bpo_\mu \in Seq(bpo_{K_\mu})$ ergibt sich aus $E = V \setminus \{q_{bpo}, s_{bpo}\}$, $\rho|_E = l|_E$ und der schon im zweiten Punkt der Aufzählung betrachteten Beziehung $G^+|_{E\times E} \subseteq <|_{E\times E}$. □

Umgekehrt definiert jedes Prozessnetz $K = (O, \rho)$, $O = (B, E, G)$, eines markierten S/T-Netzes (N, m_0), $N = (P, T, W)$, auch einen korrespondierenden Markenfluss-Prozess $\mu_K = (bpo, X)$, $bpo = (V, <, l)$, von (N, m_0). Dieser hat die Eigenschaft, dass die Prozess-BPO von K mit der Markenfluss-BPO von μ_K übereinstimmt. Die BPO bpo ist gegeben als die $*$-Erweiterung der Prozess-BPO bpo_K, d.h. $V \setminus \{q_{bpo}, s_{bpo}\} = E$, $<|_{(V\setminus\{q_{bpo}, s_{bpo}\})\times(V\setminus\{q_{bpo}, s_{bpo}\})} = G^+|_{E\times E}$ und $l|_{V\setminus\{q_{bpo}, s_{bpo}\}} = \rho|_E$. Die Menge der Markenfluss-Funktionen X ist durch folgende Eigenschaften festgelegt:

(i) $\phi^{-1}(p)(v, v') = |\{b \in B | \rho(b) = p, b \in v^\bullet \cap {}^\bullet v'\}|$ für $p \in P$ und $v, v' \in V \setminus \{q_{bpo}, s_{bpo}\}$ mit $v < v'$,

(ii) $\phi^{-1}(p)(q_{bpo}, v) = |\{b \in B | \rho(b) = p, b \in {}^\bullet v, {}^\bullet b = \emptyset\}|$ für $p \in P$ und $v \in V \setminus \{q_{bpo}, s_{bpo}\}$,

(iii) $\phi^{-1}(p)(v, s_{bpo}) = |\{b \in B | \rho(b) = p, b \in v^\bullet, b^\bullet = \emptyset\}|$ für $p \in P$ und $v \in V \setminus \{q_{bpo}, s_{bpo}\}$,

(iv) $\phi^{-1}(p)(q_{bpo}, s_{bpo}) = |\{b \in B | \rho(b) = p, {}^\bullet b = \emptyset, b^\bullet = \emptyset\}|$ für $p \in P$.

LEMMA 4.3.13 *Lemma 4.3.13*

Sei $K = (O, \rho)$, $O = (B, E, G)$, *ein Prozessnetz eines markierten S/T-Netzes* (N, m_0), $N = (P, T, W)$, *dann ist* $\mu_K = (bpo, X)$, $bpo = (V, <, l)$, *ein Markenfluss-Prozess von* (N, m_0). *Dabei gilt, dass die Prozess-BPO* bpo_K *mit der Markenfluss-BPO* bpo_{μ_K} *übereinstimmt.*

BEWEIS: Wir prüfen alle Eigenschaften eines Markenfluss-Prozesses:

- Da bpo_K eine BPO ist, ist bpo eine $*$-BPO.
- Bei (i) bis (iv) wird schon vorausgesetzt, dass X eine Menge von Markenfluss-Funktionen ist, für die eine bijektive Funktion $\phi : X \to P$ existiert.
- Die Eigenschaft $l(V \setminus \{q_{bpo}, s_{bpo}\}) \subseteq T$ ergibt sich aus der Eigenschaft $\rho(E) \subseteq T$ von K.
- Mit (ii) und (iv) gilt für $x \in X$ die Beziehung $Ab_x(q_{bpo}) = \sum_{v \in V \setminus \{q_{bpo}, s_{bpo}\}} x(q_{bpo}, v) + x(q_{bpo}, s_{bpo}) = |\{b \in Min(O) \mid \rho(b) = \phi(x)\}| = m_0(\phi(x))$.
- Mit (i) und (ii) bzw. (i) und (iii) zusammen mit den Eigenschaften $< |_{(V \setminus \{q_{bpo}, s_{bpo}\}) \times (V \setminus \{q_{bpo}, s_{bpo}\})} = G^+|_{E \times E}$ und $l|_{V \setminus \{q_{bpo}, s_{bpo}\}} = \rho|_E$ gilt für $v \in V \setminus \{q_{bpo}, s_{bpo}\}$ und $x \in X$ die Beziehung $Zu_x(v) = \sum_{v' \in V \setminus \{q_{bpo}, s_{bpo}\}, v' < v} x(v', v) + x(q_{bpo}, v) = |\{b \in {}^\bullet v \mid \rho(b) = \phi(x)\}| = W(\phi(x), l(v))$ bzw. $Ab_x(v) = \sum_{v' \in V \setminus \{q_{bpo}, s_{bpo}\}, v < v'} x(v, v') + x(v, s_{bpo}) = |\{b \in v^\bullet \mid \rho(b) = \phi(x)\}| = W(l(v), \phi(x))$.

Der Zusatz $bpo_K = bpo_{\mu_K}$ ergibt sich direkt aus $V \setminus \{q_{bpo}, s_{bpo}\} = E$, $< |_{(V \setminus \{q_{bpo}, s_{bpo}\}) \times (V \setminus \{q_{bpo}, s_{bpo}\})} = G^+|_{E \times E}$ und $l|_{V \setminus \{q_{bpo}, s_{bpo}\}} = \rho|_E$. □

Es bleibt noch anzumerken, dass per Konstruktion K_{μ_K} isomorph zu dem ursprünglichen Prozessnetz K ist, und in μ_{K_μ} gegenüber dem ursprünglichen Markenfluss-Prozess μ nur Kanten, auf denen jede Markenfluss-Funktion den Wert 0 hat, wegfallen können. Die zwei vorgestellten Konstruktionsregeln kehren sich in diesem Sinne also jeweils um.

BEISPIEL: *Wie schon zuvor informal erklärt, definiert das zweite Prozessnetz aus Abbildung 48 den Markenfluss-Prozess in Abbildung 65. Die entsprechende Prozess-BPO und die entsprechende Markenfluss-BPO stimmen überein. Diese BPO ist die zweite in Abbildung 48 dargestellte BPO. Umgekehrt definiert der Markenfluss-Prozess in Abbildung 65 auch das zweite Prozessnetz aus Abbildung 48.* *Beispiel*

Als Folgerung der in den zwei letzten Lemmata beschriebenen Zusammenhänge zwischen Prozessnetzen und Markenfluss-Prozessen erhalten wir die folgende Konsistenz zwischen den zugehörigen Ablaufbegriffen.

Satz 4.3.3

SATZ 4.3.3
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes S/T-Netz. Die partielle Sprache der Prozess-BPOs von* (N, m_0) *mit minimaler Ordnung stimmt mit der partiellen Sprache der Markenfluss-BPOs von* (N, m_0) *mit minimaler Ordnung überein.*

BEWEIS: Wir beweisen einen offensichtlich zur Aussage des Satzes äquivalenten Zusammenhang, nämlich dass der Sequentialisierungsabschluss der partiellen Sprache der Prozess-BPOs von (N, m_0) mit dem Sequentialisierungsabschluss der partiellen Sprache der Markenfluss-BPOs von (N, m_0) übereinstimmt.
Da nach Lemma 4.3.12 jede Markenfluss-BPO eine Sequentialisierung einer Prozess-BPO ist, ist auch jede Sequentialisierung einer Markenfluss-BPO eine Sequentialisierung einer Prozess-BPO. Da nach Lemma 4.3.13 jede Prozess-BPO eine Markenfluss-BPO ist, ist jede Sequentialisierung einer Prozess-BPO eine Sequentialisierung einer Markenfluss-BPO □

Prozessnetze und Markenfluss-Prozesse beschreiben also in konsistenter Weise Ablaufverhalten von S/T-Netzen. Mit Satz 3.3.1 ergibt sich daraus eine Übereinstimmung der Begriffe der Markenfluss-BPO und der aktivierten BPO.

Satz 4.3.4

SATZ 4.3.4
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes S/T-Netz. Die partielle Sprache der in* m_0 *bzgl.* N *aktivierten BPOs und die partielle Sprache der Markenfluss-BPOs von* (N, m_0) *stimmen überein.*

BEWEIS: Entsprechend Satz 3.3.1 stimmt die Menge der in m_0 bzgl. N aktivierten BPOs mit minimaler Ordnung mit der Menge der Prozess-BPOs von (N, m_0) mit minimaler Ordnung überein. Diese stimmt nach Satz 4.3.3 mit der Menge der Markenfluss-BPOs von (N, m_0) mit minimaler Ordnung überein. Da sowohl die Menge der in m_0 bzgl. N aktivierten BPOs als nach Lemma 4.3.11 auch die Menge der Markenfluss-BPOs von (N, m_0) sequentialisierungsabgeschlossen sind, stimmen beide Mengen überein. □

Beispiel

BEISPIEL: *Die partielle Sprache aus Abbildung 47 entspricht sowohl der Menge aller aktivierten BPOs des Netzes aus Abbildung 3 als auch der Menge aller Markenfluss-BPOs des Netzes. Außerdem ergibt sich diese partielle Sprache als Sequentialisierungsabschluss der Menge aller Prozess-BPOs des Netzes.*

Der letzte Satz zeigt, dass die durch Markenfluss-Prozesse definierte Ablaufsemantik der Standard-Ablaufsemantik entspricht. Da in dem Syntheseproblem aus Problemspezifikation 4.1.1 die Standard-Ablaufsemantik zugrundegelegt wird, sind somit Markenfluss-Prozesse zur Lösung des Problems geeignet. Mit Satz 4.3.4 leiten wir im Folgenden eine auf dem Begriff des Markenfluss-Prozesses basierende Regionendefinition für partielle Sprachen her.
Eine Region soll wieder ein zulässiges Stellentripel definieren. Aus den bisherigen Überlegungen ergibt sich, dass ein Stellentripel genau dann zulässig bzgl. einer partiellen Sprache ist, wenn alle BPOs der Sprache Markenfluss-BPOs des zum Stellentripel gehörigen atomaren Netzes sind. Dies ist genau dann der Fall, wenn für jede BPO der Sprache ein Markenfluss-Prozess mit folgenden Eigenschaften existiert:

- Erstens liegt dem Markenfluss-Prozess die $*$-Erweiterung der betrachteten BPO zugrunde, so dass die BPO die zum Markenfluss-Prozess zugehörige Markenfluss-BPO ist.

- Zweitens handelt es sich um einen Markenfluss-Prozess des betrachteten atomaren Netzes. Dies bedeutet, dass der Markenfluss-Prozess genau eine Markenfluss-Funktion besitzt und diese zur Stelle des atomaren Netzes zugehörig ist. Hierzu muss der Marken-Abfluss des Quellenknotens mit der Anfangsmarkierung der betrachteten Stelle übereinstimmen. Ebenso muss der Marken-Zufluss bzw. Marken-Abfluss jedes nicht-minimalen und nicht-maximalen Ereignisses dem Kantengewicht der eingehenden bzw. ausgehenden Kante der durch die Ereignisbeschriftung gegebenen Transition entsprechen.

Beispiel

BEISPIEL: *Wir betrachten die Stelle aus Abbildung 59 links und die partielle Sprache aus Abbildung 4. Die zwei Markenfluss-Prozesse aus Abbildung 67 erfüllen dann die gewünschten Eigenschaften, d.h. die zwei BPOs aus Abbildung 4 korrespondieren zu den zwei Markenfluss-Prozessen und es handelt sich um Markenfluss-Prozesse des Netzes aus Abbildung 59 links. Daher ist die dargestellte Stelle zulässig. Für die Stelle rechts in Abbildung 59 lassen sich solche Markenfluss-Prozesse nicht finden. Sie ist also nicht zulässig.*

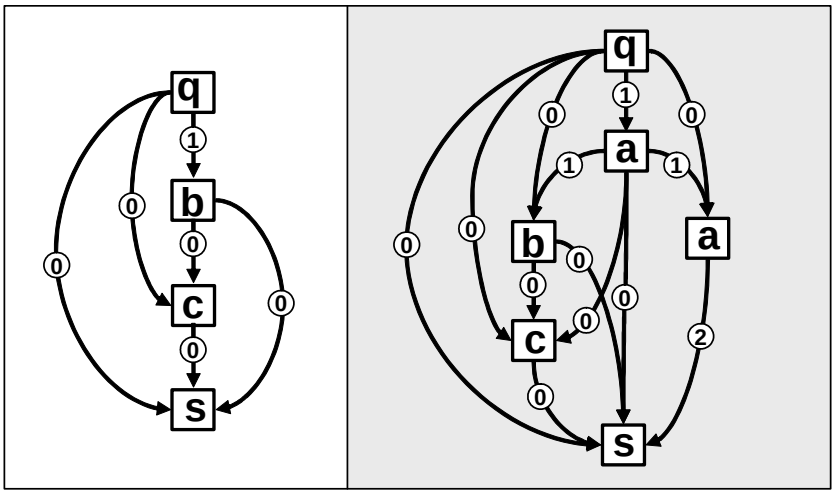

Abbildung 67: Markenfluss-Region der partiellen Sprache aus Abbildung 4.

In dem Fall, dass für jede BPO einer partiellen Sprache ein Markenfluss-Prozess, wie gerade beschrieben, existiert, stimmen insbesondere alle Marken-Abflüsse der Quellenknoten sowie alle Marken-Abflüsse bzw. Marken-Zuflüsse gleich beschrifteter nicht-minimaler und nicht-maximaler Ereignisse all dieser Markenfluss-Prozesse überein. Eine Familie von partiell geordneten Markenflüssen mit jeweils nur einer Markenfluss-Funktion, welche diese Eigenschaft erfüllt, nennen wir konsistent.

Definition 4.3.16 (Konsistenz von partiell geordneten Markenflüssen)

DEFINITION 4.3.16 (KONSISTENZ VON PARTIELL GEORDNETEN MARKENFLÜSSEN)
Ein partiell geordneter Markenfluss $\mu = (\mathrm{bpo}, X)$ *heißt atomar, falls* $|X| = 1$. *Eine Familie von atomaren partiell geordneten Markenflüssen* $(\mu_i)_{i\in I}$ $\mu_i = (\mathrm{bpo}_i, \{x_i\})$, $\mathrm{bpo}_i = (V_i, <_i, l_i)$, *für* $i \in I$, *heißt konsistent, falls für alle* $i, j \in I$ *die Gleichung*

$$(\mathrm{ANF}) \qquad \mathrm{Ab}_{x_i}(q_{\mathrm{bpo}_i}) = \mathrm{Ab}_{x_j}(q_{\mathrm{bpo}_j})$$

erfüllt ist und für alle $v_i \in V_i \setminus \{q_{bpo_i}, s_{bpo_i}\}, v_j \in V_j \setminus \{q_{bpo_j}, s_{bpo_j}\}$, $i, j \in I$, *mit* $l_i(v_i) = l_j(v_j)$ *die Gleichungen*

$$(ZU) \qquad Zu_{x_i}(v_i) = Zu_{x_j}(v_j),$$

$$(AB) \qquad Ab_{x_i}(v_i) = Ab_{x_j}(v_j)$$

erfüllt sind.

Die Idee ist nun, dass ausgehend von einer konsistenten Familie von partiell geordneten Markenflüssen in natürlicher Weise ein bzgl. der Menge der zu den partiell geordneten Markenflüssen korrespondierenden BPOs zulässiges Stellentripel definiert werden kann, indem

- die Anfangsmarkierung entsprechend der Marken-Abflüsse der Quellenknoten der partiell geordneten Markenflüsse gewählt wird,
- der Multivorbereich des Stellentripels bzgl. einer Transition, d.h. das Gewicht der Kante von der betrachteten Transition zur zu definierenden Stelle, entsprechend der Marken-Abflüsse der nicht-minimalen und nicht-maximalen, mit der betrachteten Transition beschrifteten Ereignisse der partiell geordneten Markenflüsse gewählt wird und
- der Multinachbereich des Stellentripels bzgl. einer Transition, d.h. das Gewicht der Kante von der zu definierenden Stelle zur betrachteten Transition, entsprechend der Marken-Zuflüsse der nicht-minimalen und nicht-maximalen, mit der betrachteten Transition beschrifteten Ereignisse der partiell geordneten Markenflüsse gewählt wird.

Die Konsistenz der Familie der atomaren partiell geordneten Markenflüsse garantiert, dass das Stellentripel wohldefiniert ist. Die Überlegungen vor Definition 4.3.16 zeigen zum einen, dass das Stellentripel zulässig bzgl. der Menge der zu den partiell geordneten Markenflüssen korrespondierenden BPOs ist. Zum anderen lassen sich auf diese Weise auch alle zulässigen Stellentripel identifizieren.

Beispiel

BEISPIEL: *Um eine zulässige Stelle zu finden, kehren wir also die Vorgehensweise des letzten Beispiels um. Ausgehend von einer konsistenten Familie partiell geordneter Markenflüsse wird eine bzgl. der Menge der korrespondierenden BPOs zulässige Stelle konstruiert. Beispielsweise definiert die konsistente Familie partiell geordneter Markenflüsse aus Abbildung 67 die in Abbildung 59 links dargestellte bzgl. der partiellen Sprache aus Abbildung 4 zulässige Stelle. Wäre die betrachtete Familie nicht konsistent, dann wären die festzulegenden Kantengewichte bzw. die Anfangsmarkierung nicht eindeutig.*

Dementsprechend ergibt sich das folgende Regionenkonzept. Eine Markenfluss-Region einer partiellen Sprache soll eine konsistente Familie von partiell geordneten Markenflüssen derart festlegen, dass die Menge der korrespondierenden BPOs genau der partiellen Sprache entspricht. Mit dieser Familie von partiell geordneten Markenflüssen wird dann entsprechend obigen Ausführungen ein bzgl. der partiellen Sprache zulässiges Stellentripel definiert. Da die $*$-BPOs der partiell geordneten Markenflüsse in diesem Fall als $*$-Erweiterungen der BPOs der partiellen Sprache gegeben sind, müssen nur noch die Markenfluss-Funktionen der atomaren partiell geordneten Markenflüsse geeignet

festgelegt werden, d.h. es muss eine Markenfluss-Funktion für jede $*$-Erweiterung einer BPO der partiellen Sprache festgelegt werden. Dementsprechend ist eine Markenfluss-Region einer partiellen Sprache eine sog. globale Markenfluss-Funktion, welche jeder Kante jeder $*$-Erweiterung einer BPO der partiellen Sprache eine nicht-negative ganze Zahl zuordnet. Die Einschränkung der globalen Markenfluss-Funktion auf die Kanten einer $*$-Erweiterung definiert dann die Markenfluss-Funktion auf dieser $*$-Erweiterung.

Definition 4.3.17 (Globale Markenfluss-Funktion)

DEFINITION 4.3.17 (GLOBALE MARKENFLUSS-FUNKTION)
Sei $\mathcal{L}$ eine partielle Sprache. Eine globale Markenfluss-Funktion von $\mathcal{L}$ ist eine Funktion $r : \bigcup_{(V,<,l)\in\mathcal{L}} <^* \to \mathbb{N}$. Eine globale Markenfluss-Funktion r definiert eine Markenfluss-Funktion $r|_{<^*}$ auf jeder $*$-BPO bpo^* für $bpo = (V,<,l) \in \mathcal{L}$. Dadurch legt eine globale Markenfluss-Funktion r eine Familie von atomaren partiell geordneten Markenflüssen $(\mu_{bpo})_{bpo\in\mathcal{L}}$, $\mu_{bpo} = (bpo^*, \{r|_{<^*}\})$ für $bpo = (V,<,l) \in \mathcal{L}$, fest. Für die Markenfluss-Funktionen $r|_{<^*}$ schreiben wir oft auch kurz nur r, wenn dies nicht zu Verwechslungen führen kann.

Die zentrale Anforderung an eine Markenfluss-Region ist nun, dass die von einer globalen Markenfluss-Funktion repräsentierte Familie von partiell geordneten Markenflüssen konsistent ist.

Definition 4.3.18 (Markenfluss-Region)

DEFINITION 4.3.18 (MARKENFLUSS-REGION)
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Eine Markenfluss-Region von $\mathcal{L}$ ist eine globale Markenfluss-Funktion r von $\mathcal{L}$ mit der Eigenschaft, dass die zugehörige Familie von atomaren partiell geordneten Markenflüssen $(\mu_{bpo})_{bpo\in\mathcal{L}}$, $\mu_{bpo} = (bpo^*, \{r|_{<^*}\})$ für $bpo = (V,<,l) \in \mathcal{L}$, konsistent ist.
Jede Markenfluss-Region r von $\mathcal{L}$ definiert ein korrespondierendes Stellentripel $\overrightarrow{r}$ über T durch $\overrightarrow{r}_0 = Ab_{r|_{<^*}}(q_{bpo^*})$ für ein $bpo = (V,<,l) \in \mathcal{L}$, $^\circ\overrightarrow{r}(t) = Ab_{r|_{<^*}}(v)$ für $t \in T$, ein $bpo = (V,<,l) \in \mathcal{L}$ und ein $v \in V$ mit $l(v) = t$ und $\overrightarrow{r}^\circ(t) = Zu_{r|_{<^*}}(v)$ für $t \in T$, ein $bpo = (V,<,l) \in \mathcal{L}$ und ein $v \in V$ mit $l(v) = t$.

Es ist zu beachten, dass eine Markenfluss-Region im Gegensatz zu Transitions-Regionen nicht mehr direkt die Einträge eines Stellentripels kodiert. Da wir bisher auch unendliche partielle Sprachen zulassen, ist es auch möglich, dass die Kantenmenge, welche die Definitionsmenge einer Markenfluss-Region darstellt, unendlich ist. Somit ist es im Gegensatz zu Transitions-Regionen im Allgemeinen nicht einmal möglich, eine Markenfluss-Region als endlichen Vektor darzustellen.
Es bleibt noch formal nachzuweisen, dass sich mit Markenfluss-Regionen tatsächlich die Menge der zulässigen Stellentripel charakterisieren lässt.

Satz 4.3.5

SATZ 4.3.5
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Ein Stellentripel über T ist genau dann zulässig bzgl. $\mathcal{L}$, wenn es zu einer Markenfluss-Region von $\mathcal{L}$ korrespondiert.

BEWEIS: Sei r eine Markenfluss-Region von $\mathcal{L}$. Dann ist per Definition für alle $bpo = (V,<,l) \in \mathcal{L}$ der partiell geordnete Markenfluss $\mu_{bpo} = (bpo^*, \{r|_{<^*}\})$ ein Markenfluss-Prozess von $(N_{\overrightarrow{r}}, m_{\overrightarrow{r}})$. Damit ist jede BPO aus $\mathcal{L}$ eine Markenfluss-BPO von $(N_{\overrightarrow{r}}, m_{\overrightarrow{r}})$. Aus Satz 4.3.4 folgt dann, dass $\overrightarrow{r}$ ein bzgl. $\mathcal{L}$ zulässiges Stellentripel ist.

Sei umgekehrt st ein bzgl. $\mathcal{L}$ zulässiges Stellentripel. Dann ist nach Satz 4.3.4 jede BPO aus $\mathcal{L}$ eine Markenfluss-BPO von (N_{st}, m_{st}), d.h. es gibt für jede BPO bpo $\in \mathcal{L}$ einen Markenfluss-Prozess $\mu_{bpo} = (bpo^*, \{x_{bpo}\})$. Aus der Definition eines Markenfluss-Prozesses folgt, dass die Familie $(\mu_{bpo})_{bpo\in\mathcal{L}}$ konsistent ist. Damit ist die globale Markenfluss-Funktion $r : \bigcup_{(V,<,l)\in\mathcal{L}} <^* \to \mathbb{N}$, welche durch $r|_{<^*} = x_{bpo}$ für bpo $= (V, <, l) \in \mathcal{L}$ definiert ist, eine Markenfluss-Region von $\mathcal{L}$. Außerdem folgt auch aus der Definition eines Markenfluss-Prozesses, dass $\overrightarrow{r} = st$. □

Im Gegensatz zu den zwei Arten von Transitions-Regionen der letzten beiden Unterabschnitte gibt es im Allgemeinen keine eins-zu-eins-Korrespondenz zwischen der Menge der Markenfluss-Regionen von $\mathcal{L}$ und der Menge der zulässigen Stellentripel bzgl. $\mathcal{L}$. Während es zu jeder Markenfluss-Region per Definition genau ein korrespondierendes zulässiges Stellentripel gibt, ist ein zulässiges Stellentripel im Allgemeinen zu mehreren Markenfluss-Regionen korrespondierend. Dies liegt daran, dass es für ein und dieselbe Stelle bzgl. einer Markenfluss-BPO unterschiedliche mögliche Verteilungen des Flusses der Marken geben kann. Umgekehrt können dann natürlich durch unterschiedliche Markenfluss-Regionen gegebene Verteilungen des Markenflusses derselben Stelle entsprechen.

Beispiel

BEISPIEL: *In Abbildung 67 ist eine globale Markenfluss-Funktion der partiellen Sprache aus Abbildung 4 dargestellt. Wie zuvor diskutiert, ist die zugehörige Familie von atomaren partiell geordneten Markenflüssen konsistent. Dementsprechend ist die globale Markenfluss-Funktion eine Markenfluss-Region. Gemäß den bisherigen Überlegungen korrespondiert die zulässige Stelle aus Abbildung 59 links zu dieser Region.*

Zur Berechnung von zulässigen Stellentripeln mithilfe von Markenfluss-Regionen betrachten wir im Folgenden den Fall, dass $\mathcal{L}$ endlich ist. In diesem Fall ist eine Markenfluss-Region von $\mathcal{L}$ auf einer endlichen Kantenmenge definiert. Somit lässt sich eine Markenfluss-Region durch einen Vektor darstellen, dessen Einträge die Funktionswerte der Markenfluss-Region kodieren. Ein solcher Vektor beinhaltet also für jede Kante einer $*$-Erweiterung einer BPO aus $\mathcal{L}$ einen Eintrag. Um eine Zuordnung zwischen den Vektoreinträgen und den Kanten zu erreichen, wird eine Nummerierung all dieser Kanten festgelegt. Die Stelle eines Vektoreintrages referenziert dann auf die Nummer der zugehörigen Kante. Diese Vekorrepräsentation ist generell für globale Markenfluss-Funktionen von $\mathcal{L}$ sinnvoll. Während wir im Falle von Transitions-Regionen Stellentripel durch Vektoren dargestellt haben, werden nun also globale Markenfluss-Funktionen wie folgt durch Vektoren dargestellt.

Definition 4.3.19 (Korrespondenz zwischen globalen Markenfluss-Funktionen und Vektoren)

DEFINITION 4.3.19 (KORRESPONDENZ ZWISCHEN GLOBALEN MARKENFLUSS-FUNKTIONEN UND VEKTOREN)
Sei $\mathcal{L}$ eine endliche partielle Sprache. Wir betrachten eine Nummerierung der Menge $\bigcup_{(V,<,l)\in\mathcal{L}} <^* = \{e_1, \ldots, e_n\}$ der Kanten aller $*$-Erweiterungen der BPOs aus $\mathcal{L}$. Jeder Vektor $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{N}^n$ definiert bzgl. $\{e_1, \ldots, e_n\}$ eine korrespondierende globale Markenfluss-Funktion $r_{\mathbf{x}} : \bigcup_{(V,<,l)\in\mathcal{L}} <^* \to \mathbb{N}$ von $\mathcal{L}$ durch $r_{\mathbf{x}}(e_j) = x_j$. Für eine globale Markenfluss-Funktion $r : \bigcup_{(V,<,l)\in\mathcal{L}} <^* \to \mathbb{N}$ von $\mathcal{L}$ heißt umgekehrt der eindeutige Vektor $\mathbf{x} \in \mathbb{N}^n$ mit $r_{\mathbf{x}} = r$ bzgl. $\{e_1, \ldots, e_n\}$ der zu r bzgl. $\{e_1, \ldots, e_n\}$ korrespondierende Vektor.

BEISPIEL: *Der Vektor* $(0,0,1,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,2)$ *korrespondiert bzgl. der in Abbildung 68 dargestellten Kanten-Nummerierung zu der globalen Markenfluss-Funktion aus Abbildung 67.* *Beispiel*

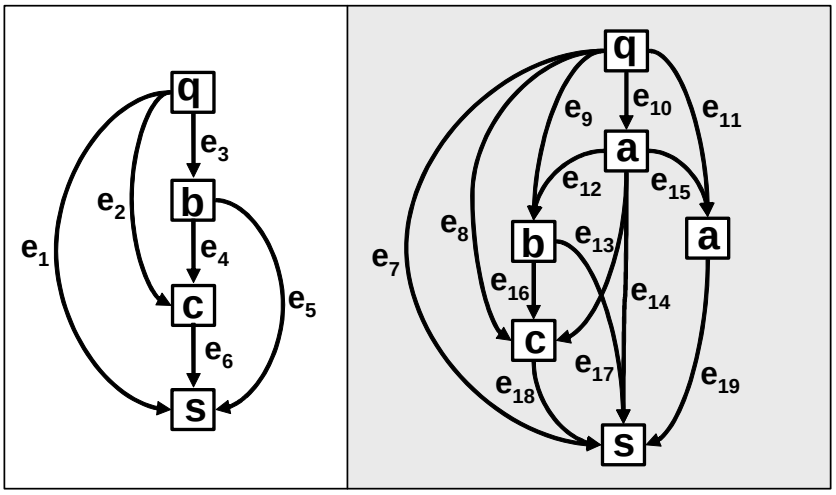

Abbildung 68: Eine Nummerierung der Kanten der $*$-Erweiterungen von bpo_1 und bpo_2 aus Abbildung 4.

Auf dieser Vektor-Ebene lassen sich nun weiter die Konsistenz-Eigenschaften (ANF), (ZU) und (AB) einer Markenfluss-Region zu einem entsprechenden homogenen linearen Gleichungssystem zusammenfassen. Dies kann grob gesagt dadurch erreicht werden, dass für ein Paar von Quellenknoten von $*$-Erweiterungen der BPOs aus $\mathcal{L}$ die Werte der globalen Markenfluss-Funktion auf ausgehenden Kanten des einen Knotens positiv und des anderen Knotens negativ gezählt werden. Eine Gleichung, welche diese Summe gleich Null setzt, garantiert, dass der Marken-Abfluss der zwei Quellenknoten identisch ist. Damit ist die Anforderung (ANF) für die entsprechenden BPOs erfüllt. Wenn eine Reihenfolge der BPOs aus $\mathcal{L}$ festgelegt wird, genügt es, die Anforderung (ANF) für die erste und zweite BPO, für die zweite und dritte BPO, usw. sicherzustellen. Ähnlich können für ein Paar von gleich beschrifteten Knoten von BPOs aus $\mathcal{L}$ die Werte der globalen Markenfluss-Funktion auf eingehenden bzw. ausgehenden Kanten des einen Knotens positiv und des anderen Knotens negativ gezählt werden. Eine Gleichung, welche diese Summe gleich Null setzt, garantiert, dass der Marken-Zufluss bzw. Marken-Abfluss der zwei Knoten identisch ist. Damit ist die Anforderung (ZU) bzw. (AB) für die entsprechenden Knoten erfüllt. Wenn für jede Beschriftung eine Reihenfolge aller entsprechend beschrifteter Knoten von BPOs aus $\mathcal{L}$ festgelegt wird, genügt es, für jede Beschriftung die Anforderungen (ZU) und (AB) für den ersten und zweiten entsprechend beschrifteten Knoten, für den zweiten und dritten entsprechend beschrifteten Knoten, usw. sicherzustellen. Aus diesen Überlegungen ergibt sich die folgende linear algebraische Charakterisierung der Menge aller Markenfluss-Regionen von $\mathcal{L}$.

LEMMA 4.3.14 *Lemma 4.3.14*

Sei $\mathcal{L} = \{bpo_1, \ldots, bpo_l\}$, $bpo_i = (V_i, <_i, l_i)$ *für* $i \in \{1, \ldots, l\}$, *eine endliche partielle Sprache mit Beschriftungsmenge* T *in geordneter Form. Wir betrachten eine Nummerierung der Menge* $\bigcup_{i=1}^{l} <_i^* = \{e_1, \ldots, e_n\}$ *der Kanten*

aller $*$-Erweiterungen der BPOs aus $\mathcal{L}$. Weiter betrachten wir für jedes $t \in T$ eine Nummerierung der Menge $\bigcup_{i=1}^{l}\{v \in V_i \mid l_i(v) = t\} = \{v_1^t, \ldots, v_{k_t}^t\}$ der mit t beschrifteten Knoten von BPOs aus $\mathcal{L}$. Damit definieren wir ein homogenes lineares Gleichungssystem $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}$. Die Matrix $\mathbf{A}_{\mathcal{L}}^{MAR}$ setzt sich aus einem Zeilenvektor $\mathbf{anf}_m = (anf_{m,1}, \ldots, anf_{m,n})$ für jedes $m \in \{1, \ldots, l-1\}$ sowie einem Zeilenvektor $\mathbf{zu}_m^t = (zu_{m,1}^t, \ldots, zu_{m,n}^t)$ und einem Zeilenvektor $\mathbf{ab}_m^t = (ab_{m,1}^t, \ldots, ab_{m,n}^t)$ für jedes $t \in T$ und $m \in \{1, \ldots, k_t - 1\}$ zusammen. Die Zeilenvektoren sind gegeben durch:

$$anf_{m,j} = \begin{cases} 1 & \text{falls } e_j \text{ eine ausgehende Kante von } q_{bpo_m} \text{ ist,} \\ -1 & \text{falls } e_j \text{ eine ausgehende Kante von } q_{bpo_{m+1}} \text{ ist,} \\ 0 & \text{sonst.} \end{cases}$$

$$zu_{m,j}^t = \begin{cases} 1 & \text{falls } e_j \text{ eine eingehende Kante von } v_m^t \text{ ist,} \\ -1 & \text{falls } e_j \text{ eine eingehende Kante von } v_{m+1}^t \text{ ist,} \\ 0 & \text{sonst.} \end{cases}$$

$$ab_{m,j}^t = \begin{cases} 1 & \text{falls } e_j \text{ eine ausgehende Kante von } v_m^t \text{ ist,} \\ -1 & \text{falls } e_j \text{ eine ausgehende Kante von } v_{m+1}^t \text{ ist,} \\ 0 & \text{sonst.} \end{cases}$$

Es gilt, dass die Menge der Markenfluss-Regionen von $\mathcal{L}$ genau der Menge der zu ganzzahligen Lösungen des homogenen linearen Ungleichungssystems

$$\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$$

bzgl. $\{e_1, \ldots, e_n\}$ korrespondierenden globalen Markenfluss-Funktionen entspricht, d.h. $r : \bigcup_{(V,<,l)\in\mathcal{L}} <^* \to \mathbb{N}$ ist genau dann eine Markenfluss-Region von $\mathcal{L}$, wenn $r = r_{\mathbf{r}}$ bzgl. $\{e_1, \ldots, e_n\}$ für eine ganzzahlige Lösung $\mathbf{r}$ des homogenen linearen Ungleichungssystems $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$.
Für eine solche Lösung $\mathbf{r}$ bezeichnen wir im Weiteren das zu $r_{\mathbf{r}}$ korrespondierende Stellentripel $\overrightarrow{r_{\mathbf{r}}}$ auch als das zu $\mathbf{r}$ korrespondierend Stellentripel $\overrightarrow{\mathbf{r}}$.

BEWEIS: Sei $\mathbf{r} \in \mathbb{N}^n$ und $r = r_{\mathbf{r}}$ die bzgl. $\{e_1, \ldots, e_n\}$ korrespondierende globale Markenfluss-Funktion von $\mathcal{L}$. Wir betrachten die zu r gehörige Familie von atomaren partiell geordneten Markenflüssen $(\mu_{bpo})_{bpo\in\mathcal{L}}$, $\mu_{bpo} = (bpo^*, \{r|_{<^*}\})$ für $bpo = (V, <, l) \in \mathcal{L}$. Der Zeilenvektor $\mathbf{anf}_m$ ist derart definiert, dass $\mathbf{anf}_m \cdot \mathbf{r} = \mathbf{0}$ äquivalent dazu ist, dass $(\mu_{bpo})_{bpo\in\mathcal{L}}$ die Eigenschaft (ANF) für die BPOs bpo_m und bpo_{m+1} erfüllt. Der Zeilenvektor $\mathbf{zu}_m^t$ ist derart definiert, dass $\mathbf{zu}_m^t \cdot \mathbf{r} = \mathbf{0}$ äquivalent dazu ist, dass $(\mu_{bpo})_{bpo\in\mathcal{L}}$ die Eigenschaft (ZU) für die Knoten v_m^t und v_{m+1}^t erfüllt. Der Zeilenvektor $\mathbf{ab}_m^t$ ist derart definiert, dass $\mathbf{ab}_m^t \cdot \mathbf{r} = \mathbf{0}$ äquivalent dazu ist, dass $(\mu_{bpo})_{bpo\in\mathcal{L}}$ die Eigenschaft (AB) für die Knoten v_m^t und v_{m+1}^t erfüllt. Insgesamt ist also $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{r} = \mathbf{0}$ äquivalent dazu, dass $(\mu_{bpo})_{bpo\in\mathcal{L}}$ die Eigenschaft (ANF) für alle BPOs aus $\mathcal{L}$ erfüllt und die Eigenschaften (ZU) und (AB) für alle gleich beschrifteten Knoten von BPOs aus $\mathcal{L}$ erfüllt. Somit ist $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{r} = \mathbf{0}$ nach Definition 4.3.16 äquivalent dazu, dass $(\mu_{bpo})_{bpo\in\mathcal{L}}$ konsistent ist. Unter Beachtung der Korrespondenz zwischen globalen Markenfluss-Funktionen und Vektoren aus Definition 4.3.19 folgt, dass eine globale Markenfluss-Funktion r genau dann eine Markenfluss-Region von $\mathcal{L}$ entsprechend Definition 4.3.18 ist, wenn $r = r_{\mathbf{r}}$ bzgl. $\{e_1, \ldots, e_n\}$ für eine ganzzahlige Lösung

$\mathbf{r}$ des homogenen linearen Ungleichungssystems $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$. Dabei entspricht der Anteil $\mathbf{x} \geqslant \mathbf{0}$ des betrachteten Ungleichungssystems zusammen mit der Forderung nach Ganzzahligkeit der Forderung, dass die Einträge von $\mathbf{r}$ aus $\mathbb{N}$ sind. □

BEMERKUNG 4.3.4 (ÜBERSETZUNG IN EIN UNGLEICHUNGSSYSTEM) *Bemerkung 4.3.4 (Übersetzung in ein Ungleichungssystem)*
Das Gleichungssystem $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}$ lässt sich in das äquivalente Ungleichungssystem $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} \geqslant \mathbf{0}, -\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} \geqslant \mathbf{0}$ überführen. Somit lassen sich alle in dieser Arbeit betrachteten Verfahren, welche sich auf Ungleichungssysteme dieser Form beziehen, auch für die linear algebraische Repräsentation der Markenfluss-Regionen einer partiellen Sprache verwenden. Hierbei ist allerdings zu beachten, dass es für viele Verfahren, wie beispielsweise das Simplex-Verfahren zur Lösung des Zulässigkeitsproblems oder die doppelte Beschreibungsmethode zur Lösung des Extremalstrahl-Aufzählungs-Problems, Versionen gibt, welche auch eine Eingabe der ursprünglichen Form $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ erlauben und für diese effizienter arbeiten. Dementsprechend sollte für Implementierungen und damit auch für Performance-Überlegungen unbedingt diese ursprüngliche Form der linear algebraischen Charakterisierung von Markenfluss-Regionen betrachtet werden.

Da wir im Gegensatz zu Transitions-Regionen Markenfluss-Regionen nicht in einer Vektorform eingeführt haben, sei noch einmal darauf hingewiesen, dass wir im Rahmen einer linear algebraischen Charakterisierung zuerst eine Vektorrepräsentation von Markenfluss-Regionen entwickelt haben. In dem vorausgehenden Lemma haben wir darauf aufbauend Markenfluss-Regionen als ganzzahlige Lösungen eines linearen Ungleichungssystems dargestellt. Aufgrund der eins-zu-eins-Korrespondenz zwischen globalen Markenfluss-Funktionen und Vektoren aus Definition 4.3.19 ergibt sich sogar eine eins-zu-eins-Korrespondenz zwischen der Menge der Markenfluss-Regionen von $\mathcal{L}$ und der Menge der ganzzahligen Lösungen von $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$.

BEISPIEL: *Abbildung 69 zeigt das lineare Gleichungssystem $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}$ für die partielle Sprache aus Abbildung 4 bzgl. der Kanten-Nummerierung aus Abbildung 68 (eine Nummerierung der BPOs und der gleich beschrifteten Knoten spielt hier keine entscheidende Rolle, da es nur zwei BPOs und höchstens zwei Knoten mit derselben Beschriftung gibt). Die erste Zeile $\mathbf{anf}_1$ der Matrix stellt sicher, dass die zwei Quellenknoten denselben Marken-Abfluss aufweisen. Genauer gesagt wird die Anforderung, dass die Summe der den von $q_{bpo_1^*}$ ausgehenden Kanten e_1, e_2 und e_3 zugeordneten Werte der Summe der den von $q_{bpo_2^*}$ ausgehenden Kanten e_7, e_8, e_9, e_{10} und e_{11} zugeordneten Werte entspricht, dargestellt. Außerdem gibt es drei Paare von Knoten, welche mit derselben Transition beschriftet sind. Die zweite bis siebte Zeile der Matrix stellen sicher, dass die entsprechenden Knoten jeweils denselben Marken-Zufluss und denselben Marken-Abfluss haben. Die zweite Zeile $\mathbf{zu}_1^a$ garantiert, dass die zwei a-Ereignisse denselben Marken-Zufluss aufweisen, und die dritte Zeile $\mathbf{ab}_1^a$ garantiert einen übereinstimmenden Marken-Abfluss. Die vierte Zeile $\mathbf{zu}_1^b$ und die fünfte Zeile $\mathbf{ab}_1^b$ bzw. die sechste Zeile $\mathbf{zu}_1^c$ und die siebte Zeile $\mathbf{ab}_1^c$ bewirken jeweils dasselbe für die beiden b-Ereignisse bzw. die beiden c-Ereignisse. Es lässt sich nachprüfen, dass der im letzten Beispiel betrachtete Vektor $(0,0,1,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,2)$, welcher zur Markenfluss-Region aus Abbildung 67 korrespondiert, eine Lösung des Gleichungssystems ist.* *Beispiel*

Bei Betrachtung der Größe der Matrix $\mathbf{A}_{\mathcal{L}}^{MAR}$ zeigt sich, dass sowohl die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{MAR}$ als auch die Anzahl der Zeilen

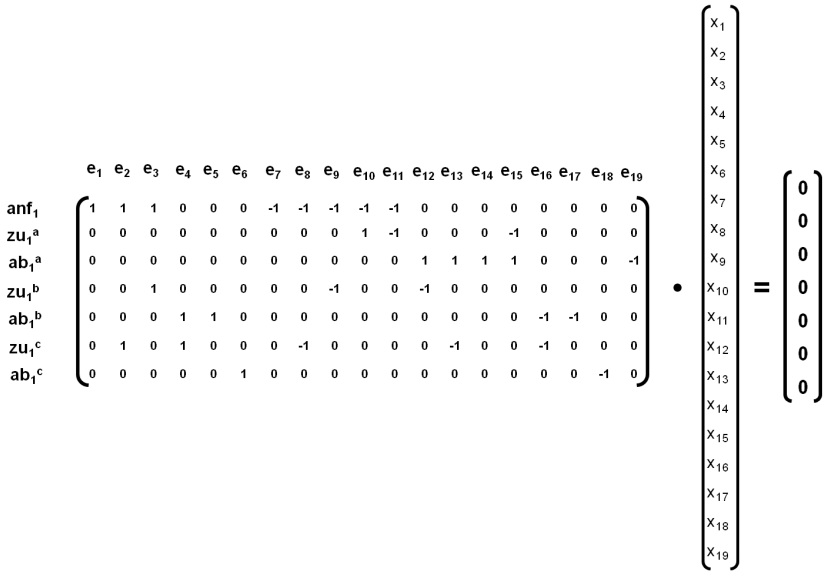

Abbildung 69: Gleichungssystem $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}$.

von $\mathbf{A}_{\mathcal{L}}^{MAR}$ linear in der Größe der Eingabe $\mathcal{L}$ sind. Die Anzahl der Spalten entspricht der Anzahl der Kanten $\bigcup_{i=1}^{l} <_i^* = \{e_1, \ldots, e_n\}$ aller $*$-Erweiterungen der BPOs aus $\mathcal{L}$. Diese Kanten setzen sich zusammen aus den Kanten der BPOs aus $\mathcal{L}$ und je zwei Kanten für jeden Knoten einer BPO aus $\mathcal{L}$, nämlich einer vom Quellenknoten zu dem Knoten führenden Kante und einer vom Knoten zu dem Senkenknoten führenden Kante. Die Zeilen von $\mathbf{A}_{\mathcal{L}}^{MAR}$ setzen sich aus den **anf**-, den **zu**- und den **ab**-Zeilen zusammen. Die Anzahl der **ab**-Zeilen entspricht gerade $|\mathcal{L}| - 1$, ist also um eins geringer als die Anzahl der BPOs aus $\mathcal{L} = \{bpo_1, \ldots, bpo_l\}$. Die Anzahl der **zu**- und der **ab**-Zeilen hängt von den Mengen von Knoten $\bigcup_{i=1}^{l} \{v \in V_i \mid l_i(v) = t\} = \{v_1^t, \ldots, v_{k_t}^t\}$ der BPOs aus $\mathcal{L}$ mit jeweils gleicher Beschriftung ab. Im schlimmsten Fall haben alle Knoten dieselbe Beschriftung. Dann ist die Anzahl der **zu**- und der **ab**-Zeilen jeweils um eins geringer als die Anzahl aller Knoten aller BPOs aus $\mathcal{L}$. Trotz dieser gutartigen Abhängigkeit der Größe von $\mathbf{A}_{\mathcal{L}}^{MAR}$ von der Größe der Eingabe $\mathcal{L}$, sind Verfahren zum Verkleinern von $\mathbf{A}_{\mathcal{L}}^{MAR}$ nützlich. Vor allem die Anzahl der Spalten, also der Variablen des Ungleichungssystems, ist im Vergleich zu Transitions-Regionen sehr hoch und kann algorithmische Probleme bereiten. Wir schlagen in Bemerkung 4.3.5 drei Optimierungsverfahren vor, um insbesondere die Anzahl der Spalten, aber auch die Anzahl der Zeilen, von $\mathbf{A}_{\mathcal{L}}^{MAR}$ zu reduzieren.

Bemerkung 4.3.5 (Optimierungsvorschläge)

BEMERKUNG 4.3.5 (OPTIMIERUNGSVORSCHLÄGE)
Die folgenden Optimierungen der linear algebraischen Charakterisierung von Markenfluss-Regionen aus Lemma 4.3.14 sind möglich.

- *Wiederum ist es entsprechend Korollar 4.3.1 möglich, nur die BPOs aus der Repräsentationsmenge* $\mathrm{Rep}(\mathcal{L})$ *anstelle aller BPOs aus* $\mathcal{L}$ *zu betrachten. Durch das Weglassen von BPOs reduzieren sich die insgesamt vorkommenden Kanten, wodurch sich die Anzahl der Spalten von* $\mathbf{A}_{\mathcal{L}}^{MAR}$ *deutlich reduzieren kann. Außerdem wird gleichzeitig die Anzahl der Zeilen von* $\mathbf{A}_{\mathcal{L}}^{MAR}$ *reduziert, da sowohl BPO-Knoten als auch*

BPOs wegfallen. Demgegenüber steht allerdings wiederum der zusätzliche Aufwand zur Berechnung von $\mathrm{Rep}(\mathcal{L})$ *aus* $\mathcal{L}$ *und das Problem, dass* $\mathrm{Rep}(\mathcal{L})$ *aus Anwendungssicht typischerweise nicht sehr viel kleiner ist als* $\mathcal{L}$*.*

- *Da die Variablen des betrachteten Ungleichungssystems gerade den Kanten der* $*$*-Erweiterungen der BPOs aus* $\mathcal{L}$ *entsprechen, müssten zum Reduzieren der Spalten von* $\mathcal{L}$ *Kanten weggelassen werden. Hierzu ist ein vielversprechender Ansatzpunkt, dass die vollen Kanteninformationen einer BPO schon allein in den Gerüstkanten enthalten sind. Wir werden im Folgenden skizzieren, inwiefern die Betrachtung der Hasse-Diagramme der* $*$*-Erweiterungen der BPOs aus* $\mathcal{L}$ *zur Definition von Markenfluss-Regionen ausreicht. Wir folgen dabei ähnlichen Überlegungen wie in [180]. Dort wurde dieser Ansatz aber in einem völlig anderen Kontext, nämlich rein zur Betrachtung von Entscheidbarkeitsproblemen für unendliche partielle Sprachen, nicht aber zu algorithmischen Zwecken, entwickelt. Die Idee ist, dass wir für jedes der zu betrachtenden Hasse-Diagramme vier Markenfluss-Funktionen berücksichtigen, d.h. eine Markenfluss-Region setzt sich dann aus vier entsprechenden globalen Markenfluss-Funktionen zusammen. Die vier Markenflüsse auf einer von einem Knoten* v *zu einem Knoten* v' *führenden Kante haben dabei verschiedene Bedeutungen. Je einer der Markenflüsse repräsentiert*
  - *die Anzahl der Marken, welche von* v *produziert und von* v' *konsumiert wird (die übliche Bedeutung eines Markenflusses),*
  - *die Anzahl der Marken, welche in der Vergangenheit von* v *produziert und von* v' *konsumiert wird (diese muss Null sein, falls* v *ein Quellenknoten ist),*
  - *die Anzahl der Marken, welche von* v *produziert und in der Zukunft von* v' *konsumiert wird (diese muss Null sein, falls* v' *ein Senkenknoten ist) und*
  - *die Anzahl der Marken, welche in der Vergangenheit von* v *produziert und in der Zukunft von* v' *konsumiert wird (diese muss Null sein, falls* v *ein Quellenknoten oder* v' *ein Senkenknoten ist).*

  Wichtig ist, dass die verschiedenen Flüsse konsistent zueinander sein müssen, d.h. Marken, welche für die Zukunft bestimmt sind, müssen dort dann auch in entsprechenden Flüssen, welche aus der Vergangenheit stammen, wieder auftauchen und umgekehrt. Für jedes Schaltereignis ergibt sich damit die Anforderung, dass die Summe der eingehenden Flüsse, welche in der Zukunft des Ereignisses konsumiert werden, der Summe der ausgehenden Flüsse, welche in der Vergangenheit des Ereignisses produziert wurden, entspricht. Mit diesem Ansatz lassen sich dann Markenflüsse auf Kanten des transitiven Abschlusses der Gerüstkanten durch entsprechende Markenflüsse auf den zugehörigen Gerüstkanten kodieren. Somit lässt sich eine analoge Definition von Markenfluss-Regionen gewinnen. Ein wesentlicher Nachteil dieses Ansatzes ist, dass zur Sicherstellung der Konsistenzbedingungen zwischen den vier Markenflüssen zusätzliche Gleichungen formuliert werden müssen. Für jeden Knoten außer den Quellen- und Senkenknoten ergibt sich dadurch eine zusätzliche Zeile von $\mathbf{A}_{\mathcal{L}}^{\mathrm{MAR}}$*. Außerdem müssen vier globale Markenfluss-Funktionen betrachtet werden. Dadurch vervierfacht sich im Prinzip die Anzahl der Variablen des Ungleichungssystems und somit die Anzahl der Spalten von* $\mathbf{A}_{\mathcal{L}}^{\mathrm{MAR}}$*. Al-*

lerdings wirkt hier normalerweise der Effekt stärker, dass die globalen Markenfluss-Funktionen nur noch auf Gerüstkanten betrachtet werden. Es gilt, dass die Anzahl aller Kanten einer BPO quadratisch in der Anzahl der Gerüstkanten sein kann. Also kann es zu einer entsprechenden Reduzierung der Anzahl der Variablen des Ungleichungssystems und damit der Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{MAR}$ kommen.

- *Unabhängig von obigen Optimierungsvorschlägen kann es auch sinnvoll sein, generelle Verfahren zur Größenreduktion linearer Ungleichungssysteme anzuwenden. Hierbei kommt vor allem wieder das Suchen und Entfernen redundanter Ungleichungen des Ungleichungssystems in Frage. Aber auch das Suchen und Eliminieren nicht benötigter Variablen des Ungleichungssystems kann angesichts der Vielzahl an Variablen nützlich sein. Natürlich bleibt durch solche Verfahren eine korrekte linear algebraische Charakterisierung der Menge der Markenfluss-Regionen erhalten. Entsprechende Verfahren sind kurz in Abschnitt 3.4 diskutiert.*

Beispiel

BEISPIEL: *In Abbildung 68 können mit dem zweiten Optimierungsvorschlag die Kanten e_1, e_2, e_5, e_7, e_8, e_9, e_{11}, e_{13}, e_{14} und e_{17} weggelassen werden. Somit verbleiben nur 9 der 19 Kanten. Allerdings ergibt sich in diesem Fall dennoch eine Erhöhung der Variablenanzahl eines entsprechenden Gleichungssystems, da für jede Kante vier Variablen betrachtet werden müssen. Außerdem ergeben sich 6 zusätzliche Gleichungen für die Konsistenzbedingungen bzgl. der 6 Schaltereignisse.*

Die Regionendefinition dieses Abschnittes ist mit den zwei vorhergehenden Transitions-Regionen-Konzepten kaum zu vergleichen. Sie gewährleistet im Gegensatz zu Transitions-Regionen eine polynomielle Abhängigkeit der Größe der entsprechenden linearen Ungleichungssysteme von der Größe der Eingabe. Allerdings basieren die Ungleichungssysteme auf einer relativ hohen Anzahl an Variablen.
Im nächsten Unterabschnitt stellen wir noch eine vierte indirekte Möglichkeit zur Definition von Regionen für partielle Sprachen vor. Es ist möglich, geeignete Transitionssysteme zu betrachten und darauf aufbauend Regionen für partielle Sprachen zu definieren. Da dies aber einerseits eher als eine unnatürliche und komplizierte Hilfskonstruktion anmutet und andererseits weit in das in der Literatur schon intensiv untersuchte Gebiet der Regionen von Transitionssystemen führt, sollen hier nur die Grundlagen eines solchen Ansatzes geklärt werden.

4.3.4 *Schritt-Transitionssystem-Regionen*

In diesem Unterabschnitt stellen wir eine Möglichkeit zur Definition von Regionen für partielle Sprachen vor, welche sehr ähnlich zu den Schritt-Transitions-Regionen aus Unterabschnitt 4.3.1 ist. Wie in Lemma 4.3.4 diskutiert, lassen sich Regionen für partielle Sprachen mit Hilfe der Menge der von der partiellen Sprache erzeugten Schrittfolgen definieren. Diese Menge lässt sich mit Hilfe eines Schritt-Transitionssystems darstellen. Die Idee ist nun, diesen Zusammenhang zu nutzen, um bekannte Regionenkonzepte für Schritt-Transitionssysteme im Rahmen einer Definition von Regionen für partielle Sprachen zu verwenden.
Es gibt verschiedenste Regionendefinitionen für Transitionssysteme und Schritt-Transitionssysteme, u.a. [90, 173, 13, 17, 18]. Für den Fall von S/T-Netzen lassen sich alle Definitionen auf die folgende allgemeine

Darstellung zurückführen. Für Details und weitere Erklärungen sei auf die einschlägige Literatur verwiesen.

DEFINITION 4.3.20 (TRANSITIONSSYSTEM-REGION)
Sei T *eine endliche Menge an Transitionen. Eine Transitionssystem-Region eines deterministischen, erreichbaren, initialisierten Schritt-Transitionssystems* $TS = (S, \mathbb{N}^T, \to, s_0)$ *ist ein Paar* $r = (\sigma, \eta)$, *wobei* $\sigma : S \to \mathbb{N}$ *und* $\eta = (\eta_1, \eta_2) : T \to \mathbb{N} \times \mathbb{N}$ *Funktionen sind, welche für jedes* $(s, \tau, s') \in \to$ *die folgenden Eigenschaften erfüllen:*

Definition 4.3.20 (Transitionssystem-Region)

$$(TS1) \qquad \sigma(s) \geqslant \sum_{t \in T} \tau(t) \cdot \eta_1(t),$$

$$(TS2) \qquad \sigma(s') = \sigma(s) + \sum_{t \in T} \tau(t) \cdot (\eta_2(t) - \eta_1(t)).$$

Eine Transitionssystem-Region $r = (\sigma, \eta)$ *definiert ein korrespondierendes Stellentripel* $\overrightarrow{r}$ *durch* $\overrightarrow{r}_0 = \sigma(s_0)$, $^\circ\overrightarrow{r}(t) := \eta_2(t)$ *und* $\overrightarrow{r}^\circ(t) := \eta_1(t)$ *für* $t \in T$.

Es ist zu beachten, dass wir keine speziellen Anforderungen an ein Schritt-Transitionssystems stellen. Insbesondere setzen wir keinerlei Axiome wie die „Intermediate State Property“ [173] oder die „Subset Property“ [173, 17, 18] (siehe auch [146]) voraus, welche sicherstellen, dass die als Beschriftungen fungierenden Multimengen tatsächlich nebenläufige Schritte repräsentieren. Weiter ist zu beachten, dass eine Transitionssystem-Region $r = (\sigma, \eta)$ alleine durch $\sigma(s_0)$ und η oder aber auch durch σ und η_1 bzw. σ und η_2 festgelegt ist. Insbesondere gilt also, dass verschiedene Regionen auch verschiedene korrespondierende Stellentripel haben.
Ein Stellentripel st ist zulässig bzgl. eines Schritt-Transitionssystems TS, wenn der Schritterreichbarkeitsgraph $\mathfrak{NEG}(N_{st}, m_{st})$ grob gesagt das gegebene Schritt-Transitionssystem enthält (wobei Zustände zusammenfallen können). Dies lässt sich formal dadurch ausdrücken, dass ein Graph-Morphismus von TS zu $\mathfrak{NEG}(N_{st}, m_{st})$ existiert. Ein solches Stellentripel ist im Hinblick auf die Synthese eines Netzes aus einem Schritt-Transitionssystem nicht zu restriktiv. Auch wenn wir an dieser Stelle nicht näher auf den Begriff der Zulässigkeit eines Stellentripels bzgl. eines Schritt-Transitionssystems eingehen, so sei dennoch angemerkt, dass sich analoge Ergebnisse wie im Falle von Sprachen herleiten lassen, d.h. die von Transitionssystem-Regionen definierten Stellentripel entsprechen gerade der Menge der bzgl. des betrachteten Schritt-Transitionssystems zulässigen Stellentripel. Es sei in diesem Rahmen auf ähnliche Resultate in [65] verwiesen.
Die zentrale Idee zur Nutzung der letzten Definition für die Synthese aus partiellen Sprachen ist die Übersetzung der Menge der von einer partiellen Sprache erzeugten Schrittfolgen in ein Schritt-Transitionssystem ohne zusammenfallende Zustände. Dies bedeutet, dass wir für jede Schrittfolge jeweils ausgehend von dem initialen Knoten des zu konstruierenden Transitionssystems einen neuen Ast des Transitionssystems bilden. Der Hintergrund hierzu gestaltet sich wie folgt. Sei eine Region definiert durch $\sigma(s_0)$ und η. Falls zwei verschiedene vom initialen Knoten ausgehende Wege im Transitionssystem sich in einem Zustand treffen, so ergibt sich mit (TS2) die Anforderung an η, dass diese Wege dieselbe Veränderung bewirken. Auf diese Weise hat die Struktur des Transitionssystems Einfluss auf die Regionendefinition.

Ansonsten ergibt sich als einzige Anforderung an η, dass es so gewählt werden muss, dass alle Schrittfolgen des Schritt-Transitionssystems jeweils in $\sigma(s_0)$ aktiviert sind. Dies ist aber gerade auch die Anforderung an Regionen von Mengen von Schrittfolgen, welche in Definition 4.3.7 konsistent auf partielle Sprachen übertragen wurde. Wird also bei der Konstruktion eines Schritt-Transitionssystems aus einer partiellen Sprache, wie beschrieben, die Situation von zusammenführenden Wegen vermieden, so sind die Anforderungen an die Regionen des entstehenden Schritt-Transitionssystems analog zu den Anforderungen an Schritt-Transitions-Regionen. Dementsprechend ergibt sich die folgende Regionen-Definition für partielle Sprachen.

Definition 4.3.21 (Schritt-Transitionssystem-Region)

Definition 4.3.21 (Schritt-Transitionssystem-Region)
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Wir konstruieren das kleinste Schritt-Transitionssystem $TS_{\mathcal{L}} = (S, \mathbb{N}^T, \rightarrow, s_0)$, welches folgender Bedingung genügt:

$$\tau_1 \dots \tau_n \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\mathcal{L})\} \implies s_{\tau_1}, \dots, s_{\tau_1 \dots \tau_n} \in S \wedge$$

$$(s_0, \tau_1, s_{\tau_1}), (s_{\tau_1}, \tau_2, s_{\tau_1\tau_2}), \dots, (s_{\tau_1 \dots \tau_{n-1}}, \tau_n, s_{\tau_1 \dots \tau_n}) \in \rightarrow$$

Eine Transitionssystem-Region $r = (\sigma, \eta)$ von $TS_{\mathcal{L}}$ heißt Schritt-Transitionssystem-Region von $\mathcal{L}$. Eine Schritt-Transitionssystem-Region r definiert ein korrespondierendes Stellentripel $\overrightarrow{r}$.

Wie angedeutet, lässt sich zeigen, dass dieser Ansatz äquivalent zur Definition von Schritt-Transitions-Regionen ist und damit der folgende Zusammenhang gilt.

Satz 4.3.6

Satz 4.3.6
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Ein Stellentripel über T ist genau dann zulässig bzgl. $\mathcal{L}$, wenn es zu einer Schritt-Transitionssystem-Region von $\mathcal{L}$ korrespondiert.

Beweis: Sei $T = \{t_1, \dots, t_m\}$ und $TS_{\mathcal{L}} = (S, \mathbb{N}^T, \rightarrow, s_0)$.
Sei nun $r = (\sigma, \eta)$ eine Schritt-Transitionssystem-Region von $\mathcal{L}$ und $\tau_1 \dots \tau_n \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\mathcal{L})\}$ sowie $j \in \{1, \dots, n\}$. Dann ergibt sich durch wiederholtes Anwenden von (TS2) die Beziehung $\sigma(s_{\tau_1 \dots \tau_{j-1}}) = \sigma(s_{\tau_1 \dots \tau_{j-2}}) + \sum_{i=1}^{m} \tau_{j-1}(t_i) \cdot (\eta_2(t_i) - \eta_1(t_i)) = \dots = \sigma(s_0) + \sum_{i=1}^{m} (\tau_1 + \dots + \tau_{j-1})(t_i) \cdot (\eta_2(t_i) - \eta_1(t_i))$. Aus (TS1) ergibt sich dann weiter $\sigma(s_0) + \sum_{i=1}^{m} (\tau_1 + \dots + \tau_{j-1})(t_i) \cdot (\eta_2(t_i) - \eta_1(t_i)) \geqslant \sum_{i=1}^{m} \tau_j(t_i) \cdot \eta_1(t_i)$ und somit $\sigma(s_0) + \sum_{i=1}^{m} (\tau_1 + \dots + \tau_{j-1})(t_i) \cdot \eta_2(t_i) - (\tau_1 + \dots + \tau_j)(t_i) \cdot \eta_1(t_i) \geqslant 0$. Damit ist $(\sigma(s_0), \eta_2(t_1), \dots, \eta_2(t_m), \eta_1(t_1), \dots, \eta_1(t_m))$ eine Schritt-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \dots, t_m\}$. Mit Satz 4.3.1 folgt, dass $\overrightarrow{r}$ bzgl. $\mathcal{L}$ zulässig ist.
Sei umgekehrt st ein bzgl. $\mathcal{L}$ zulässiges Stellentripel. Wir definieren $\sigma : S \to \mathbb{N}$ und $\eta = (\eta_1, \eta_2) : T \to \mathbb{N} \times \mathbb{N}$ durch $\sigma(s_0) = st_0$, $\eta_2(t_i) = {}^\circ st(t_i)$ und $\eta_1(t_i) = st^\circ(t_i)$ für $i \in \{1, \dots, m\}$ sowie $\sigma(s_{\tau_1 \dots \tau_i}) = \sigma(s_0) + \sum_{i=1}^{m} (\tau_1 + \dots + \tau_j)(t_i) \cdot (\eta_2(t_i) - \eta_1(t_i))$ für $\tau_1 \dots \tau_n \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\mathcal{L})\}$, $j \in \{1, \dots, n\}$ (letztere Festlegung ist aufgrund der Struktur von $TS_{\mathcal{L}}$ wohldefiniert). Dann erfüllt $r = (\sigma, \eta)$ per Konstruktion (TS2). Außerdem ist $(\sigma(s_0), \eta_2(t_1), \dots, \eta_2(t_m), \eta_1(t_1), \dots, \eta_1(t_m))$ der bzgl. $T = \{t_1, \dots, t_m\}$ zu st korrespondierende Vektor und somit nach Satz 4.3.1 eine Schritt-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \dots, t_m\}$. Es folgt $\sigma(s_0) + \sum_{i=1}^{m} (\tau_1 + \dots + \tau_{j-1})(t_i) \cdot \eta_2(t_i) - (\tau_1 + \dots + \tau_j)(t_i) \cdot \eta_1(t_i) \geqslant 0$ für $\tau_1 \dots \tau_n \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\mathcal{L})\}$, $j \in \{1, \dots, n\}$. Hieraus ergibt sich $\sigma(s_{\tau_1 \dots \tau_{j-1}}) = \sigma(s_0) + \sum_{i=1}^{m} (\tau_1 +$

$\ldots + \tau_{j-1})(t_i) \cdot (\eta_2(t_i) - \eta_1(t_i)) \geqslant \sum_{i=1}^{m} \tau_j(t_i) \cdot \eta_1(t_i)$ und somit (TS1). Folglich ist r eine Schritt-Transitionssystem-Region von $\mathcal{L}$, zu der per Definition st korrespondiert, i.e. $st = \overrightarrow{r}$. □

BEISPIEL: *Abbildung 70 zeigt das Schritt-Transitionssystem $TS_{\mathcal{L}}$ für die partielle Sprache aus Abbildung 4. In grau ist eine Schritt-Transitionssystem-Region der partiellen Sprache dargestellt (σ ist an den Zuständen notiert und η ist unter dem Transitionssystem dargestellt). Zu dieser Region korrespondiert wiederum die zulässige Stelle aus Abbildung 59 links.* *Beispiel*

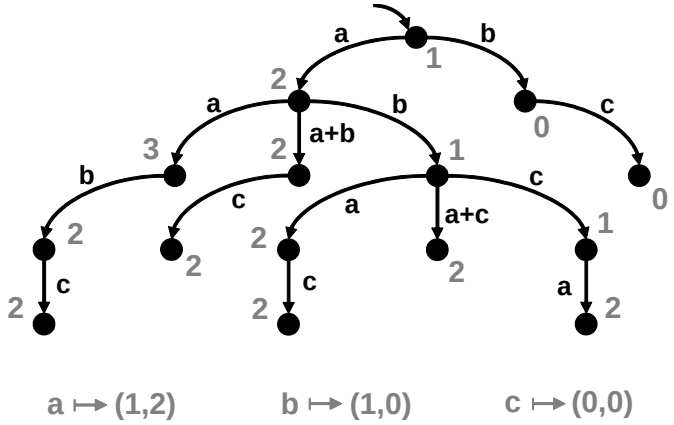

Abbildung 70: Eine Schritt-Transitionssystem-Region der partiellen Sprache aus Abbildung 4.

Da eine Schritt-Transitionssystem-Region alleine durch $\sigma(s_0)$ und η festgelegt ist, ergibt sich hier sogar wieder eine eins-zu-eins-Korrespondenz zwischen der Menge der zulässigen Stellentripel bzgl. $\mathcal{L}$ und der Menge der Schritt-Transitionssystem-Regionen von $\mathcal{L}$. Außerdem lässt sich mit dieser Überlegung, wie im letzten Beweis dargestellt, eine Schritt-Transitionssystem-Region (σ, η) von $\mathcal{L}$ bzgl. einer Ordnung $T = \{t_1, \ldots, t_m\}$ auf der Menge der Transitionen als Vektor $(\sigma(s_0), \eta_2(t_1), \ldots, \eta_2(t_m), \eta_1(t_1), \ldots, \eta_1(t_m)) \in \mathbb{N}^{2m+1}$ auffassen. Der Beweis von Satz 4.3.6 zeigt, dass die Menge der Schritt-Transitionssystem-Regionen in dieser Form bzgl. $T = \{t_1, \ldots, t_m\}$ exakt der Menge der Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ entspricht.
Trotz dieser Analogie resultieren aus den Überlegungen zu Schritt-Transitionssystem-Regionen neue Möglichkeiten zur Berechnung von zulässigen Stellentripeln. Eine Schritt-Transitionssystem-Region einer partiellen Sprache ist eine Region eines speziellen Transitionssystems. Dementsprechend ergeben sich im Rahmen der Berechnung solcher Regionen Möglichkeiten zur Nutzung von Verfahren zur Berechnung von Regionen von üblichen Transitionssystemen, z.B. [60, 53]. Diese Verfahren umfassen spezielle Konstruktionsmethoden auf Graphenebene. Teilweise gibt es hier auch verschiedene sehr gute Heuristiken. Insgesamt eröffnen die Überlegungen dieses Unterabschnittes also die Möglichkeit zur Berechnung von zulässigen Stellentripeln einer partiellen Sprache mithilfe entsprechender Graphalgorithmen für Transitionssysteme und ähnlicher auf Transitionssystemen basierender Verfahren. Diese sollen hier aber nicht näher diskutiert werden, sondern es sei auf die umfassende Literatur, welche sich mit Berechnungsverfahren für Regionen von Transitionssystemen beschäftigt, verwiesen.

Abgesehen von der Nutzbarkeit von auf Transitionssystemen basierenden Verfahren stellen Schritt-Transitionssystem-Regionen eine kompliziertere und ineffizientere Form des Ansatzes der Schritt-Transitions-Regionen dar. Wie im letzten Beweis deutlich wird, erweitern Schritt-Transitionssystem-Regionen zum einen die Variablen von Schritt-Transitions-Regionen um die Einträge von σ (außer $\sigma(s_0)$) und zum anderen kodieren sie die Restriktion (ST) durch die zwei Restriktionen (TS1) und (TS2). Dementsprechend diskutieren wir hier keine linear algebraische Charakterisierung von Schritt-Transitionssystem-Regionen (eine Übersetzung der Restriktionen in ein Ungleichungssystem ist aber natürlich wieder möglich) und werden die Definition der Schritt-Transitionssystem-Regionen im Folgenden auch nicht weiter verwenden.
Insgesamt haben wir mit dem Begriff der Schritt-Transitionssystem-Region eine Brücke von der Synthese von Petrinetzen aus partiellen Sprachen zur Theorie der Regionen von Transitionssystemen geschlagen. Darauf aufbauend haben wir die Möglichkeiten zur Nutzung von Methoden aus dem Bereich der Regionen von Transitionssystemen in unserem Kontext kurz aufgezeigt. Es bleibt hierzu noch zu bemerken, dass die verwendete Analogiebildung zwischen Regionen von Sprachen und Regionen von Transitionssystemen neuartig ist, wenn auch ähnliche Ideen schon in [7, 84] vorkommen und in [18, 65] angedeutet sind.

4.4 VERFAHREN ZUR BERECHNUNG VON REGIONEN

Wir haben bisher in diesem Kapitel dargestellt, dass die Transitionen eines aus einer endlichen partiellen Sprache $\mathcal{L}$ zu synthetisierenden Netzes in natürlicher Weise durch die Beschriftungen der Sprache gegeben sind. Wird hierzu die Menge aller zulässigen Stellentripel hinzugefügt, so entsteht das gesättigt zulässige Netz, welches zur Lösung unseres Syntheseproblems geeignet ist. Genauer gesagt stimmt das Ablaufverhalten des gesättigt zulässigen Netzes entweder mit $\mathrm{PS}(\mathcal{L})$ überein oder es gibt kein Netz mit dieser Eigenschaft. Das Hinzufügen von irgendeinem nicht-zulässigen Stellentripel führt zu einem Netz, welches unser Syntheseproblem nicht lösen kann. Die zentrale Idee ist also, zulässige Stellentripel zu identifizieren und zur Netzkonstruktion zu verwenden. Zur Identifikation zulässiger Stellentripel wurden verschiedene Regionendefinitionen auf der Struktur der partiellen Sprache entwickelt, so dass die Menge aller Regionen die Menge aller zulässigen Stellentripel festlegt. Für jede Regionendefinition wurde dann weiter eine linear algebraische Charakterisierung hergeleitet. Die Menge aller Regionen und damit die Menge aller zulässiger Stellen kann also jeweils durch die Lösungsmenge eines homogenen linearen Ungleichungssystems dargestellt werden. Das Problem dabei ist, dass die Menge aller zulässiger Stellentripel von $\mathcal{L}$ unendlich ist. In diesem Abschnitt sollen ausgehend von den verschiedenen Regionendefinitionen Verfahren zur Berechnung geeigneter endlicher Teilmengen der Menge aller zulässiger Stellentripel entwickelt werden. Werden nur die Stellentripel einer solchen Teilmenge betrachtet, so soll ein endliches Netz mit derselben Eigenschaft wie das gesättigt zulässige Netz entstehen, nämlich dass das Ablaufverhalten des Netzes entweder mit $\mathrm{PS}(\mathcal{L})$ übereinstimmt oder es kein Netz mit dem Ablaufverhalten $\mathrm{PS}(\mathcal{L})$ gibt. Die Herangehensweise ist derart, dass geeignete Regionen durch die Auswahl

geeigneter Lösungen des entsprechenden Ungleichungssystems gesucht werden und dann die entsprechenden zulässigen Stellentripel zum Netz hinzugefügt werden. Um das Syntheseproblem zu lösen, muss schließlich noch getestet werden, ob das Ablaufverhalten des resultierenden Netzes mit $PS(\mathcal{L})$ übereinstimmt oder nicht. Der letzte Punkt ist aber erst Thema des nächsten Abschnittes.
Da die Mengen der Schritt-Transitions-Regionen und der BPO-Transitions-Regionen übereinstimmen, betrachten wir in diesem Abschnitt von diesen zwei Konzepten ausschließlich die fortschrittlichere Definition der BPO-Transitions-Regionen und die zugehörige linear algebraische Charakterisierung. Es ist hierbei zu beachten, dass Schritt-Transitions-Regionen und deren linear algebraische Charakterisierung für die Berechnungsverfahren dieses Abschnittes vollkommen analog verwendet werden können. Schritt-Transitionssystem-Regionen werden wir in diesem Abschnitt nicht mehr explizit behandeln. Sie lassen sich aber, wie gezeigt, auf Schritt-Transitions-Regionen zurückführen und sind damit in den Überlegungen dieses Abschnittes indirekt beinhaltet. Neben der Diskussion von BPO-Transitions-Regionen werden die Berechnungsverfahren dieses Abschnittes noch für Markenfluss-Regionen entwickelt. Ähnlich wie im letzten Abschnitt werden wir in den einzelnen Unterabschnitten erst im Rahmen der Verwendung linear algebraischer Charakterisierungen von Regionen eine Einschränkung auf endliche partielle Sprachen vornehmen.

4.4.1 *Schrittseparation*

Interessante Ansätze zur Berechnung einer endlichen Repräsentation aller Regionen finden sich in [17] für Regionen von Schritt-Transitionssystemen und in [63] für Regionen von regulären Sprachen jeweils bzgl. S/T-Netzen (siehe [18, 65] für allgemeinere Darstellungen). Die beiden Ansätze sind sehr ähnlich. Durch eine Kombination der Ideen lässt sich ein analoges Vorgehen auch für Schritt-Sprachen herleiten. Auf einem derartigen Vorgehen basieren die Ausführungen dieses Unterabschnittes.
Vorweg soll noch einmal betont werden, dass genau für die Netze (N, m_0), welche ausschließlich bzgl. einer gegeben partiellen Sprache $\mathcal{L}$ zulässige Stellen enthalten, $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$ gilt. Die zentrale Idee des Ansatzes zur Berechnung einer geeigneten endlichen Menge an zulässigen Stellentripeln in diesem Unterabschnitt ist es, solche Stellentripel zu betrachten, welche das durch $\mathcal{L}$ spezifizierte Verhalten $PS(\mathcal{L})$ von dem nicht-spezifizierten Verhalten, also dem Komplement von $PS(\mathcal{L})$, separieren (ähnlich wie in [17, 63, 18, 65]). Der entscheidende Schritt hierzu ist, die gesamte unendliche Menge von nicht in $\mathcal{L}$ spezifiziertem Verhalten durch eine endliche Menge von sog. falschen Fortsetzungen zu repräsentieren. Dann wird versucht, für jede falsche Fortsetzung ein zulässiges Stellentripel zu finden, welches die falsche Fortsetzung verhindert. Falls dies nicht möglich ist, so soll folgen, dass das Syntheseproblem eine negative Antwort hat, da dann durch entsprechende falsche Fortsetzungen gegebenes nicht-spezifiziertes Verhalten mit zulässigen Stellentripeln nicht ausgeschlossen werden kann, d.h. für jedes Netz gilt in diesem Fall, dass, falls das Ablaufverhalten des Netzes $PS(\mathcal{L})$ enthält, es dann auch nicht-spezifiziertes Verhalten enthalten muss. Wird dahingegen für jede falsche Fortsetzung ein entsprechendes zulässiges Stellentripel gefunden, so wird pro falscher Fortsetzung ein

solches Stellentripel ausgewählt und zum Netz hinzugefügt. In diesem Fall soll das resultierende Netz die gewünschte Eigenschaft, dass das Netz entweder das Syntheseproblem positiv löst, oder, falls dies nicht der Fall ist, auch kein anderes Netz das Problem löst, erfüllen. Dazu muss die endliche Menge der falschen Fortsetzungen grob gesagt so definiert sein, dass, falls keine der falschen Fortsetzungen in einem Netz schalten kann, dann auch kein anderes nicht-spezifiziertes Verhalten in dem Netz möglich ist. Es ist zu beachten, dass sich dies in unserem Fall aber nur für Verhalten eines bestimmten Typs, nämlich durch Schrittfolgen gegebenes Verhalten, realisieren lässt.
Im Folgenden leiten wir eine entsprechende Definition einer falschen Fortsetzung basierend auf Überlegungen zu Schrittfolgen her. Damit weist die Herangehensweise Ähnlichkeiten zum Ansatz der Schritt-Transitions-Regionen in Unterabschnitt 4.3.1 auf. Das bedeutet, dass wir die Menge der von $PS(\mathcal{L})$ erzeugten Schrittfolgen betrachten. Die Aktiviertheit aller Schrittfolgen im Komplement dieser Menge soll durch zulässige Stellentripel verhindert werden. Im Prinzip soll also für jede solche Schrittfolge ein zulässiges Stellentripel berechnet werden, so dass die Schrittfolge im zugehörigen atomaren Netz nicht aktiviert ist. Hierzu sind die folgenden Überlegungen wichtig, welche ähnlich zu den Überlegungen im Rahmen von Lemma 4.3.7 sind. Falls eine Schrittfolge $\tau_1 \dots \tau_j$ in einem Netz nicht aktiviert ist, so sind auch die Schrittfolgen $\tau_1 \dots \tau_j \tau_{j+1}$ und $\tau_1 \dots \tau'_j$ mit $\tau_j \leqslant \tau'_j$ nicht aktiviert. Werden also bestimmte nicht von $PS(\mathcal{L})$ erzeugte Schrittfolgen verhindert, so werden auch alle längeren Schrittfolgen, die natürlich auch nicht von $PS(\mathcal{L})$ erzeugt werden, verhindert. Entsprechend genügt es, alle von $PS(\mathcal{L})$ erzeugten Schrittfolgen in allen Möglichkeiten um eine Transition zu erweitern und, falls die resultierende Schrittfolge nicht von $PS(\mathcal{L})$ erzeugt wird, diese durch ein zulässiges Stellentripel zu verhindern.

Definition 4.4.1 (Falsche Schrittfortsetzung)

Definition 4.4.1 (Falsche Schrittfortsetzung)
Sei $\mathcal{L}$ eine partielle Sprache mit Beschriftungsmenge T. Sei $S = \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}$ die Menge der von $PS(\mathcal{L})$ erzeugten Schrittfolgen. Die Menge der falschen Schrittfortsetzungen von $\mathcal{L}$ ist definiert als $\mathcal{L}_{ws} = \{\tau_1 \dots \tau_{j-1}(\tau_j + t) \notin S \mid \tau_1 \dots \tau_j \in S, t \in T\} \cup \{\tau_1 \dots \tau_j t \notin S \mid \tau_1 \dots \tau_j \in S, t \in T\}$.

Wichtig ist, dass diese Menge der falschen Schrittfortsetzungen für eine endliche partielle Sprache endlich ist, während das Komplement der von $PS(\mathcal{L})$ erzeugten Schrittfolgen in diesem Fall unendlich ist. Das weitere Vorgehen beruht darauf, zulässige Stellentripel zu betrachten, welche falsche Schrittfortsetzungen verhindern.

Definition 4.4.2 (Schrittseparierendes zulässiges Stellentripel)

Definition 4.4.2 (Schrittseparierendes zulässiges Stellentripel)
Sei $\sigma = \tau_1 \dots \tau_j \in \mathcal{L}_{ws}$ eine falsche Schrittfortsetzung von $\mathcal{L}$. Ein bzgl. σ schrittseparierendes zulässiges Stellentripel ist ein zulässiges Stellentripel st *von $\mathcal{L}$ mit der Eigenschaft, dass σ in (N_{st}, m_{st}) nicht aktiviert ist.*

Es ist hier zu beachten, dass $\tau_1 \dots \tau_{j-1} \in \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}$, d.h. es gibt eine BPO $bpo \in PS(\mathcal{L})$ mit $\tau_1 \dots \tau_{j-1} = \sigma(bpo)$. Da st zulässig bzgl. $\mathcal{L}$ ist, folgt, dass $\tau_1 \dots \tau_{j-1}$ in (N_{st}, m_{st}) aktiviert ist. Der Grund dafür, dass σ nicht aktiviert ist, kann also nur darin liegen, dass der letzte Schritt τ_j nicht mehr aktiviert ist.

Beispiel

BEISPIEL: *Wir betrachten die partielle Sprache $\mathcal{L}$ aus Abbildung 4. Ausgehend von der von $\mathrm{PS}(\mathcal{L})$ erzeugten Schrittfolge a ergeben sich beispielsweise die falschen Fortsetzungen ac, $2a$, $a+b$ und $a+c$. Entsprechend der ersten falschen Fortsetzung soll verhindert werden, dass c nach a schalten kann. Entsprechend der weiteren drei falschen Fortsetzungen soll ausgeschlossen werden, dass am Anfang a nebenläufig zu einer anderen Transition schalten kann. Die Folgen ab und aa sind keine falschen Fortsetzungen, da diese von $\mathrm{PS}(\mathcal{L})$ erzeugt werden und somit erlaubt sind. Insgesamt hat $\mathcal{L}$ 71 falsche Fortsetzungen.*
Die zulässige Stelle p_2 in Abbildung 3 ist sowohl bzgl. $2a$ als auch bzgl. $a+b$ schrittseparierend, da sowohl a als auch b eine Marke aus der Stelle konsumieren, diese zu Beginn jedoch nur mit einer Marke belegt ist, d.h. sie verhindert die Aktiviertheit beider Schrittfolgen $2a$ und $a+b$. Die zulässige Stelle p_4 ist entsprechend bzgl. ac und $a+c$ schrittseparierend.

Die zentrale Syntheseidee ist nun, zur Menge der Beschriftungen von $\mathcal{L}$ für jede falsche Schrittfortsetzung σ ein bzgl. σ schrittseparierendes zulässiges Stellentripel hinzuzufügen, falls dies möglich ist. Es wird also versucht, ein Netz zu konstruieren, in dem alle BPOs aus $\mathcal{L}$ aktiviert sind, aber keine falsche Schrittfortsetzung von $\mathcal{L}$ aktiviert ist. Dies führt zu einer sog. Schrittseparations-Repräsentation des gesättigt zulässigen Netzes.

Definition 4.4.3 (Schrittseparations-Repräsentation)

DEFINITION 4.4.3 (SCHRITTSEPARATIONS-REPRÄSENTATION)
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge. Sei ST eine Menge von zulässigen Stellentripeln von $\mathcal{L}$ mit folgender Eigenschaft: Für jedes $\sigma \in \mathcal{L}_{ws}$ enthält ST ein bzgl. σ schrittseparierendes zulässiges Stellentripel oder es gibt kein bzgl. σ schrittseparierendes zulässiges Stellentripel. Das markierte S/T-Netz (N, m_0), dessen Menge von atomaren Teilnetzen der Menge aller von Stellentripeln aus ST definierten atomaren Netzen entspricht, ist eine Schrittseparations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$.

Ist $\mathcal{L}$ endlich, so ist auch $\mathcal{L}_{ws}$ endlich. Daher gibt es in diesem Fall immer eine endliche Schrittseparations-Repräsentation. Eine solche ergibt sich beispielsweise dadurch, dass für jede falsche Schrittfortsetzung, für die es ein schrittseparierendes zulässiges Stellentripel gibt, ein solches zur durch die Beschriftungsmenge von $\mathcal{L}$ gegebenen Transitionsmenge hinzugefügt wird.

Beispiel

BEISPIEL: *Das Netz aus Abbildung 3 ist eine Schrittseparations-Repräsentation des gesättigt zulässigen Netzes der partiellen Sprache aus Abbildung 4. Die 71 falschen Fortsetzungen der Sprache werden alle durch eine der vier Stellen des Netzes separiert.*

Offensichtlich ist jedes markierte S/T-Netz (N, m_0) eine Schrittseparations-Repräsentation des gesättigt zulässigen Netzes von $\mathfrak{Bpo}(N, m_0)$, d.h. es separiert alle falschen Fortsetzungen seines Ablaufverhaltens. Im Rahmen der Synthese eines Netzes aus einer partiellen Sprache $\mathcal{L}$ gilt damit, dass, falls es ein Lösungsnetz für das Syntheseproblem gibt, dieses dann eine Schrittseparations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$ sein muss.
Das für die Synthese entscheidende Ergebnis ist nun, dass jede Schrittseparations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$ auch die zentrale Eigenschaft des gesättigt zulässigen Netzes, dass das Netz entweder das Syntheseproblem positiv löst, oder aber es keine positive Lösung gibt, aufweist.

Satz 4.4.1

SATZ 4.4.1
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge und sei (N, m_0) eine Schrittseparations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$. Falls $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$, so gilt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N', m_0')$ für alle markierten S/T-Netze (N', m_0').

BEWEIS: Angenommen es gilt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ und es gibt ein markiertes S/T-Netz (N', m_0') mit $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$. Aus $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ folgt entsprechend Lemma 4.3.1 $PS(\mathcal{L}) \subsetneq \mathfrak{Bpo}(N, m_0)$, da sich (N, m_0) nur aus zulässigen Stellentripeln zusammensetzt. Mit der Bezeichnung $\mathcal{S} = \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}$ folgt entsprechend Definition 3.3.8 daraus $\mathcal{S} \subseteq \{\sigma(bpo) \mid bpo \in Slin(\mathfrak{Bpo}(N, m_0))\} = \mathfrak{Step}(N, m_0)$. Wir unterscheiden nun zwei Fälle.
Im ersten Fall gilt $\mathcal{S} \subsetneq \mathfrak{Step}(N, m_0)$, d.h. es gibt $\sigma = \tau_1 \dots \tau_j \in \mathfrak{Step}(N, m_0) \setminus \mathcal{S}$. Sei $i \in \{0, \dots, j-1\}$ maximal, derart, dass $\tau_1 \dots \tau_i \in \mathcal{S}$. Dann gilt $\tau_1 \dots \tau_{i+1} \in \mathfrak{Step}(N, m_0) \setminus \mathcal{S}$. Da $\tau_1 \dots \tau_i \in \mathcal{S}$, aber $\tau_1 \dots \tau_{i+1} \notin \mathcal{S}$, gibt es ein $\tau_1 \dots \tau_i \tau \in \mathcal{L}_{ws}$ mit $\tau \leqslant \tau_{i+1}$. Aus $\tau_1 \dots \tau_{i+1} \in \mathfrak{Step}(N, m_0)$ und $\tau \leqslant \tau_{i+1}$ folgt $\tau_1 \dots \tau_i \tau \in \mathfrak{Step}(N, m_0)$. Entsprechend der Definition der Schrittseparations-Repräsentation hat (N, m_0) ein atomares Teilnetz, welches einem bzgl. $\tau_1 \dots \tau_i \tau \in \mathcal{L}_{ws}$ schrittseparierenden zulässigen Stellentripel entspricht, oder es gibt kein bzgl. $\tau_1 \dots \tau_i \tau \in \mathcal{L}_{ws}$ schrittseparierendes zulässiges Stellentripel. Da $\tau_1 \dots \tau_i \tau \in \mathfrak{Step}(N, m_0)$ ist der erste Fall nicht möglich. Aus $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$ folgt nach Lemma 4.3.1, dass alle atomaren Teilnetze von (N', m_0') zulässigen Stellentripeln entsprechen. Da es kein bzgl. $\tau_1 \dots \tau_i \tau \in \mathcal{L}_{ws}$ schrittseparierendes zulässiges Stellentripel gibt folgt $\tau_1 \dots \tau_i \tau \in \mathfrak{Step}(N', m_0')$. Für die eindeutige BPO bpo mit $\sigma(bpo) = \tau_1 \dots \tau_i \tau$ gilt dann, dass $bpo \in \mathfrak{Bpo}(N', m_0')$. Da $\tau_1 \dots \tau_i \tau \notin \mathcal{S}$, gilt außerdem $bpo \notin PS(\mathcal{L})$. Dies ist ein Widerspruch zu $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$.
Im zweiten Fall gilt $\mathcal{S} = \mathfrak{Step}(N, m_0)$. Da $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$, gilt auch $\mathcal{S} = \mathfrak{Step}(N', m_0')$. Aus $PS(\mathcal{L}) \subsetneq \mathfrak{Bpo}(N, m_0)$ folgt, dass ein $bpo \in \mathfrak{Bpo}(N, m_0) \setminus PS(\mathcal{L})$ existiert. Dann folgt für $bpo' \in Slin(bpo)$, dass $\sigma(bpo') \in \mathfrak{Step}(N, m_0) = \mathcal{S} = \mathfrak{Step}(N', m_0')$. Daraus folgt direkt $bpo \in \mathfrak{Bpo}(N', m_0')$. Zusammen mit $bpo \notin PS(\mathcal{L})$ ergibt auch dieser Fall einen Widerspruch zu $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$. □

Wenn das Syntheseproblem also positiv lösbar ist, so ist jede Schrittseparations-Repräsentation ein Lösungsnetz. Dementsprechend genügt es, eine beliebige Schrittseparations-Repräsentation zu betrachten. Sei eine Schrittseparations-Repräsentation (N, m_0) gegeben, dann ist das Syntheseproblem genau dann positive lösbar, wenn $PS(\mathcal{L}) = \mathfrak{Bpo}(N, m_0)$ gilt. In dem folgendem Lemma wird diese Charakterisierung der Lösbarkeit des Syntheseproblems noch verfeinert, indem wir zeigen, welche zwei Gründe es haben kann, wenn $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ gilt.

Lemma 4.4.1

LEMMA 4.4.1
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge und sei (N, m_0) eine Schrittseparations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$. Es gilt genau dann $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$, wenn es eine falsche Schrittfortsetzung gibt, bzgl. der kein schrittseparierendes zulässiges Stellentripel existiert, oder wenn $PS(\mathcal{L})$ nicht schrittabgeschlossen ist.

BEWEIS: Die „genau dann"-Aussage lässt sich wie im Beweis des letzten Satzes nachweisen. Für $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ lassen sich also dieselben zwei Fälle unterscheiden. Im ersten Fall folgt wie im letzten Beweis, dass es eine falsche Schrittfortsetzung gibt, bzgl. der kein schrittseparierendes zulässiges Stellentripel existiert. Im zweiten Fall folgt auch wie im letzten Beweis, dass es ein bpo $\notin PS(\mathcal{L})$ gibt derart, dass $bpo' \in PS(\mathcal{L})$ für alle $bpo' \in Slin(bpo)$. Somit ist $PS(\mathcal{L})$ nicht schrittabgeschlossen.
Für die „wenn"-Aussage betrachten wir zuerst den Fall, dass es bzgl. einer falschen Schrittfortsetzung $\sigma = \sigma(bpo)$ kein schrittseparierendes zulässiges Stellentripel gibt. Dann folgt, dass σ in (N, m_0) aktiviert ist und somit $bpo \in \mathfrak{Bpo}(N, m_0)$. Nach Definition einer falschen Schrittfortsetzung folgt außerdem $bpo \notin PS(\mathcal{L})$ und somit $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$. Der zweite Fall ist, dass $PS(\mathcal{L})$ nicht schrittabgeschlossen ist, d.h. es gibt ein $bpo \notin PS(\mathcal{L})$ derart, dass $bpo' \in PS(\mathcal{L})$ für alle $bpo' \in Slin(bpo)$. Für eine Schrittseparations-Repräsentation gilt $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$. Damit folgt $bpo' \in \mathfrak{Bpo}(N, m_0)$ für alle $bpo' \in Slin(bpo)$ und somit $\sigma(bpo') \in \mathfrak{Step}(N, m_0)$ für alle $bpo' \in Slin(bpo)$. Dies bedeutet, dass $bpo \in \mathfrak{Bpo}(N, m_0)$. Somit folgt wieder $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$. □

Wichtig ist hier, dass auch in dem Fall, dass es für jede falsche Schrittfortsetzung ein schrittseparierendes zulässiges Stellentripel gibt, eine Schrittseparations-Repräsentation (N, m_0) nicht notwendigerweise eine positive Lösung für das Syntheseproblem darstellt. Dies ist auf den ersten Blick verwunderlich, da in diesem Fall alle falschen Schrittfortsetzungen nicht in (N, m_0) aktiviert sind. Es lässt sich insbesondere folgern, dass die Schrittsemantik von (N, m_0) mit dem durch $\mathcal{L}$ spezifizierten Schrittverhalten übereinstimmt.

LEMMA 4.4.2 *Lemma 4.4.2*
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge und sei (N, m_0) eine Schrittseparations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$ mit der Eigenschaft, dass (N, m_0) für jede falsche Schrittfortsetzung ein schrittseparierendes zulässiges Stellentripel enthält. Dann gilt $\mathfrak{Step}(N, m_0) = \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}$. Für das Ablaufverhalten lässt sich hiermit die Minimalitätseigenschaft $\mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m_0')$ für alle markierten S/T-Netze (N', m_0') mit $\mathcal{L} \subseteq \mathfrak{Bpo}(N', m_0')$ folgern.

BEWEIS: Die erste Aussage wird durch den ersten Fall im Beweis zu Satz 4.4.1 gezeigt. Aus dieser Aussage folgt $\mathfrak{Step}(N, m_0) \subseteq \mathfrak{Step}(N', m_0')$. Mit Definition 3.3.8 ergibt sich somit die zweite Aussage. □

Dennoch ist in diesem Fall nicht gewährleistet, dass die Ablaufsemantik von (N, m_0) mit dem durch $\mathcal{L}$ spezifizierten Ablaufverhalten übereinstimmt. Das Problem ist, dass es eine BPO $bpo \notin PS(\mathcal{L})$ geben kann, derart, dass $\sigma(bpo') \in \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\} = \mathfrak{Step}(N, m_0)$ für alle $bpo' \in Slin(\mathcal{L})$. Solch eine BPO ist bzgl. (N, m_0) aktiviert, aber eben nicht durch $\mathcal{L}$ spezifiziert. Allerdings ist dann wie in Lemma 4.4.1 bewiesen $PS(\mathcal{L})$ nicht schrittabgeschlossen.
Das Syntheseproblem lässt sich also nicht direkt durch die Betrachtung aller falscher Schrittfortsetzungen lösen. Aber wir haben in Satz 4.4.1 gezeigt, dass sich eine Schrittseparations-Repräsentation dennoch zur Lösung des Syntheseproblems eignet. Es ist nur noch ein Test erforderlich, welcher prüft, ob die Schrittseparations-Repräsentation das spezifizierte Verhalten aufweist oder nicht (hierzu kann insbesondere Lemma 4.4.1 hilfreich sein). Auf diesen Punkt wird im nächsten Abschnitt näher eingegangen.

Beispiel

BEISPIEL: *Das Netz aus Abbildung 57 ist eine Schrittseparations-Repräsentation für die partielle Sprache $\mathcal{L}'$ aus Abbildung 54. Dieses Netz hat nicht das Ablaufverhalten $\mathrm{PS}(\mathcal{L}')$. Dies liegt daran, dass es kein bzgl. der falschen Schrittfortsetzung $\mathtt{bcaa}$ schrittseparierendes zulässiges Stellentripel gibt. Somit lässt sich diese falsche Schrittfortsetzung (in einer Schrittseparations-Repräsentation) nicht verhindern. Das Syntheseproblem hat eine negative Antwort.*
Die partielle Sprache $\mathcal{L}''$ aus Abbildung 71 hat dieselben 71 falschen Schrittfortsetzungen wie die partielle Sprache aus Abbildung 4. Das Netz aus Abbildung 3 ist daher auch für diese partielle Sprache eine Schrittseparations-Repräsentation, welche alle falschen Schrittfortsetzungen verhindert (es genügt also Lemma 4.4.2). Aber dieses Netz weist dennoch nicht das durch die Sprache spezifizierte Ablaufverhalten auf. Dies liegt diesmal daran, dass $\mathrm{PS}(\mathcal{L}'')$ nicht schrittabgeschlossen ist. Die BPO $\mathrm{bpo}_2 \notin \mathrm{PS}(\mathcal{L}'')$ aus Abbildung 4 erfüllt $\mathrm{Slin}(\mathrm{bpo}_2) \subseteq \mathrm{Slin}(\mathrm{PS}(\mathcal{L}''))$, d.h. alle Schrittfolgen der BPO werden durch die betrachtete Sprache spezifiziert, nicht jedoch die BPO selbst. Es sind daher die Schrittfolgen von bpo_2 und damit automatisch auch die nicht-spezifizierte BPO bpo_2 (in einer Schrittseparations-Repräsentation) aktiviert. Das Syntheseproblem hat wiederum eine negative Antwort. Ein weiteres Beispiel für eine Sprache deren Präfix- und Sequentialisierungsabschluss nicht schrittabgeschlossen ist, ist in Abbildung 64 dargestellt.
Lassen sich für eine partielle Sprache wie die in Abbildung 4 alle falschen Schrittfortsetzungen mit zulässigen Stellen verhindern, so ist es plausibel, dass eine Schrittseparations-Repräsentation die gewünschte Schrittsemantik aufweist. Ist der Präfix- und Sequentialisierungsabschluss der Sprache wie in diesem Fall zusätzlich noch schrittabgeschlossen, so ist garantiert, dass eine Schrittseparations-Repräsentation auch die gewünschte Ablaufsemantik hat und somit das Syntheseproblem positiv löst.

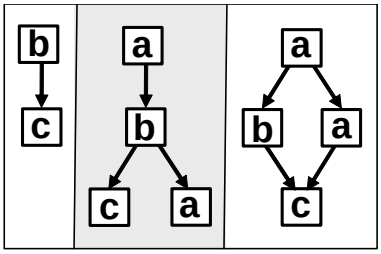

Abbildung 71: Eine partielle Sprache, deren Präfix- und Sequentialisierungsabschluss nicht schrittabgeschlossen ist.

In dem Rest dieses Unterabschnittes beschäftigen wir uns mit der Frage, wie eine Schrittseparations-Repräsentation effektiv berechnet werden kann. Hierzu entwickeln wir im Folgenden geeignete Verfahren zur Berechnung schrittseparierender zulässiger Stellentripel. Wir greifen dazu auf die verschiedenen Regionenkonzepte des letzten Abschnittes zurück. Um ein bzgl. einer falschen Schrittfortsetzung schrittseparierendes zulässiges Stellentripel auf der Regionenebene zu charakterisieren, definieren wir sog. schrittseparierende BPO-Transitions-Regionen und Markenfluss-Regionen. Eine solche Definition lässt sich für Transitions-Regionen leicht formulieren, da die Einträge einer Transitions-Region direkt den Komponenten des entsprechenden zulässigen Stellentripels entsprechen und sich daher die betrachtete Separierungseigenschaft für zulässige Stellentripel direkt auf die Ebene der Regionen übertragen

lässt. Für Markenfluss-Regionen benötigen wir noch die folgenden Überlegungen, um die Separierungseigenschaft für zulässige Stellentripel auf Eigenschaften von entsprechenden Markenflüssen zurückführen zu können.

- Für jede Transition $t \in T$ fixieren wir ein Beispielereignis ν_t einer BPO aus $\mathcal{L}$, welches mit t beschriftet ist.
- Außerdem fixieren wir das Quellenereignis ν_0 einer beliebigen BPO aus $\mathcal{L}$.

Der Markenfluss auf den eingehenden bzw. ausgehenden Kanten von ν_t repräsentiert dann den Multivorbereich bzw. den Multinachbereich von t bzgl. eines entsprechenden Stellentripels und der Markenfluss auf den ausgehenden Kanten von ν_0 repräsentiert die Anfangsmarkierung eines entsprechenden Stellentripels. Wir bezeichnen die Menge der in ν_t eingehenden Kanten mit E_t^{zu}, die Menge der aus ν_t ausgehenden Kanten mit E_t^{ab} und die Menge der aus ν_0 ausgehenden Kanten mit E_0.

Definition 4.4.4 (Schrittseparierende Region)

DEFINITION 4.4.4 (SCHRITTSEPARIERENDE REGION)
Sei $\sigma = \tau_1 \dots \tau_j \in \mathcal{L}_{ws}$ eine falsche Schrittfortsetzung von $\mathcal{L}$.

- *Eine bzgl. σ schrittseparierende BPO-Transitions-Region ist eine BPO-Transitions-Region $\mathbf{r} = (r_0, \dots, r_{2m}) \in \mathbb{N}^{2m+1}$ von $\mathcal{L}$ bzgl. $T = \{t_1, \dots, t_m\}$ mit der Eigenschaft*

$$\text{(TSS)} \qquad r_0 + \sum_{i=1}^{m} ((\tau_1 + \dots + \tau_{j-1})(t_i) \cdot r_i - (\tau_1 + \dots + \tau_j)(t_i) \cdot r_{m+i}) < 0.$$

- *Eine bzgl. σ schrittseparierende Markenfluss-Region ist eine Markenfluss-Region r von $\mathcal{L}$ mit der Eigenschaft*

$$\text{(MSS)} \qquad \sum_{e \in E_0} r(e) + \sum_{t \in T} \Big(\sum_{e \in E_t^{ab}} (\tau_1 + \dots + \tau_{j-1})(t) \cdot r(e) - \sum_{e \in E_t^{zu}} (\tau_1 + \dots + \tau_j)(t) \cdot r(e) \Big) < 0.$$

Lemma 4.4.3

LEMMA 4.4.3
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge. Für ein Stellentripel st und eine falsche Schrittfortsetzung σ von $\mathcal{L}$ sind die folgenden drei Aussagen äquivalent:

- *Das Stellentripel st ist ein bzgl. σ schrittseparierendes zulässiges Stellentripel.*
- *Es gibt eine bzgl. σ schrittseparierende BPO-Transitions-Region $\mathbf{r}$ von $\mathcal{L}$ bzgl. $T = \{t_1, \dots, t_m\}$, so dass $st = \overrightarrow{\mathbf{r}}$.*
- *Es gibt eine bzgl. σ schrittseparierende Markenfluss-Region r von $\mathcal{L}$, so dass $st = \overrightarrow{r}$.*

BEWEIS: Ähnlich wie im Beweis zu Lemma 4.3.5 lässt sich die Ungleichung (TSS) folgendermaßen umformen. (TSS) $\Longleftrightarrow m_{\overrightarrow{r}}(p_{\overrightarrow{r}}) + \sum_{i=1}^{m}((\tau_1+\ldots+\tau_{j-1})(t_i)\cdot W(t_i,p_{\overrightarrow{r}}) - (\tau_1+\ldots+\tau_j)(t_i)\cdot W(p_{\overrightarrow{r}},t_i)) < 0 \Longleftrightarrow m_{\overrightarrow{r}}(p_{\overrightarrow{r}}) + \sum_{i=1}^{m}((\tau_1+\ldots+\tau_{j-1})(t_i)\cdot W(t_i,p_{\overrightarrow{r}})) - \sum_{i=1}^{m}((\tau_1+\ldots+\tau_{j-1})(t_i)\cdot W(p_{\overrightarrow{r}},t_i)) < \sum_{i=1}^{m}(\tau_j(t_i)\cdot W(p_{\overrightarrow{r}},t_i)) \Longleftrightarrow m_{\overrightarrow{r}}(p_{\overrightarrow{r}}) + (\tau_1+\ldots+\tau_{j-1})^\circ(p_{\overrightarrow{r}}) - {}^\circ(\tau_1+\ldots+\tau_{j-1})(p_{\overrightarrow{r}}) < {}^\circ\tau_j(p_{\overrightarrow{r}})$. Unter Beachtung, dass für ein zulässiges Stellentripel st gilt, dass $\tau_1\ldots\tau_{j-1}$ in (N_{st}, m_{st}) aktiviert ist (vergleiche Lemma 4.3.4) folgt: Für eine BPO-Transitions-Region $\mathbf{r}$ ist die Eigenschaft (TSS) äquivalent dazu, dass $\sigma = \tau_1\ldots\tau_j$ entsprechend Definition 3.2.4 nicht bzgl. $(N_{\overrightarrow{r}}, m_{\overrightarrow{r}})$ aktiviert ist. Mit Satz 4.3.2 ergibt sich die Äquivalenz der ersten und zweiten Aussage.
Mit der folgenden Umformung der Ungleichung (MSS) lässt sich unter Verwendung von Satz 4.3.5 analog die Äquivalenz der ersten und dritten Aussage herleiten. (MSS) $\Longleftrightarrow Ab_r(\nu_0) + \sum_{t\in T}((\tau_1+\ldots+\tau_{j-1})(t)Ab_r(\nu_t) - (\tau_1+\ldots+\tau_j)(t)Zu_r(\nu_t)) < 0 \Longleftrightarrow m_{\overrightarrow{r}}(p_{\overrightarrow{r}}) + \sum_{t\in T}((\tau_1+\ldots+\tau_{j-1})(t)\cdot W(t,p_{\overrightarrow{r}}) - (\tau_1+\ldots+\tau_j)(t)\cdot W(p_{\overrightarrow{r}},t)) < 0 \Longleftrightarrow m_{\overrightarrow{r}}(p_{\overrightarrow{r}}) + \sum_{t\in T}((\tau_1+\ldots+\tau_{j-1})(t)\cdot W(t,p_{\overrightarrow{r}})) - \sum_{t\in T}((\tau_1+\ldots+\tau_{j-1})(t)\cdot W(p_{\overrightarrow{r}},t)) < \sum_{t\in T}(\tau_j(t)\cdot W(p_{\overrightarrow{r}},t)) \Longleftrightarrow m_{\overrightarrow{r}}(p_{\overrightarrow{r}}) + (\tau_1+\ldots+\tau_{j-1})^\circ(p_{\overrightarrow{r}}) - {}^\circ(\tau_1+\ldots+\tau_{j-1})(p_{\overrightarrow{r}}) < {}^\circ\tau_j(p_{\overrightarrow{r}})$. □

Wir betrachten nun eine endliche partielle Sprache. Im Rahmen einer Konstruktion einer Schrittseparations-Repräsentation soll für jede der endlich vielen falschen Schrittfortsetzungen entschieden werden, ob es ein entsprechendes schrittseparierendes zulässiges Stellentripel gibt und im positiven Fall ein solches berechnet werden. Entsprechend dem letzten Lemma lässt sich diese Fragestellung dadurch lösen, dass für jede falsche Schrittfortsetzung entschieden wird, ob es eine entsprechende schrittseparierende Region eines bestimmten Typs gibt, und im positiven Fall eine solche Region berechnet und das korrespondierende zulässige Stellentripel betrachtet wird. Sowohl für BPO-Transitions-Regionen als auch für Markenfluss-Regionen lässt sich dieses Problem auf die Suche nach ganzzahligen Lösungen von homogenen linearen Ungleichungssystemen reduzieren. Hierzu werden die linear algebraischen Charakterisierungen der beiden Regionentypen aus dem letzten Abschnitt verwendet.

Lemma 4.4.4

LEMMA 4.4.4
Sei $\mathcal{L}$ eine endliche partielle Sprache und $\sigma = \tau_1\ldots\tau_j \in \mathcal{L}_{ws}$ eine falsche Schrittfortsetzung von $\mathcal{L}$.

- *Sei die Beschriftungsmenge $T = \{t_1,\ldots,t_m\}$ geordnet. Die Menge der bzgl. σ schrittseparierenden BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1,\ldots,t_m\}$ entspricht der Menge der ganzzahligen Lösungen des homogenen linearen Ungleichungssystems*

$$\mathbf{A}_{\mathcal{L}}^{BPOT}\cdot\mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0},$$

$$\mathbf{b}_{\sigma}^{TSS}\cdot\mathbf{x} < 0.$$

Der Vektor $\mathbf{b}_{\sigma}^{TSS} = (b_{\sigma,0},\ldots,b_{\sigma,2m})$ ist gegeben durch

$$b_{\sigma,i} = \begin{cases} 1 & \text{für } i = 0, \\ (\tau_1+\ldots+\tau_{j-1})(t_i) & \text{für } i = 1,\ldots,m \\ -(\tau_1+\ldots+\tau_j)(t_{i-m}) & \text{für } i = m+1,\ldots,2m. \end{cases}$$

- *Sei eine Nummerierung der Menge* $\bigcup_{(V,<,l)\in\mathcal{L}} <^* = \{e_1,\ldots,e_n\}$ *der Kanten aller* $*$*-Erweiterungen der BPOs aus* $\mathcal{L}$ *gegeben. Die Menge der bzgl.* σ *schrittseparierenden Markenfluss-Regionen von* $\mathcal{L}$ *entspricht der Menge der zu ganzzahligen Lösungen des homogenen linearen Ungleichungssystems*

$$\mathbf{A}_{\mathcal{L}}^{MAR}\cdot\mathbf{x}=\mathbf{0},\mathbf{x}\geqslant\mathbf{0},$$
$$\mathbf{b}_{\sigma}^{MSS}\cdot\mathbf{x}<0$$

bzgl. $\{e_1,\ldots,e_n\}$ *korrespondierenden globalen Markenfluss-Funktionen. Der Vektor* $\mathbf{b}_{\sigma}^{MSS}=(b_{\sigma,0},\ldots,b_{\sigma,n})$ *ist gegeben durch*

$$b_{\sigma,i} = \begin{cases} 1 & \text{für } e_i\in E_0 \\ & \text{und } e_i\notin\bigcup_{t'\in T}E_{t'}^{zu}, \\ 1-(\tau_1+\ldots+\tau_j)(t') & \text{für } e_i\in E_0 \\ & \text{und } e_i\in E_{t'}^{zu}, \\ (\tau_1+\ldots+\tau_{j-1})(t) & \text{für } e_i\in E_t^{ab} \\ & \text{und } e_i\notin\bigcup_{t'\in T}E_{t'}^{zu}, \\ -(\tau_1+\ldots+\tau_j)(t') & \text{für } e_i\notin(\bigcup_{t\in T}E_t^{ab}\cup E_0) \\ & \text{und } e_i\in E_{t'}^{zu}, \\ (\tau_1+\ldots+\tau_{j-1})(t) & \text{für } e_i\in E_t^{ab} \\ -(\tau_1+\ldots+\tau_j)(t') & \text{und } e_i\in E_{t'}^{zu}, \\ 0 & \text{für } e_i\notin(\bigcup_{t\in T}E_t^{ab}\cup E_0) \\ & \text{und } e_i\notin\bigcup_{t'\in T}E_{t'}^{zu}. \end{cases}$$

Beweis: Der Zeilenvektor $\mathbf{b}_{\sigma}^{TSS}$ ist derart definiert, dass $\mathbf{b}_{\sigma}^{TSS}\cdot\mathbf{r}<0\Leftrightarrow$ (TSS). Daher folgt mit Lemma 4.3.8 die Behauptung für BPO-Transitions-Regionen.
Sei $\mathbf{r}\in\mathbb{N}^n$ und $r=r_{\mathbf{r}}$ die bzgl. $\{e_1,\ldots,e_n\}$ korrespondierende globale Markenfluss-Funktion von $\mathcal{L}$. Der Zeilenvektor $\mathbf{b}_{\sigma}^{MSS}$ ist derart definiert, dass $\mathbf{b}_{\sigma}^{MSS}\cdot\mathbf{r}<0\Leftrightarrow$ (MSS). Daher folgt mit Lemma 4.3.14 die Behauptung für Markenfluss-Regionen. □

Beispiel: *Wir betrachten die falsche Schrittfortsetzung* $\sigma=a+b$ *der partiellen Sprache aus Abbildung 4. Es gilt* $\mathbf{b}_{\sigma}^{TSS}=(1,0,0,0,-1,-1,0)$ *bzgl. der Transitionsreihenfolge* a,b,c*. Die BPO-Transitions-Region* $(1,2,0,0,1,1,0)$ *ist eine Lösung von* $\mathbf{A}_{\mathcal{L}}^{BPOT}\cdot\mathbf{x}\geqslant\mathbf{0},\mathbf{x}\geqslant\mathbf{0},\mathbf{b}_{\sigma}^{TSS}\cdot\mathbf{x}<0$*. Die in Abbildung 59 links dargestellte korrespondierende zulässige Stelle verhindert dementsprechend die Aktiviertheit von* σ*.*
Für die Kanten-Nummerierung aus Abbildung 68 sowie der Festlegung, dass v_0 *der Quellenknoten von* bpo_1*,* v_a *das erste* a*-Ereignis in* bpo_2*,* v_b *das* b*-Ereignis aus* bpo_1 *und* v_c *das* c*-Ereignis aus* bpo_1 *ist, ergibt sich* $\mathbf{b}_{\sigma}^{MSS}=(1,1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0)$*. Der zur Markenfluss-Region aus Abbildung 67 korrespondierende Vektor* $(0,0,1,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,2)$ *ist eine Lösung von* $\mathbf{A}_{\mathcal{L}}^{MAR}\cdot\mathbf{x}=\mathbf{0},\mathbf{x}\geqslant\mathbf{0},\mathbf{b}_{\sigma}^{MSS}\cdot\mathbf{x}<0$*. Die korrespondierende Stelle aus Abbildung 59 links verhindert die Aktiviertheit von* σ*.* *Beispiel*

Falls es also für eine falsche Schrittfortsetzung σ keine ganzzahlige Lösung des Ungleichungssystems $\mathbf{A}_{\mathcal{L}}^{BPOT}\cdot\mathbf{x}\geqslant\mathbf{0},\mathbf{x}\geqslant\mathbf{0},\mathbf{b}_{\sigma}^{TSS}\cdot\mathbf{x}<0$ bzw. $\mathbf{A}_{\mathcal{L}}^{MAR}\cdot\mathbf{x}=\mathbf{0},\mathbf{x}\geqslant\mathbf{0},\mathbf{b}_{\sigma}^{MSS}\cdot\mathbf{x}<0$ gibt, so gibt es kein bzgl. σ schrittseparierendes zulässiges Stellentripel. Falls es eine ganzzahlige Lösung des Ungleichungssystems gibt, so ist für jede solche Lösung

das korrespondierende Stellentripel ein bzgl. σ schrittseparierendes zulässiges Stellentripel.
Aufgrund der „< 0"-Restriktionen definieren beide Ungleichungssysteme in dieser Form kein Polyeder. Da aber alle Ungleichungen nur ganzzahlige Koeffizienten besitzen und wir nur an ganzzahligen Lösungen interessiert sind, können wir äquivalent die Ungleichungssysteme $\mathbf{A}_{\mathcal{L}}^{BPOT} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, -\mathbf{b}_{\sigma}^{TSS} \cdot \mathbf{x} \geqslant 1$ bzw. $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, -\mathbf{b}_{\sigma}^{MSS} \cdot \mathbf{x} \geqslant 1$ betrachten. Die Mengen der ganzzahligen Lösungen dieser Ungleichungssysteme stimmen mit den Mengen der ganzzahligen Lösungen der ursprünglichen Ungleichungssysteme überein. Trotzdem soll erst einmal die Ganzzahligkeitsrestriktion ignoriert werden. Die beiden Ungleichungssysteme definieren nämlich in der neuen Form ein rationales Polyeder (aber aufgrund der „$\geqslant 1$"-Restriktionen definieren sie keinen Kegel). Daher können wir die verschiedenen Lösungsverfahren für das lineare Zulässigkeitsproblem aus Abschnitt 3.4 auf diese Ungleichungssysteme anwenden. Wichtig ist nun, dass beide Ungleichungssysteme die Form $\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{b}$ für ein $\mathbf{b} \geqslant \mathbf{0}$ haben (hier wäre auch eine Argumentation über die Homogenität der ursprünglichen Ungleichungssysteme möglich). Daher lässt sich, wie in Abschnitt 3.4 beschrieben, das für uns interessante ganzzahlige Zulässigkeitsproblem in diesen Fällen auf des normale lineare Zulässigkeitsproblem zurückführen (Multiplikation eines rationalen Lösungsvektors mit dem Hauptnenner der Einträge des Lösungsvektors ergibt einen ganzzahligen Lösungsvektor). Das bedeutet, dass wir mit einem Lösungsverfahren für das lineare Zulässigkeitsproblem, z.B. dem Simplex-Algorithmus, entscheiden können, ob das Ungleichungssystem $\mathbf{A}_{\mathcal{L}}^{BPOT} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, -\mathbf{b}_{\sigma}^{TSS} \cdot \mathbf{x} \geqslant 1$ bzw. $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, -\mathbf{b}_{\sigma}^{MSS} \cdot \mathbf{x} \geqslant 1$ eine ganzzahlige Lösung besitzt, und im positiven Fall auch eine solche berechnen können.

Beispiel

BEISPIEL: *Wir betrachten wiederum die partielle Sprache aus Abbildung 4. Im Rahmen der Berechnung einer bzgl. der falschen Schrittfortsetzung $\sigma = a + b$ schrittseparierenden BPO-Transitions-Region ergibt sich das Ungleichungssystem aus Abbildung 72.*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 1 & 0 & -2 & -1 & -1 \\ 1 & 1 & 0 & 0 & -2 & -1 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \bullet \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Abbildung 72: Ungleichungssystem $\mathbf{A}_{\mathcal{L}}^{BPOT} \cdot \mathbf{x} \geqslant \mathbf{0}, -\mathbf{b}_{\sigma}^{TSS} \cdot \mathbf{x} \geqslant 1$.

Damit ergibt sich der folgende Algorithmus zur Konstruktion einer Schrittseparations-Repräsentation ausgehend von einer partiellen Sprache. Zuerst wird ein Regionentyp gewählt und die entsprechende Matrix konstruiert (im Prinzip können die Regionentypen auch gemischt verwendet werden). Dann werden alle falschen Schrittfortsetzungen konstruiert. Außerdem wird ein Ausgangsnetz bestehend aus der durch die Beschriftungsmenge der partiellen Sprache gegebenen Transitionsmenge und leerer Stellenmenge festgelegt. Anschließend

wird die Menge der falschen Schrittfortsetzungen in einer beliebigen Reihenfolge abgearbeitet. Dabei wird für jede falsche Schrittfortsetzung das bzgl. des gewählten Regionentyps entsprechende Ungleichungssystem betrachtet und entschieden, ob es eine ganzzahlige Lösung des Ungleichungssystems gibt. Falls es eine Lösung gibt, wird eine solche Lösung berechnet und das korrespondierende zulässige Stellentripel zum Ausgangsnetz hinzugefügt.
Dieses Verfahren erzeugt offenbar eine Schrittseparations-Repräsentation. Nun ist uns einerseits natürlich die Effizienz des Algorithmus zur Konstruktion einer Schrittseparations-Repräsentation wichtig. Andererseits sind wir, wie in Abschnitt 4.1 diskutiert, auch an der Konstruktion einer möglichst kompakten Schrittseparations-Repräsentation interessiert. Während die Anzahl der Transitionen aller Schrittseparations-Repräsentationen identisch ist, soll eine Schrittseparations-Repräsentation mit möglichst wenig Stellen, möglichst wenig Kanten und möglichst kleinen Kantengewichten sowie einer möglichst kleinen Anfangsmarkierung berechnet werden. Wie schon angesprochen, ist ein solches Netz für eine effiziente automatische Weiterverarbeitung sehr vorteilhaft. Dies betrifft sowohl den Vergleichstest des nächsten Abschnittes, als auch weiterführende auf dem synthetisierten Netz basierende Simulations-, Analyse- und Optimierungsverfahren, welche in verschiedenen Anwendungskontexten von Synthese von Bedeutung sind. Daneben ist ein kompaktes Netz auch im Rahmen einer sowohl in der wissenschaftlichen als auch in der industriellen Praxis häufig so bedeutsamen manuellen Inspektion des Netzes wichtig. Einerseits werden Analysen, Optimierungen und Feinabstimmungen von Netzen in vielen Fällen per Hand durchgeführt und andererseits müssen Netze oft auch in der Modellierung unerfahrenen Personen, wie Managern, vermittelt werden. Hierfür sind gut lesbare Netze entscheidend, welche schnell verstanden und interpretiert werden können.
Daher schlagen wir die folgende naheliegende Verbesserung des ersten Vorschlages zur Konstruktion einer Schrittseparations-Repräsentation vor. Die Grundüberlegung ist hier, dass ein zulässiges Stellentripel häufig sehr viele falsche Schrittfortsetzungen verhindert. Falls also eine falsche Schrittfortsetzung von einem in einem vorherigen Schritt zum Netz hinzugefügten zulässigen Stellentripel verhindert wird, so muss für diese falsche Schrittfortsetzung kein weiteres zulässiges Stellentripel mehr gesucht und hinzugefügt werden, da das Netz ohnehin schon eine geeignete Stelle enthält. Um dies zu überprüfen, muss nur getestet werden, ob die falsche Schrittfortsetzung in dem bisher konstruierten Netz aktiviert ist. Ist dies nicht der Fall, so kann auf die Betrachtung des entsprechenden Ungleichungssystems, auf das relativ aufwändige Lösen des Zulässigkeitsproblems für das Ungleichungssystem und auf das Hinzufügen eines entsprechenden Stellentripels verzichtet werden. Dieses Vorgehen macht den Konstruktionsalgorithmus meist sehr viel effizienter. Typischerweise kann dadurch die Anzahl der zu lösenden Zulässigkeitsprobleme sehr stark gesenkt werden, während der zusätzliche Aufwand der Tests auf Aktiviertheit der falschen Schrittfortsetzungen sehr gering ist. Außerdem wird die Anzahl der Stellentripel, welche zu dem Netz hinzugefügt werden, auch entsprechend geringer. Somit kann normalerweise auch die Größe der konstruierten Schrittseparations-Repräsentationen deutlich reduziert werden. Mit diesen Überlegungen ergibt sich nun in Pseudo-Code der folgende Algorithmus zur Konstruktion einer Schrittseparations-Repräsentation.

Algorithmus 4.4.1 (Berechnung Schrittseparations-Repräsentation)

ALGORITHMUS 4.4.1 (BERECHNUNG SCHRITTSEPARATIONS-REPRÄSENTATION)

```
1   Eingabe: Partielle Sprache ℒ
2   Ausgabe: Schrittseparations−Repräsentation von ℒ
3   BEGIN
4       A_ℒ^BPOT bzw. A_ℒ^MAR :=
            ℒ.regionenMatrix(BPOT bzw. MAR);
5       ℒ_ws := ℒ.falscheSchrittFortsetzungen();
6       (N, m_0) := (∅, ℒ.beschriftungsMenge(), ∅, ∅);
7       FORALL σ ∈ ℒ_ws DO
8           IF σ.istAktiviertIn(N, m_0) THEN
9               r := ganzzahligeLoesung
                   (A_ℒ^BPOT · x ⩾ 0, x ⩾ 0, b_σ^TSS · x < 0 bzw.
                   A_ℒ^MAR · x = 0, x ⩾ 0, b_σ^MSS · x < 0);
10              IF r ≠ null THEN
11                  (N, m_0).fuegeHinzu(r⃗);
12              ENDIF
13          ENDIF
14      ENDFORALL
15      RETURN (N, m_0);
16  END
```

Die einzelnen Zulässigkeitsprobleme des Algorithmus können in polynomieller Zeit in Abhängigkeit von der Größe des jeweiligen Ungleichungssystems gelöst werden. Hier spielt also die in Abschnitt 4.3 diskutierte Größe der entsprechenden Matrizen eine Rolle. Problematisch an dem Algorithmus ist, dass die Anzahl der falschen Schrittfortsetzungen exponentiell in der Größe der gegebenen partiellen Sprache $\mathcal{L}$ sein kann. Im schlechtesten Fall kann die Anzahl der falschen Schrittfortsetzungen $2 * |T| * |\{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}|$ entsprechen, wobei zu beachten ist, dass die Anzahl der Schrittlinearisierungen einer BPO exponentiell in der Größe der BPO sein kann (siehe Abbildung 62). Somit kann es vorkommen, dass sowohl die Anzahl der Zulässigkeitsprobleme, welche gelöst werden müssen, als dementsprechend auch die Anzahl der Stellentripel, welche zu dem konstruierten Netz hinzugefügt werden, exponentiell in der Größe der Eingabe $\mathcal{L}$ sind. Typischerweise sind diese Anzahlen aber wesentlich kleiner, da der Test, ob eine falsche Schrittfortsetzung in dem bisher konstruierten Netz aktiviert ist, häufig negativ ausfällt.

Beispiel

BEISPIEL: *Wir haben schon diskutiert, dass die vier Stellen des Netzes aus Abbildung 3 alle 71 falschen Fortsetzungen der Sprache aus Abbildung 4 verhindern. Ergeben sich im Rahmen von Algorithmus 4.4.1 beim Lösen der ersten vier zu lösenden Ungleichungssysteme also gerade diese vier Stellen, so entsteht eben dieses Netz mit nur vier Stellen als Endergebnis. In diesem Fall werden dann trotz der 71 falschen Fortsetzungen auch nur 4 Zulässigkeitsprobleme betrachtet, d.h. der Aktiviertheitstest* σ.istAktiviertIn(N, m_0) *in Zeile 8 ist nur in 4 der 71 Fälle positiv.*
Es ist aber auch möglich, dass vollkommen andere Stellen erzeugt werden, diese sind aber jeweils zumindest für die betrachtete falsche Fortsetzung separierend. Eine solche Stelle, welche eine falsche Fortsetzung separiert, weist eine gewisse Restriktivität auf und verhindert daher meist wie in unserem Beispiel auch weitere falsche Fortsetzungen.
Es ist auch möglich, dass es für eine falsche Fortsetzung keine Lösung des entsprechenden Ungleichungssystems und damit auch keine für die falsche

Fortsetzung geeignete separierende Stelle gibt (Fall $\mathbf{r} = \mathtt{null}$). Ein Beispiel hierfür ist die falsche Schrittfortsetzung $\mathtt{bcaa}$ *der partiellen Sprache aus Abbildung 54. In diesem Fall wird natürlich dann keine Stelle zum Netz hinzugefügt und direkt die nächste falsche Fortsetzung bearbeitet.*

Das Einbeziehen des Aktiviertheitstests in Zeile 8 stellt, wie beschrieben, schon eine zentrale und wichtige Verbesserung des ursprünglichen Vorschlages für einen Algorithmus zur Konstruktion einer Schrittseparations-Repräsentation dar. Darüber hinaus gibt es noch etliche weitere, meist etwas kompliziertere, Möglichkeiten, den betrachteten Algorithmus im Hinblick auf die Laufzeit und auf die Größe der konstruierten Schrittseparations-Repräsentation zu optimieren. Einige solche Möglichkeiten sollen in folgender Bemerkung diskutiert werden.

BEMERKUNG 4.4.1 (OPTIMIERUNGSVORSCHLÄGE)
Die folgenden Optimierungen von Algorithmus 4.4.1 sind möglich.

Bemerkung 4.4.1 (Optimierungsvorschläge)

- *Eine erste Möglichkeit, die Effizienz des Algorithmus zu verbessern, ist im Falle, dass es für eine falsche Schrittfortsetzung kein separierendes zulässiges Stellentripel gibt, den Algorithmus sofort abzubrechen. Entsprechend Lemma 4.4.1 kann in diesem Fall unmittelbar eine negative Antwort auf das Syntheseproblem gegeben werden und somit ist insbesondere kein nachfolgender Übereinstimmungstest mehr erforderlich. Das Syntheseproblem lässt sich also mit diesem Vorgehen in manchen Fällen sehr schnell lösen, allerdings wird durch den Abbruch keine vollständige Schrittseparations-Repräsentation mehr berechnet.*

- *Der zweite Optimierungsvorschlag zielt darauf ab, die Anzahl der zu betrachtenden falschen Schrittfortsetzungen zu reduzieren. Hier sind ähnliche Verfahren wie in Bemerkung 4.3.1 zur Reduktion der Anzahl der Zeilen von $\mathbf{A}^{ST}_{\mathcal{L}}$ von Interesse. Zuerst einmal bietet sich wieder die Möglichkeit, auf der Ebene der falschen Schrittfortsetzungen nach Präfixen und Sequentialisierungen zu suchen. Gilt für zwei falsche Schrittfortsetzungen $\sigma = \sigma(\mathrm{bpo})$ und $\sigma' = \sigma(\mathrm{bpo}')$, dass $\mathrm{bpo} \in \mathrm{PS}(\mathrm{bpo}')$, so ist offensichtlich jedes bzgl. σ schrittseparierende zulässige Stellentripel auch bzgl. σ' schrittseparierend. Dasselbe trifft aber auch auf den Fall von zwei falschen Schrittfortsetzungen $\sigma = \tau_1 \ldots \tau_j$ und $\sigma' = \tau'_1 \ldots \tau'_k$ mit $\sum_{i=1}^{j-1} \tau_i = \sum_{i=1}^{k-1} \tau'_i$ und $\tau_j = \tau'_k$ (ähnlich wie in [63, 18, 65]) oder zumindest $\tau_j \leqslant \tau'_k$ zu. Daher muss in beiden Situationen zu Synthesezwecken σ' nicht als falsche Schrittfortsetzung betrachtet werden. Zwar wird dann in dem Fall, dass es bzgl. σ kein schrittseparierendes zulässige Stellentripel gibt, es jedoch bzgl. σ' eines gibt, nicht mehr notwendigerweise eine vollständige Schrittseparations-Repräsentation berechnet. Aber in diesem Fall kann ohnehin eine negative Antwort auf das Syntheseproblem gegeben werden. Zu entscheiden, welche falschen Schrittfortsetzungen nach diesem Prinzip weggelassen werden können, kostet natürlich zusätzlichen Aufwand. Ob sich dieser Aufwand lohnt ist fraglich, da, falls σ zuerst separiert wird, die falsche Schrittfortsetzung σ' in dem bisher konstruierten Netz nicht aktiviert ist. Damit ist dann für σ' nur der einfache Aktiviertheitstest aus Zeile 8 nötig.*

- *Im Allgemeinen kann die Menge der falschen Schrittfortsetzungen auch nach dem letzten Optimierungsvorschlag noch viele falsche Schrittfortsetzungen σ' enthalten, für welche gilt, dass es eine andere falsche Schrittfortsetzung σ gibt, so dass jedes bzgl. σ schrittseparierende zulässige Stellentripel auch bzgl. σ' schrittseparierend ist. Wie im letzten*

*Absatz argumentiert, können solche falschen Schrittfortsetzungen σ' weggelassen werden. Mit den Resultaten dieses Unterabschnittes lässt sich folgern, dass dieser Zusammenhang aus einem linear algebraischen Blickwinkel heraus bedeutet, dass in $\mathbf{A}_{\mathcal{L}}^{\mathrm{BPOT}} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, \mathbf{b}_{\sigma}^{\mathrm{BPOSS}} \cdot \mathbf{x} < 0, \mathbf{b}_{\sigma'}^{\mathrm{BPOSS}} \cdot \mathbf{x} < 0$ bzw. $\mathbf{A}_{\mathcal{L}}^{\mathrm{MAR}} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, \mathbf{b}_{\sigma}^{\mathrm{MSS}} \cdot \mathbf{x} < 0, \mathbf{b}_{\sigma'}^{\mathrm{MSS}} \cdot \mathbf{x} < 0$ die $\mathbf{b}_{\sigma'}$-Ungleichung jeweils redundant ist. Dementsprechend können solche falschen Schrittfortsetzungen σ' durch entsprechendes Suchen nach redundanten Ungleichungen gefunden werden. Wie in Abschnitt 3.4 beschrieben, gibt es generell verschiedene Verfahren mit verschieden hohem rechnerischen Aufwand zum Auffinden redundanter Ungleichungen. In der beschriebenen Situation lassen sich aber auch spezielle vereinfachte Verfahren sinnvoll einsetzen, z.B. den Ungleichungsteil mit der **A**-Matrix erst einmal wegzulassen. Neben dem vorausgegangenen Optimierungsvorschlag sind auch weitere direkt auf falschen Schrittfortsetzungen basierende Verfahren, also Verfahren auf der Ebene der Schrittfolgen, denkbar, um die Menge der falschen Schrittfortsetzungen zu reduzieren. Allerdings ist eine entsprechende Argumentation für Schrittfolgen meist kompliziert. Der falsche Fortsetzungsbegriff des nächsten Unterabschnittes eignet sich für solche Zwecke besser.*

- *Einen wesentlichen Einfluss auf die Effizienz des Algorithmus hat die Größe der zu lösenden Ungleichungssysteme. Daher ist es für die Laufzeit wichtig, eine Regionendefinition mit einer möglichst kleinen entsprechenden **A**-Matrix zu wählen. Darüber hinaus können auch noch Optimierungsverfahren aus Abschnitt 4.3 genutzt werden, um die Größe der betrachteten **A**-Matrix so weit wie möglich zu reduzieren.*

- *Daneben spielt natürlich auch das verwendete Verfahren zur Lösung der Zulässigkeitsprobleme für die betrachteten Ungleichungssysteme eine entscheidende Rolle für die Laufzeit des Algorithmus. Mögliche Lösungsverfahren wurden in Abschnitt 3.4 diskutiert. Es ist schwer zu sagen, welches Lösungsverfahren sich für unsere Zwecke am besten eignet. Hier können allgemeine Erfahrungswerte über die verschiedenen Verfahren zu Rate gezogen werden, es ist aber auch sinnvoll, spezielle Tests durchzuführen. Eine nennenswerte Möglichkeit ist hier auch die Verwendung inkrementeller Lösungsverfahren. Die zu lösenden Ungleichungssysteme unterschieden sich alle nur in der **b**-Ungleichung. In einem solchen Fall ist es möglich, nur einmal eine Lösung des restlichen Ungleichungssystems zu berechnen. Dann können spezielle Verfahren angewandt werden, um jeweils ausgehend von dieser Lösung die im Algorithmus vorkommenden Ungleichungssystems mit einer einzigen zusätzlichen **b**-Ungleichung zu lösen.*

- *Neben der Laufzeit des verwendeten Lösungsverfahren hat das Lösungsverfahren auch noch einen weiteren wichtigen Einfluss auf den Algorithmus. Im Falle, dass es für eine falsche Schrittfortsetzung ein schrittseparierendes zulässiges Stellentripel gibt, so wird entsprechend dem Lösungsverfahren ein solches berechnet. Welches Stellentripel berechnet wird, beeinflusst das weitere Verhalten des Algorithmus. Dies liegt daran, dass der Test, ob eine falsche Schrittfortsetzung in dem bisher konstruierten Netz aktiviert ist, von den bisher berechneten Stellentripeln abhängt. Werden also Stellentripel berechnet, welche möglichst viele weitere falsche Schrittfortsetzungen verhindern, so müssen entsprechend weniger Zulässigkeitsprobleme gelöst und Stellentripel zu dem Netz hinzugefügt werden. Dies hat also einen positiven Einfluss*

sowohl auf die Laufzeit des Algorithmus, als auch auf die Größe des berechneten Netzes. Es lässt sich allerdings nicht ohne Weiteres einschätzen, welche Lösungsverfahren, welche Art von Stellentripeln erzeugen. Aber es ist hier möglich, durch das Betrachten geeigneter Zielfunktionen die berechnete Lösung des Ungleichungssystems derart zu leiten, dass besonders „gute" Stellentripel berechnet werden (vgl. Abschnitt 5.4). Allerdings sind dazu aufwändige ganzzahlige lineare Optimierungsverfahren (vgl. Abschnitt 3.4) nötig. Im Rahmen eines solchen Vorgehens kann zur Erzeugung gut lesbarer Netze auch eine Zielfunktion betrachtet werden, welche Stellentripel mit möglichst kleinen Kantengewichten und Anfangsmarkierungen bevorzugt.

- *Auch die Reihenfolge, in der die falschen Schrittfortsetzungen abgearbeitet werden, hat einen Einfluss darauf, wie viele Zulässigkeitsprobleme gelöst und wie viele Stellentripel zu dem Netz hinzugefügt werden. Dies lässt sich folgendermaßen erklären. Sei ein für eine falsche Schrittfortsetzung σ berechnetes Stellentripel auch separierend bzgl. einer anderen falschen Schrittfortsetzung σ'. Falls nun σ in der betrachteten Reihenfolge vor σ' abgearbeitet wird, so ist der Aktiviertheitstest für σ' in Zeile 8 negativ und es wird direkt die nächste falsche Schrittfortsetzung betrachtet. Hierfür ist also entscheidend, dass σ vor σ' geordnet ist. Dementsprechend sind Verfahren und Heuristiken von Interesse, welche eine in diesem Sinne möglichst effiziente Reihenfolge der falschen Schrittfortsetzungen berechnen. Auch der Zeitpunkt eines Abbruches des Algorithmus entsprechend dem ersten Optimierungsvorschlag hängt natürlich von dieser Reihenfolge ab.*

- *Sei im Beispiel des letzten Absatzes σ' vor σ geordnet. Dann kann es vorkommen, dass ein bzgl. σ' separierendes zulässiges Stellentripel zum Netz hinzugefügt wird und zusätzlich das Stellentripel für σ hinzugefügt wird, welches auch bzgl. σ' separierend ist. Hier kann es sinnvoll sein, nach solchen Situationen zu suchen, um das Stellentripel für σ' wieder zu entfernen und somit die Anzahl der Stellen des Netzes zu reduzieren. Dies ist aber nicht möglich, falls dieses Stellentripel aber kein anderes betrachtetes Stellentripel falsche Schrittfortsetzungen verhindert, welche zwischen σ' und σ abgearbeitet werden. Ein ähnliches Vorgehen, welches immer möglich ist, ist, in dem konstruierten Netz nach impliziten Stellen zu suchen und solche zu entfernen. In der Literatur gibt es etliche verschieden aufwändige Verfahren, um implizite Stellen zu finden. Einige interessante Verfahren werden anschließend in Bemerkung 4.4.2 diskutiert. Es ist natürlich auch möglich, nicht „on the fly", sondern erst nach Abarbeitung des gesamten Algorithmus nach impliziten Stellen zu suchen, um die Netzgröße im nachhinein so weit wie möglich zu reduzieren. Allerdings beschleunigt eine „on the fly"-Reduktion der Stellen die Aktiviertheitstests in Zeile 8. Neben dem Entfernen von impliziten Stellen kann auch versucht werden, mehrere synthetisierte Stellen derart durch eine neue Stelle zu ersetzen, dass sich das Ablaufverhalten des Netzes dabei nicht ändert. Siehe auch hierzu Bemerkung 4.4.2.*

Beispiel

BEISPIEL: *Wir illustrieren den zweiten Optimierungsvorschlag anhand der partiellen Sprache aus Abbildung 4. Die falsche Schrittfortsetzung ac stellt eine Sequentialisierung der falschen Schrittfortsetzung $a + c$ und ein Präfix der falschen Schrittfortsetzung $ab + c$ dar. Daher bewirkt eine Separierung der ersten falschen Schrittfortsetzung auch eine Separierung der letzteren beiden. Dementsprechend ist es nicht notwendig, diese zu betrachten.*

Für die falschen Schrittfortsetzungen $abaa$ *und* $aaba$ *bzw. auch* $abaa$ *und* $aaba + c$ *trifft der in dem Optimierungsvorschlag als zweites beschriebene Fall zu. Die jeweils zweite falsche Schrittfortsetzung kann also vernachlässigt werden.*

Bemerkung 4.4.2 (Suchen impliziter Stellen)

Bemerkung 4.4.2 (Suchen impliziter Stellen)
Entsprechend Definition 3.2.6 ist eine Stelle $p \in P$ *eines Netzes* (N, m_0), $N = (P, T, W)$ *implizit, wenn sich durch Weglassen des zugehörigen atomaren Teilnetzes das Ablaufverhalten bzw. äquivalent die Schrittsemantik des Netzes nicht verändert. Es kann also geprüft werden, ob eine Stelle implizit ist, indem sie entsprechend entfernt wird und die Ablaufsemantik oder die Schrittsemantik des neuen Netzes mit der des ursprünglichen Netzes verglichen wird.*
Alternativ kann entsprechend Satz 4.4.1 auch getestet werden, ob nach dem Weglassen der Stelle noch alle falschen Schrittfortsetzungen von $\mathfrak{Bpo}(N, m_0)$ *ausgeschlossen sind, d.h. ob das neue Netz noch immer eine Schrittseparations-Repräsentation bzgl.* $\mathfrak{Bpo}(N, m_0)$ *ist. Im positiven Fall ist die Stelle dann implizit. Im Rahmen der Synthese bietet es sich hier auch an, die ohnehin schon berechnete Menge der falschen Schrittfortsetzungen der gegebenen partiellen Sprache* $\mathcal{L}$ *anstelle von* $\mathfrak{Bpo}(N, m_0)$ *zu betrachten, d.h. zu überprüfen, ob das durch das Weglassen der Stelle resultierende Netz noch eine Schrittseparations-Repräsentation bzgl.* $\mathcal{L}$ *ist. Im positiven Fall kann die Stelle im Hinblick auf die Synthese weggelassen werden, sie ist jedoch nicht notwendigerweise implizit.*
Die genannten Vorgehensweisen sind allerdings sehr aufwändig. Daher sind wir an effizient überprüfbaren hinreichenden Bedingungen für implizite Stellen interessiert. Offensichtlich erzeugt beispielsweise eine Stelle $p \in P$ *mit* $W(p,t) = 0$ *für alle* $t \in T$ *keine Verhaltenseinschränkung und ist somit implizit. Diese Bedingung lässt sich für eine Stelle unmittelbar prüfen.*
Weiter ist für zwei Stellen $p, p' \in P$ *und ein positives reelles* λ *mit* $\lambda \cdot m_0(p) \geqslant m_0(p')$ *und* $\lambda \cdot W(t,p) \geqslant W(t,p')$ *sowie* $\lambda \cdot W(p,t) \leqslant W(p',t)$ *für alle* $t \in T$ *die Stelle* p *weniger restriktiv als die Stelle* p'*. Es lässt sich leicht einsehen, dass in diesem Fall* p *implizit ist. Die Bedingung lässt sich für ein Paar von Stellen leicht überprüfen.*
Das im weiteren Verlauf dieses Abschnittes zu beweisende Lemma 4.4.9 zeigt, dass für paarweise verschiedene $p, p_1, \ldots, p_k \in P$, $k \in \mathbb{N}$, *und nichtnegativen reelle Zahlen* $\lambda_1, \ldots, \lambda_k$ *mit* $m_0(p) = \sum_{i=1}^{k} \lambda_i \cdot m_0(p_i)$, $W(p,t) = \sum_{i=1}^{k} \lambda_i \cdot W(p_i,t)$ *und* $W(t,p) = \sum_{i=1}^{k} \lambda_i \cdot W(t,p_i)$ *für alle* $t \in T$ *die Stelle* p *implizit ist. Entsprechend den Überlegungen des letzten Absatzes gilt gleiches auch im Falle* $m_0(p) \geqslant \sum_{i=1}^{k} \lambda_i \cdot m_0(p_i)$, $W(p,t) \leqslant \sum_{i=1}^{k} \lambda_i \cdot W(p_i,t)$ *und* $W(t,p) \geqslant \sum_{i=1}^{k} \lambda_i \cdot W(t,p_i)$ *für alle* $t \in T$*. Ob eine Stelle* p *nach diesem Kriterium implizit ist, lässt sich durch das Lösen eines linearen Zulässigkeitsproblems bestimmen. Hierzu wird* $\{p_1, \ldots, p_k\} = P \setminus \{p\}$ *gesetzt und* $(\lambda_1, \ldots, \lambda_k)$ *als Variablenvektor interpretiert.*
Interessante Ansätze zum Entfernen impliziter Stellen finden sich auch in einigen Arbeiten in der Literatur, z.B. [39, 57, 93]. Es sei hier noch angemerkt, dass es auch sinnvoll sein kann, eine Stelle zu einem Netz hinzuzufügen, so dass mehrere andere Stellen dadurch implizit werden und somit entfernt werden können. Dies lässt sich, ähnlich wie in [194] dargestellt, durchführen. Zuerst werden zwei Stellen weggelassen. Dann wird geprüft, welche falschen Schrittfortsetzungen von $\mathfrak{Bpo}(N, m_0)$ *nun nicht mehr ausgeschlossen sind. Anschließend wird versucht, all diese falschen Schrittfortsetzungen durch ein einzelnes zulässiges Stellentripel auszuschließen und eine entsprechende Stelle zum Netz hinzuzufügen. Dies lässt sich analog wie im Falle einer einzelnen*

falschen Schrittfortsetzung durch das Lösen eines Ungleichungssystems realisieren. Im positiven Fall werden zwei Stellen durch eine ersetzt, ohne dass sich das Verhalten des Netzes ändert. Im negativen Fall wird das Netz nicht verändert. Durch ein iteratives Betrachten von Stellen-Paaren lässt sich auf diese Weise die Anzahl der Stellen eines Netzes reduzieren. Es sei noch angemerkt, dass es sich bei diesem Vorgehen im Rahmen der Synthese wiederum anbietet, die ohnehin schon berechnete Menge der falschen Schrittfortsetzungen der gegebenen partiellen Sprache $\mathcal{L}$ anstelle von $\mathfrak{Bpo}(N, m_0)$ zu betrachten.

Im schlechtesten Fall bleibt auch bei der Verwendung aller vorgestellter Optimierungen die exponentielle Abhängigkeit der Anzahl der Zulässigkeitsprobleme, welche gelöst werden müssen, als dementsprechend auch der Anzahl der Stellentripel, welche zu dem konstruierten Netz hinzugefügt werden, von der Größe der Eingabe $\mathcal{L}$ bestehen. Dies liegt an der exponentiellen Anzahl von falschen Schrittfortsetzungen, welche sich in einigen Fällen aufgrund einer exponentiellen Anzahl von Schrittlinearisierungen in der Größe einer BPO ergibt. Daher wird im nächsten Abschnitt ein verbesserter Begriff einer falschen Fortsetzung eingeführt, welcher nicht mehr auf Schrittfolgen basiert.

4.4.2 *BPO-Separation*

In diesem Unterabschnitt wird ein weiteres auf der im letzten Unterabschnitt erläuterten Idee der falschen Fortsetzungen basierendes Berechnungsverfahren für Regionen vorgestellt (siehe auch [17, 63, 18, 65]). Während im letzten Unterabschnitt falsche Fortsetzungen in der Form von Schrittfolgen verwendet wurden, wird nun eine fortschrittlichere direkt auf der Struktur der partiellen Sprache basierende Definition von falschen Fortsetzungen eingeführt. Die Vorteile dieses Ansatzes sind ähnlich wie die Vorteile von BPO-Transitions-Regionen gegenüber Schritt-Transitions-Regionen und lassen sich an ähnlichen Beispielen illustrieren.
Der zweite Optimierungsvorschlag der letzten Bemerkung zeigt, dass es auch im Rahmen von falschen Schrittfortsetzungen genügt, Präfixschritte $\pi = \sum_{i=1}^{j-1} \tau_i$ zusammen mit darauf folgenden Schritten $\tau = \tau_j$ zu betrachten. Daneben wird deutlich, dass das in der Konstruktionsvorschrift für falsche Schrittfortsetzungen vorkommende Ergänzen um einzelne Transitionen vielfach zu vernachlässigbaren falschen Fortsetzungen führt. Insbesondere ergeben sich Präfix- und Sequentialisierungsbeziehungen. Hier ist es sinnvoll, für ein Präfix π nur minimale Schritte τ zu betrachten. Ähnlich wie in Unterabschnitt 4.3.2 ist es sogar wieder möglich, nur bzgl. $\ll$ minimale Paare entsprechender Transitionsschritte (π, τ) zu verwenden. Dieses allgemeine Kriterium wird aber erst bei den Optimierungsvorschlägen am Ende des Unterabschnittes aufgegriffen.
Entsprechend diesen Überlegungen definieren wir im Folgenden falsche BPO-Fortsetzungen. Hierbei gehen wir von der Menge der richtigen Fortsetzungen von $\mathcal{L}$ aus und entwickeln eine Repräsentation der Paare von Transitionsschritten im Komplement dieser Menge. Wir betrachten dazu jeden Präfixschritt π zusammen mit allen minimalen Schritten, welche keinen auf π folgenden Schritt darstellen. Für eine endliche partielle Sprache ist diese Repräsentation wiederum endlich.

Definition 4.4.5 (Falsche BPO-Fortsetzung)

DEFINITION 4.4.5 (FALSCHE BPO-FORTSETZUNG)
Sei $\mathcal{L}$ eine partielle Sprache mit der Beschriftungsmenge T. Die Menge der falschen BPO-Fortsetzungen von $\mathcal{L}$ ist definiert als $\mathcal{L}_{wb} = \{(\pi,t) \notin \mathcal{L}^{\Pi}_{co} \mid \pi \in \Pi_{\mathcal{L}}, t \in \mathbb{N}^T, \forall t' < t : (\pi,t') \in \mathcal{L}^{\Pi}_{co}\}$.

Beispiel

BEISPIEL: *Die falschen BPO-Fortsetzungen der partiellen Sprache aus Abbildung 4 sind* $(0,2a)$, $(0,2b)$, $(0,a+b)$, $(0,c)$, $(a,2a)$, $(a,2b)$, (a,c), (b,a), (b,b), $(b,2c)$, $(2a,a)$, $(2a,2b)$, $(2a,c)$, $(a+b,2a)$, $(a+b,b)$, $(a+b,2c)$, $(b+c,a)$, $(b+c,b)$, $(b+c,c)$, $(2a+b,a)$, $(2a+b,b)$, $(2a+b,2c)$, $(a+b+c,2a)$, $(a+b+c,b)$, $(a+b+c,c)$, $(2a+b+c,a)$, $(2a+b+c,b)$, $(2a+b+c,c)$. *Diese 28 Paare umfassen jeden Präfixschritt der partiellen Sprache zusammen mit allen minimalen Schritten, so dass sich keine richtige Fortsetzung ergibt.*

Die weitere Entwicklung eines auf dieser Definition basierenden Berechnungsverfahrens ist nun sehr ähnlich zu den Ausführungen des letzten Unterabschnittes. Auf umfassende Erläuterungen der Konzepte wird daher in diesem Unterabschnitt verzichtet. Wir führen entsprechend zum letzten Unterabschnitt die Begriffe des BPO-separierenden zulässigen Stellentripels und der BPO-Separations-Repräsentation ein. Anschließend beweisen wir, dass das Syntheseproblem genau dann positive lösbar ist, wenn es von einer BPO-Separations-Repräsentation positiv gelöst wird. Als Ergänzung geben wir dann wieder eine genauere auf dem Begriff der BPO-Separations-Repräsentation basierende Charakterisierung, in welchen Fällen das Syntheseproblem eine negative Antwort hat, an. Hierfür benötigen wir einen strengeren Abgeschlossenheitsbegriff als die übliche im letzten Unterabschnitt verwendete Schrittabgeschlossenheit. Die Beweise der Resultate sind zwar ähnlich zu den entsprechenden Überlegungen des letzten Unterabschnittes, erfordern aber eine andere Argumentationsweise.

Definition 4.4.6 (BPO-separierendes zulässiges Stellentripel)

DEFINITION 4.4.6 (BPO-SEPARIERENDES ZULÄSSIGES STELLENTRIPEL)
Sei $(\pi,t) \in \mathcal{L}_{wb}$ eine falsche BPO-Fortsetzung von $\mathcal{L}$. Ein bzgl. (π,t) BPO-separierendes zulässiges Stellentripel ist ein zulässiges Stellentripel st von $\mathcal{L}$ mit der Eigenschaft, dass (π,t) in (N_{st}, m_{st}) nicht aktiviert ist.

Definition 4.4.7 (BPO-Separations-Repräsentation)

DEFINITION 4.4.7 (BPO-SEPARATIONS-REPRÄSENTATION)
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge. Sei ST eine Menge von zulässigen Stellentripeln von $\mathcal{L}$ mit folgender Eigenschaft: Für jedes $(\pi,t) \in \mathcal{L}_{wb}$ enthält ST ein bzgl. (π,t) BPO-separierendes zulässiges Stellentripel oder es gibt kein bzgl. (π,t) BPO-separierendes zulässiges Stellentripel. Das markierte S/T-Netz (N, m_0), dessen Menge von atomaren Teilnetzen der Menge aller von Stellentripeln aus ST definierten atomaren Netzen entspricht, ist eine BPO-Separations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$.

Satz 4.4.2

SATZ 4.4.2
Sei $\mathcal{L}$ eine partielle Sprache mit einer endlichen Beschriftungsmenge und sei (N, m_0) eine BPO-Separations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$. Falls $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$, so gilt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N', m'_0)$ für alle markierten S/T-Netze (N', m'_0).

BEWEIS: Angenommen es gilt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ und es gibt ein markiertes S/T-Netz (N', m_0') mit $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$. Aus $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ folgt $PS(\mathcal{L}) \subsetneq \mathfrak{Bpo}(N, m_0)$ und somit $\mathcal{L}_{co}^{\Pi} = PS(\mathcal{L})_{co}^{\Pi} \subseteq \mathfrak{Bpo}(N, m_0)_{co}^{\Pi}$. Wir unterscheiden nun zwei Fälle.
Im ersten Fall gilt $\mathcal{L}_{co}^{\Pi} \subsetneq \mathfrak{Bpo}(N, m_0)_{co}^{\Pi}$, d.h. es gibt $(\pi, \tau) \in \mathfrak{Bpo}(N, m_0)_{co}^{\Pi} \setminus \mathcal{L}_{co}^{\Pi}$. Der Präfixschritt π korrespondiert zu einer BPO $bpo = (V, <, l) \in \mathfrak{Bpo}(N, m_0)$, d.h. $\pi = |V|_l$. Falls $\pi \in \Pi_{\mathcal{L}}$, so setzen wir $(\pi', \tau') := (\pi, \tau)$. Falls $\pi \notin \Pi_{\mathcal{L}}$, so gibt es ein bzgl. der Relation $\lhd$ maximales Präfix $bpo' = (V', <', l')$ von bpo, so dass der korrespondierende Präfixschritt $\pi' = |V'|_{l'}$ die Beziehung $\pi' \in \Pi_{\mathcal{L}}$ erfüllt. Sei τ' ein nicht-leerer auf π' folgender Schritt in bpo (ein solcher existiert, da bpo' ein echtes Präfix von bpo ist). Dann gilt $(\pi', \tau') \in \mathfrak{Bpo}(N, m_0)_{co}^{\Pi}$. Außerdem gilt aufgrund der Maximalitätseigenschaft von bpo', dass $(\pi', \tau') \notin \mathcal{L}_{co}^{\Pi}$ (sonst wäre $\pi' + \tau' \in \Pi_{\mathcal{L}}$). In beiden Fällen gilt also $(\pi', \tau') \in \mathfrak{Bpo}(N, m_0)_{co}^{\Pi} \setminus \mathcal{L}_{co}^{\Pi}$ und $\pi' \in \Pi_{\mathcal{L}}$. Aus $(\pi', \tau') \notin \mathcal{L}_{co}^{\Pi}$ und $\pi' \in \Pi_{\mathcal{L}}$ folgt, dass es ein $(\pi', \tau'') \notin \mathcal{L}_{co}^{\Pi}$ mit $\tau'' \leqslant \tau'$ und $(\pi', \tau'') \in \mathcal{L}_{wb}$ gibt. Da $(\pi', \tau') \in \mathfrak{Bpo}(N, m_0)_{co}^{\Pi}$ und $\tau'' \leqslant \tau'$, gilt $(\pi', \tau'') \in \mathfrak{Bpo}(N, m_0)_{co}^{\Pi}$. Entsprechend der Definition der BPO-Separations-Repräsentation hat (N, m_0) ein atomares Teilnetz, welches einem bzgl. $(\pi', \tau'') \in \mathcal{L}_{wb}$ BPO-separierenden zulässigen Stellentripel entspricht, oder es gibt kein bzgl. $(\pi', \tau'') \in \mathcal{L}_{wb}$ BPO-separierendes zulässiges Stellentripel. Da $(\pi', \tau'') \in \mathfrak{Bpo}(N, m_0)_{co}^{\Pi}$, ist der erste Fall nicht möglich. Aus $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$ folgt nach Lemma 4.3.1, dass alle atomaren Teilnetze von (N', m_0') zulässigen Stellentripeln entsprechen. Da es kein bzgl. $(\pi', \tau'') \in \mathcal{L}_{wb}$ BPO-separierendes zulässiges Stellentripel gibt, folgt, dass (π', τ'') in (N', m_0') aktiviert ist. Da $\pi' \in \Pi_{\mathcal{L}}$ und $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$, lässt sich weiter folgern, dass $(\pi', \tau'') \in \mathfrak{Bpo}(N', m_0')_{co}^{\Pi}$. Da $(\pi', \tau'') \notin \mathcal{L}_{co}^{\Pi}$, folgt $PS(\mathcal{L})_{co}^{\Pi} = \mathcal{L}_{co}^{\Pi} \neq \mathfrak{Bpo}(N', m_0')_{co}^{\Pi}$. Es folgt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N', m_0')$, ein Widerspruch.
Im zweiten Fall gilt $\mathcal{L}_{co}^{\Pi} = \mathfrak{Bpo}(N, m_0)_{co}^{\Pi}$. Damit folgt $\mathfrak{Bpo}(N, m_0)_{co}^{\Pi} = \mathcal{L}_{co}^{\Pi} = PS(\mathcal{L})_{co}^{\Pi} = \mathfrak{Bpo}(N', m_0')_{co}^{\Pi}$. Aus $PS(\mathcal{L}) \subsetneq \mathfrak{Bpo}(N, m_0)$ folgt, dass ein $bpo \in \mathfrak{Bpo}(N, m_0) \setminus PS(\mathcal{L})$ existiert. Es gilt offensichtlich $\{bpo\}_{co}^{\Pi} \subseteq \mathfrak{Bpo}(N, m_0)_{co}^{\Pi} = \mathfrak{Bpo}(N', m_0')_{co}^{\Pi}$. Dies bedeutet, dass jede richtige Fortsetzung $(\pi, \tau) \in \{bpo\}_{co}^{\Pi}$ von bpo in (N', m_0') aktiviert ist. Damit lässt sich folgern, dass $bpo \in \mathfrak{Bpo}(N', m_0')$. Zusammen mit $bpo \notin PS(\mathcal{L})$ ergibt auch dieser Fall einen Widerspruch zu $PS(\mathcal{L}) = \mathfrak{Bpo}(N', m_0')$. □

Für das folgende Resultat definieren wir den Fortsetzungsabschluss $Fort(\mathcal{L}) = \{bpo \mid bpo \text{ ist BPO mit } \{bpo\}_{co}^{\Pi} \subseteq \mathcal{L}_{co}^{\Pi}\}$ einer partiellen Sprache $\mathcal{L}$ als die Menge aller BPOs, deren erzeugte richtige Fortsetzungen auch richtige Fortsetzungen von $\mathcal{L}$ sind. Eine partielle Sprache $\mathcal{L}$ heißt fortsetzungsabgeschlossen, wenn $Fort(\mathcal{L}) = \mathcal{L}$. Es lässt sich leicht prüfen, dass jede fortsetzungsabgeschlossene partielle Sprache schrittabgeschlossen sowie präfix- und sequentialisierungsabgeschlossen ist. Selbst für eine präfix- und sequentialisierungsabgeschlossene partielle Sprache gilt aber im Allgemeinen nicht, dass Schrittabgeschlossenheit auch Fortsetzungsabgeschlossenheit impliziert.

LEMMA 4.4.5 *Lemma 4.4.5*
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge und sei (N, m_0) eine BPO-Separations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$. Es gilt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ genau dann, wenn es eine falsche BPO-Fortsetzung gibt, bzgl. der kein BPO-separierendes zulässiges Stellentripel existiert, oder wenn $PS(\mathcal{L})$ nicht fortsetzungsabgeschlossen ist.

BEWEIS: Die „genau dann"-Aussage lässt sich wie im Beweis des letzten Satzes nachweisen. Für $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ lassen sich also dieselben zwei Fälle unterscheiden. Im ersten Fall folgt wie im letzten Beweis, dass es eine falsche BPO-Fortsetzung gibt, bzgl. der kein BPO-separierendes zulässiges Stellentripel existiert. Im zweiten Fall folgt auch wie im letzten Beweis, dass es ein bpo $\notin PS(\mathcal{L})$ gibt mit $\{bpo\}^{\Pi}_{co} \subseteq PS(\mathcal{L})^{\Pi}_{co}$. Somit ist $PS(\mathcal{L})$ nicht fortsetzungsabgeschlossen.
Für die „wenn"-Aussage betrachten wir zuerst den Fall, dass es bzgl. einer falschen BPO-Fortsetzung (π, τ) kein BPO-separierendes zulässiges Stellentripel gibt. Da $\pi \in \Pi_{\mathcal{L}}$, gibt es eine BPO bpo $= (V, <, l) \in PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$ mit $|V|_l = \pi$. Sei $\sigma = \tau_1 \dots \tau_n \in \{\sigma(bpo') \mid bpo' \in Slin(bpo)\}$, dann folgt, dass $\sigma' = \tau_1 \dots \tau_n \tau$ in (N, m_0) aktiviert ist und somit für die BPO bpo$'$ mit $\sigma(bpo') = \sigma'$ die Beziehung $bpo' \in \mathfrak{Bpo}(N, m_0)$ gilt. Nach Definition einer falschen BPO-Fortsetzung folgt außerdem bpo$' \notin PS(\mathcal{L})$ und somit $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$. Der zweite Fall ist, dass $PS(\mathcal{L})$ nicht fortsetzungsabgeschlossen ist, d.h. es gibt eine bpo $\notin PS(\mathcal{L})$ mit $\{bpo\}^{\Pi}_{co} \subseteq PS(\mathcal{L})^{\Pi}_{co} \subseteq \mathfrak{Bpo}(N, m_0)^{\Pi}_{co}$. Dies bedeutet, dass jede richtige Fortsetzung $(\pi, \tau) \in \{bpo\}^{\Pi}_{co}$ von bpo in (N, m_0) aktiviert ist. Damit lässt sich folgern, dass bpo $\in \mathfrak{Bpo}(N, m_0)$. Somit folgt wieder $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$. □

Zu Synthesezwecken ist insbesondere wieder der Fall speziell zu beachten, in dem es für jede falsche BPO-Fortsetzung ein BPO-separierendes zulässiges Stellentripel gibt. Dann gilt ähnlich wie in Lemma 4.4.2 der folgende Zusammenhang.

Lemma 4.4.6

LEMMA 4.4.6
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge und sei (N, m_0) eine BPO-Separations-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$ mit der Eigenschaft, dass (N, m_0) für jede falsche BPO-Fortsetzung ein BPO-separierendes zulässiges Stellentripel enthält. Dann gilt $\mathfrak{Bpo}(N, m_0)^{\Pi}_{co} = \mathcal{L}^{\Pi}_{co}$. Für das Ablaufverhalten lässt sich hiermit die Minimalitätseigenschaft $\mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m'_0)$ für alle markierten S/T-Netze (N', m'_0) mit $\mathcal{L} \subseteq \mathfrak{Bpo}(N', m'_0)$ folgern.

BEWEIS: Die erste Aussage wird durch den ersten Fall im Beweis zu Satz 4.4.2 gezeigt. Aus dieser Aussage folgt $\mathfrak{Bpo}(N, m_0)^{\Pi}_{co} \subseteq \mathfrak{Bpo}(N', m'_0)^{\Pi}_{co}$. Mit Lemma 3.3.1 ergibt sich somit die zweite Aussage. □

Trotzdem stellt eine BPO-Separations-Repräsentation in diesem Fall nicht notwendigerweise eine positive Lösung für das Syntheseproblem dar. Wie in Lemma 4.4.5 gezeigt, trifft dies dann zu, wenn $PS(\mathcal{L})$ nicht fortsetzungsabgeschlossen ist.
Das Syntheseproblem lässt sich also nicht unmittelbar durch die Betrachtung aller falscher BPO-Fortsetzungen lösen. Aber wie in Satz 4.4.2 gezeigt, lässt es sich analog wie im Falle einer Schrittseparations-Repräsentation dadurch lösen, dass eine BPO-Separations-Repräsentation berechnet wird und dann geprüft wird, ob die BPO-Separations-Repräsentation das spezifizierte Verhalten aufweist oder nicht. Letzterer Punkt wird im nächsten Abschnitt behandelt.

Beispiel

BEISPIEL: *Das Netz aus Abbildung 3 ist auch eine BPO-Separations-Repräsentation der partiellen Sprache aus Abbildung 4. Beispielsweise ist die Stelle p_2 BPO-separierend bzgl. der 9 falschen BPO-Fortsetzungen $(0, 2a)$, $(0, 2b)$, $(0, a + b)$, (b, a), (b, b), $(a + b, 2a)$, $(b + c, a)$, $(b + c, b)$ und $(a + b + c, 2a)$. Die übrigen 19 falschen BPO-Fortsetzungen werden durch die*

anderen drei Stellen verhindert. Das Netz löst, wie schon besprochen, das Syntheseproblem.
Wir betrachten nun die partielle Sprache $\mathcal{L}'$ aus Abbildung 54. Das Netz aus Abbildung 57 ist eine BPO-Separations-Repräsentation, welche alle falschen BPO-Fortsetzungen verhindert (es genügt also Lemma 4.4.6). Hierbei ist zu beachten, dass sich die falsche Schrittfortsetzung $bcaa$ *nicht separieren lässt, es aber keine entsprechende falsche BPO-Fortsetzung gibt, da* $(a+b+c, a)$ *eine richtige Fortsetzung ist. Obwohl alle falschen BPO-Fortsetzungen separiert sind, hat das Netz dennoch nicht das Ablaufverhalten* $PS(\mathcal{L}')$*. Dies liegt daran, dass* $PS(\mathcal{L}')$ *zwar schrittabgeschlossen, jedoch nicht fortsetzungsabgeschlossen ist. Hierfür ist nun gerade die zu* $bcaa$ *korrespondierende totale BPO verantwortlich. Das Syntheseproblem hat also eine negative Antwort.*
Ähnliches ergibt sich für die partielle Sprache $\mathcal{L}''$ aus Abbildung 71. Das Netz aus Abbildung 3 ist hierzu eine BPO-Separations-Repräsentation, welche alle falschen BPO-Fortsetzungen verhindert. In diesem Fall ist $PS(\mathcal{L}'')$ *schon nicht schrittabgeschlossen und damit natürlich auch nicht fortsetzungsabgeschlossen.*

Im weiteren Verlauf dieses Unterabschnittes soll der Begriff des BPO-separierenden zulässigen Stellentripels mit Hilfe von Regionen charakterisiert werden. Auf diese Weise lässt sich dann eine für Berechnungszwecke verwendbare linear algebraische Repräsentation gewinnen. Die entsprechenden Überlegungen sind analog zum letzten Unterabschnitt. Der einzige Unterschied der nächsten Definition und der zwei darauf folgenden Lemmata zu den entsprechenden Ausführungen des letzten Unterabschnittes ist, dass bei der Betrachtung einer falschen BPO-Fortsetzung (π, τ) anstelle einer falschen Schrittfortsetzung $\sigma = \tau_1 \dots \tau_j$ der Präfixschritt π die Rolle von $\tau_1 + \ldots + \tau_{j-1}$ einnimmt und der darauf folgende Transitionsschritt τ dem letzten Schritt τ_j der falschen Schrittfortsetzung entspricht.
Für die Definition einer BPO-separierenden Markenfluss-Region fixieren wir wieder für jede Transition $t \in T$ ein Beispielereignis ν_t einer BPO aus $\mathcal{L}$, welches mit t beschriftet ist, und das Quellenereignis ν_0 einer beliebigen BPO aus $\mathcal{L}$. Wir bezeichnen die Menge der in ν_t eingehenden Kanten mit E_t^{zu}, die Menge der aus ν_t ausgehenden Kanten mit E_t^{ab} und die Menge der aus ν_0 ausgehenden Kanten mit E_0

Definition 4.4.8 (BPO-separierende Region)

Definition 4.4.8 (BPO-separierende Region)
Sei $(\pi, \tau) \in \mathcal{L}_{wb}$ eine falsche BPO-Fortsetzung von $\mathcal{L}$.

- *Eine bzgl. (π, τ) BPO-separierende BPO-Transitions-Region ist eine BPO-Transitions-Region $\mathbf{r} = (r_0, \dots, r_{2m}) \in \mathbb{N}^{2m+1}$ von $\mathcal{L}$ bzgl. $T = \{t_1, \dots, t_m\}$ mit der Eigenschaft*

$$\text{(TBS)} \qquad r_0 + \sum_{i=1}^{m} (\pi(t_i) \cdot r_i - (\pi + \tau)(t_i) \cdot r_{m+i}) < 0.$$

- *Eine bzgl. (π, τ) BPO-separierende Markenfluss-Region ist eine Markenfluss-Region r von $\mathcal{L}$ mit der Eigenschaft*

$$\text{(MBS)} \qquad \sum_{e \in E_0} r(e) + \sum_{t \in T} \Big(\sum_{e \in E_t^{ab}} \pi(t) \cdot r(e) - \sum_{e \in E_t^{zu}} (\pi + \tau)(t) \cdot r(e) \Big) < 0.$$

Lemma 4.4.7

LEMMA 4.4.7
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge. Für ein Stellentripel st *und eine falsche BPO-Fortsetzung (π,τ) von $\mathcal{L}$ sind folgende drei Aussagen äquivalent:*

- *Das Stellentripel* st *ist ein bzgl. (π,τ) BPO-separierendes zulässiges Stellentripel.*
- *Es gibt eine bzgl. (π,τ) BPO-separierende BPO-Transitions-Region $\mathbf{r}$ von $\mathcal{L}$ bzgl. $T=\{t_1,\ldots,t_m\}$, so dass* $st=\overrightarrow{\mathbf{r}}$.
- *Es gibt eine bzgl. (π,τ) BPO-separierende Markenfluss-Region r von $\mathcal{L}$, so dass* $st=\overrightarrow{r}$.

BEWEIS: Wird im Gegensatz zum Beweis von Lemma 4.4.3 (π,τ) anstatt $\sigma=\tau_1\ldots\tau_j$, π anstatt $\tau_1+\ldots+\tau_{j-1}$ und τ anstatt τ_j betrachtet, so lässt sich hier ein analoger Beweis führen (nur die Argumentation im Beweis zu Lemma 4.4.3 über die Aktiviertheit von $\tau_1\ldots\tau_{j-1}$ in (N_{st},m_{st}) ist hier unnötig). □

Lemma 4.4.8

LEMMA 4.4.8
Sei $\mathcal{L}$ eine endliche partielle Sprache und $(\pi,\tau)\in\mathcal{L}_{wb}$ eine falsche BPO-Fortsetzung von $\mathcal{L}$.

- *Sei die Beschriftungsmenge $T=\{t_1,\ldots,t_m\}$ geordnet. Die Menge der bzgl. (π,τ) BPO-separierenden BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T=\{t_1,\ldots,t_m\}$ entspricht der Menge der ganzzahligen Lösungen des homogenen linearen Ungleichungssystems*

$$\mathbf{A}_{\mathcal{L}}^{BPOT}\cdot\mathbf{x}\geqslant\mathbf{0},\mathbf{x}\geqslant\mathbf{0},$$

$$\mathbf{b}_{(\pi,\tau)}^{TBS}\cdot\mathbf{x}<0.$$

Der Vektor $\mathbf{b}_{(\pi,\tau)}^{TBS}=(b_{(\pi,\tau),0},\ldots,b_{(\pi,\tau),2m})$ ist gegeben durch

$$b_{(\pi,\tau),i}=\begin{cases}1 & \text{für } i=0,\\ \pi(t_i) & \text{für } i=1,\ldots,m\\ -(\pi+\tau)(t_{i-m}) & \text{für } i=m+1,\ldots,2m.\end{cases}$$

- *Sei eine Nummerierung der Menge $\bigcup_{(V,<,l)\in\mathcal{L}}<^*=\{e_1,\ldots,e_n\}$ der Kanten aller $*$-Erweiterungen der BPOs aus $\mathcal{L}$ gegeben. Die Menge der bzgl. (π,τ) BPO-separierenden Markenfluss-Regionen von $\mathcal{L}$ entspricht der Menge der zu ganzzahligen Lösungen des homogenen linearen Ungleichungssystems*

$$\mathbf{A}_{\mathcal{L}}^{MAR}\cdot\mathbf{x}=\mathbf{0},\mathbf{x}\geqslant\mathbf{0},$$

$$\mathbf{b}_{(\pi,\tau)}^{MBS}\cdot\mathbf{x}<0$$

bzgl. $\{e_1, \ldots, e_n\}$ *korrespondierenden globalen Markenfluss-Funktionen. Der Vektor* $\mathbf{b}^{MBS}_{(\pi,\tau)} = (b_{(\pi,\tau),0}, \ldots, b_{(\pi,\tau),n})$ *ist gegeben durch*

$$b_{(\pi,\tau),i} = \begin{cases} 1 & \text{für } e_i \in E_0 \\ & \text{und } e_i \notin \bigcup_{t' \in T} E^{zu}_{t'}, \\ 1-(\pi+\tau)(t') & \text{für } e_i \in E_0 \\ & \text{und } e_i \in E^{zu}_{t'}, \\ \pi(t) & \text{für } e_i \in E^{ab}_t \\ & \text{und } e_i \notin \bigcup_{t' \in T} E^{zu}_{t'}, \\ -(\pi+\tau)(t') & \text{für } e_i \notin (\bigcup_{t \in T} E^{ab}_t \cup E_0) \\ & \text{und } e_i \in E^{zu}_{t'}, \\ \pi(t)-(\pi+\tau)(t') & \text{für } e_i \in E^{ab}_t \\ & \text{und } e_i \in E^{zu}_{t'}, \\ 0 & \text{für } e_i \notin (\bigcup_{t \in T} E^{ab}_t \cup E_0) \\ & \text{und } e_i \notin \bigcup_{t' \in T} E^{zu}_{t'}. \end{cases}$$

BEWEIS: Hier kann analog zum Beweis von Lemma 4.4.4 argumentiert werden. □

BEISPIEL: *Wir betrachten die falsche BPO-Fortsetzung* (b, a) *der partiellen Sprache aus Abbildung 4. Es gilt* $\mathbf{b}^{TBS}_{(b,a)} = (1,0,1,0,-1,-1,0)$ *bzgl. der Transitionsreihenfolge* a, b, c. *Die BPO-Transitions-Region* $(1,2,0,0,1,1,0)$ *ist eine Lösung von* $\mathbf{A}^{BPOT}_{\mathcal{L}} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, \mathbf{b}^{TBS}_{(b,a)} \cdot \mathbf{x} < 0$. *Die korrespondierende zulässige Stelle verhindert dementsprechend die Aktiviertheit von* (b, a).
Für die Kanten-Nummerierung aus Abbildung 68 sowie der Festlegung, dass v_0 *der Quellenknoten von* bpo_1, v_a *das erste* a*-Ereignis in* bpo_2, v_b *das* b*-Ereignis aus* bpo_1 *und* v_c *das* c*-Ereignis aus* bpo_1 *ist, ergibt sich* $\mathbf{b}^{MBS}_{(b,a)} = (1,1,0,1,1,0,0,0,0,-1,0,0,0,0,0,0,0,0,0)$. *Der zur Markenfluss-Region aus Abbildung 67 korrespondierende Vektor* $(0,0,1,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,2)$ *ist eine Lösung von* $\mathbf{A}^{MAR}_{\mathcal{L}} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, \mathbf{b}^{MBS}_{(b,a)} \cdot \mathbf{x} < 0$. *Die korrespondierende zulässige Stelle verhindert daher die Aktiviertheit von* (b, a).

Beispiel

Nun ergibt sich ebenfalls analog zum letzten Unterabschnitt in Pseudo-Code der folgende Algorithmus zur Konstruktion einer BPO-Separations-Repräsentation.

ALGORITHMUS 4.4.2 (BERECHNUNG BPO-SEPARATIONS-REPRÄSENTATION)

Algorithmus 4.4.2 (Berechnung BPO-Separations-Repräsentation)

```
1   Eingabe: Partielle Sprache L
2   Ausgabe: BPO–Separations–Repräsentation von L
3   BEGIN
4       A^BPOT_L bzw. A^MAR_L :=
          L.regionenMatrix(BPOT bzw. MAR);
5       L_wb := L.falscheBPOFortsetzungen();
6       (N, m_0) := (∅, L.beschriftungsMenge(), ∅, ∅);
7       FORALL (π, τ) ∈ L_wb DO
8           IF (π, τ).istAktiviertIn(N, m_0) THEN
9               r := ganzzahligeLoesung
                   (A^BPOT_L · x ⩾ 0, x ⩾ 0, b^TBS_(π,τ) · x < 0 bzw.
                   A^MAR_L · x = 0, x ⩾ 0, b^MBS_(π,τ) · x < 0);
10              IF r ≠ null THEN
11                  (N, m_0).fuegeHinzu(r⃗);
```

```
12                ENDIF
13            ENDIF
14        ENDFORALL
15        RETURN (N, m_0);
16    END
```

Wiederum können die einzelnen Zulässigkeitsprobleme des Algorithmus in polynomieller Zeit in Abhängigkeit von der Größe des jeweiligen Ungleichungssystems gelöst werden. Problematisch ist auch an diesem Algorithmus, dass die Anzahl der falschen BPO-Fortsetzungen exponentiell in der Größe der gegebenen partiellen Sprache $\mathcal{L}$ sein kann. Wird $|T|$ als konstant angesehen, so ist die Anzahl der falschen BPO-Fortsetzungen im schlechtesten Fall linear in $|\mathrm{Pref}(\mathcal{L})|$, wobei zu beachten ist, dass die Anzahl der Präfixe einer BPO exponentiell in der Größe der BPO sein kann. Somit kann es auch hier vorkommen, dass sowohl die Anzahl der Zulässigkeitsprobleme, welche gelöst werden müssen, als dementsprechend auch die Anzahl der Stellentripel, welche zu dem konstruierten Netz hinzugefügt werden, exponentiell in der Größe der Eingabe $\mathcal{L}$ sind. Wie schon im letzten Unterabschnitt sind diese Anzahlen aber meist wesentlich kleiner, da der Test, ob eine falsche BPO-Fortsetzung in dem bisher konstruierten Netz aktiviert ist, häufig negativ ausfällt.

Beispiel

BEISPIEL: *Ähnlich wie im Falle von Algorithmus 4.4.1 ist es möglich, dass Algorithmus 4.4.2 ausgehend von der partiellen Sprache aus Abbildung 4 die BPO-Separations-Repräsentation mit vier Stellen aus Abbildung 3 berechnet. In dieser Situation werden nur 4 Zulässigkeitsprobleme gelöst, obwohl es 28 falsche BPO-Fortsetzungen gibt.*

In folgender Bemerkung diskutieren wir Möglichkeiten, den betrachteten Algorithmus im Hinblick auf die Laufzeit und auf die Größe der konstruierten BPO-Separations-Repräsentation zu optimieren.

Bemerkung 4.4.3 (Optimierungsvorschläge)

BEMERKUNG 4.4.3 (OPTIMIERUNGSVORSCHLÄGE)
Die folgenden Optimierungen von Algorithmus 4.4.2 sind möglich.

- *Im Wesentlichen sind dieselben Optimierungen wie in Bemerkung 4.4.1 möglich, d.h. Abbruch des Algorithmus, falls ein Zulässigkeitsproblem eine negative Antwort hat, Reduktion der Menge der falschen BPO-Fortsetzungen sowohl direkt auf der Ebene der falschen Schrittfortsetzungen als auch durch die Suche nach entsprechenden redundanten Ungleichungen, Wahl einer möglichst kleinen* $\mathbf{A}$*-Matrix, Wahl eines geeigneten Verfahrens zur Lösung der Zulässigkeitsprobleme einschließlich der Möglichkeit der Betrachtung von Zielfunktionen, Wahl einer geeigneten Reihenfolge der falschen BPO-Fortsetzungen und das Entfernen von Stellen des konstruierten Netzes insbesondere durch das Suchen nach impliziten Stellen.*

- *Der einzige konzeptuell entscheidende Unterschied ergibt sich bei der Suche nach unnötigen falschen BPO-Fortsetzungen direkt auf der Ebene der falschen BPO-Fortsetzungen. Eine sehr interessante Möglichkeit ist hier, nur die bzgl.* $\ll$ *minimalen falschen BPO-Fortsetzungen zu betrachten. Gilt für zwei falsche BPO-Fortsetzungen* (π, t) *und* (π', t')*, dass* $(\pi, t) \ll (\pi', t')$*, so lässt sich mit analogen Überlegungen wie in Lemma 4.3.9 folgern, dass jedes bzgl.* (π, t) *BPO-separierende zulässige Stellentripel auch bzgl.* (π', t') *BPO-separierend ist. Daher muss*

(π', τ'), wie im zweiten Punkt von Bemerkung 4.4.1 erklärt, zu Synthesezwecken nicht als falsche BPO-Fortsetzung betrachtet werden. Allerdings ist es wiederum fraglich, ob sich der zusätzliche Berechnungsaufwand zur Reduktion der Menge der falschen BPO-Fortsetzungen lohnt.

BEISPIEL: *Für die falschen BPO-Fortsetzungen der partiellen Sprache aus Abbildung 4 gilt* $(b, b) \ll (0, 2b)$, $(b, a) \ll (0, a + b)$, $(2a, a) \ll (a, 2a)$, $(a + b, b) \ll (a, 2b)$, $(b + c, c) \ll (b, 2c)$, $(2a + b, b) \ll (2a, 2b)$, $(2a + b, a) \ll (a + b, 2a)$, $(a + b + c, c) \ll (a + b, 2c)$, $(2a + b + c, c) \ll (2a + b, 2c)$, $(2a + b + c, a) \ll (a + b + c, 2a)$. *Mit dem zweiten Optimierungsvorschlag kann somit die jeweils $\ll$-größere falsche BPO-Fortsetzung weggelassen werden, wodurch sich die Anzahl der zu betrachtenden falschen BPO-Fortsetzungen von 28 auf 18 reduziert.* *Beispiel*

Insgesamt stellt die Verwendung von falschen BPO-Fortsetzungen eine grundlegende Weiterentwicklung gegenüber der Verwendung falscher Schrittfortsetzungen dar. Zum einen ist die zur Berechnung von falschen BPO-Fortsetzungen nötige Betrachtung aller Schnitte der BPOs einer partiellen Sprache weniger aufwändig als die im Falle von falschen Schrittfortsetzungen notwendige Betrachtung aller Präfixe von Schrittlinearisierungen der BPOs. Zum anderen haben die Überlegungen zu Beginn des Unterabschnittes gezeigt, dass im Falle von falschen BPO-Fortsetzungen weniger falsche Fortsetzungen betrachtet werden als im Falle von falschen Schrittfortsetzungen. Ähnlich wie in Lemma 4.3.10 lässt sich für alle Regionendefinitionen auch formal nachweisen, dass einerseits zwei verschiedene falsche BPO-Fortsetzungen keine identische **b**-Ungleichung erzeugen und andererseits für jede von einer falschen BPO-Fortsetzung erzeugten **b**-Ungleichungen eine falsche Schrittfortsetzung existiert, welche dieselbe **b**-Ungleichung erzeugt.
Im schlechtesten Fall gilt aber auch für falsche BPO-Fortsetzungen, dass selbst bei der Verwendung aller vorgestellten Optimierungen die exponentielle Abhängigkeit der Anzahl der Zulässigkeitsprobleme, welche gelöst werden müssen, als dementsprechend auch der Anzahl der Stellentripel, welche zu dem konstruierten Netz hinzugefügt werden, von der Größe der Eingabe $\mathcal{L}$ bestehen bleibt. Eine polynomielle Abhängigkeit scheint hier im Rahmen der Idee der Betrachtung falscher Fortsetzungen nicht realisierbar. Im nächsten Unterabschnitt werden wir eine zu den bisherigen auf falschen Fortsetzungen basierenden Ausführungen konzeptuell andere Herangehensweise zur Berechnung einer geeigneten endlichen Menge von zulässigen Stellentripeln vorstellen.

4.4.3 *Erzeugendensystem*

Die Idee in diesem Unterabschnitt ist, eine endliche Menge an zulässigen Stellentripeln von $\mathcal{L}$ zu finden, welche dasselbe Verhalten erzeugt wie die unendliche Menge aller zulässiger Stellentripel. Wir sind also an einem endlichen Erzeugendensystem der Menge aller zulässiger Stellentripel interessiert, welches eine ebenso starke Verhaltenseinschränkung darstellt wie die gesamte Menge der zulässigen Stellentripel. Die zentrale Eigenschaft eines solchen Erzeugendensystems ist also, dass jede BPO, welche bzgl. der Menge aller zulässiger Stellentripel nicht aktiviert ist, auch bzgl. des Erzeugendensystems nicht aktiviert ist. Die umgekehrte Eigenschaft, dass eine BPO, welche bzgl. der Menge aller zulässiger Stellentripel aktiviert ist, auch bzgl. dem Erzeugenden-

system, also einer Teilmenge der Menge aller zulässiger Stellentripel, aktiviert ist, ist ohnehin klar. Ein Netz, welches aus den Stellentripeln eines solchen Erzeugendensystems besteht, hat also exakt das gleiche Ablaufverhalten wie das gesättigt zulässige Netz. Es stellt somit eine geeignete endliche Repräsentation des gesättigt zulässigen Netzes dar. Insbesondere erfüllt es auch die Eigenschaft, dass das Ablaufverhalten des Netzes entweder mit $PS(\mathcal{L})$ übereinstimmt oder es kein Netz mit dem Ablaufverhalten $PS(\mathcal{L})$ gibt.
Ein entsprechendes Erzeugendensystem wäre also für die Lösung des Syntheseproblems geeignet, bisher haben wir aber die Frage, ob es im Falle einer endlichen Sprache $\mathcal{L}$ solch ein endliches Erzeugendensystem im Allgemeinen überhaupt gibt, noch offen gelassen. Hierzu werden wir nun das wichtige Resultat herleiten, dass, falls ein Stellentripel eine nicht-negative Linearkombination einer Menge von Stellentripeln ist, dann jede BPO, welche bzgl. des Stellentripels nicht aktiviert ist, auch bzgl. der Menge an Stellentripeln nicht aktiviert ist. Somit stellt eine Menge von zulässigen Stellentripeln mit der Eigenschaft, dass sich jedes zulässige Stellentripel als entsprechende Linearkombination dieser Menge darstellen lässt, ein geeignetes Erzeugendensystem dar. Für eine endliche Sprache $\mathcal{L}$ lässt sich die Menge aller zulässiger Stellentripel nun entsprechend der verschiedenen Regionenkonzepte durch die Lösungsmenge eines linearen Ungleichungssystems charakterisieren. Dieses Ungleichungssystem definiert jeweils einen polyedrischen Kegel. Entsprechend den Ausführungen in Abschnitt 3.4 gibt es für einen solchen Kegel ein endliches Erzeugendensystem, so dass sich jeder Lösungsvektor des Ungleichungssystems als nicht-negative Linearkombination von Vektoren des Erzeugendensystems darstellen lässt. Diese Eigenschaft überträgt sich auch auf die zu Lösungsvektoren korrespondierenden zulässigen Stellentripel. Daher stellt die endliche Menge der zu einem Erzeugendensystem des Kegels korrespondierenden zulässigen Stellentripel ein geeignetes endliches Erzeugendensystem der Menge aller zulässiger Stellentripel dar.
Das Vorgehen der Betrachtung eines Erzeugendensystems wurde schon in [153] für den Spezialfall der dort verwendeten Definition von Markenfluss-Regionen vorgeschlagen (auch in [18, 65] wird ein solcher Ansatz zumindest implizit schon für Regionen von sequentiellen Sprachen verwendet). Hier wiederholen wir diesen Ansatz in dem betrachteten allgemeineren Kontext. Allerdings sind die Ausführungen dieses Unterabschnittes klarer strukturiert und in einigen Aspekten detaillierter. Insbesondere die Optimierungsvorschläge für den resultierenden Synthesealgorithmus am Ende des Unterabschnittes sind neu. Außerdem stellen wir im Gegensatz zu [153] in den folgenden Abschnitten eine Implementierung des Algorithmus vor und diskutieren und testen die Komplexität des Algorithmus.
Entsprechend obigen Überlegungen definieren wir zuerst formal den Begriff einer nicht-negativen Linearkombination von Stellentripeln, um dann zu beweisen, dass das Weglassen eines Stellentripels aus einer Menge von Stellentripeln das Ablaufverhalten eines zugehörigen Netzes nicht verändert, falls das Stellentripel eine Linearkombination der anderen Stellentripel ist. Sei $ST \neq \emptyset$ eine Menge von Stellentripel über T und sei st ein weiteres Stellentripel über T. Falls eine Familie von nicht-negativen reellen Zahlen $(\lambda_{st'})_{st' \in ST}$ existiert, wobei nur endliche viele Zahlen der Familie ungleich Null sind, so dass $st_0 = \sum_{st' \in ST} \lambda_{st'} \cdot st'_0$, $^\circ st(t) = \sum_{st' \in ST} \lambda_{st'} \cdot {}^\circ st'(t)$ für alle $t \in T$

und $st(t)^\circ = \sum_{st' \in ST} \lambda_{st'} \cdot st'(t)^\circ$ für alle $t \in T$, so schreiben wir $st = \sum_{st' \in ST} \lambda_{st'} \cdot st'$ und nennen st eine nicht-negative Linearkombination von ST. Für nicht-negative Linearkombinationen gilt der folgende Zusammenhang. Ein ähnliches Resultat wurde auch in [153] mit der dort verwendeten Notation bewiesen.

LEMMA 4.4.9 *Lemma 4.4.9*

Sei ST eine Menge von Stellentripeln über T und $st = \sum_{st' \in ST} \lambda_{st'} \cdot st' \notin ST$ *eine nicht-negative Linearkombination von ST. Sei weiter* (N, m_0), $N = (P, T, W)$, *das markierte S/T-Netz, dessen Menge von atomaren Teilnetzen der Menge aller von Stellentripeln aus* $ST \cup \{st\}$ *definierten atomaren Netzen entspricht, und* (N', m_0'), $N' = (P', T, W')$, *das markierte S/T-Netz, dessen Menge von atomaren Teilnetzen der Menge aller von Stellentripeln aus ST definierten atomaren Netzen entspricht. Dann gilt* $\mathfrak{Bpo}(N, m_0) = \mathfrak{Bpo}(N', m_0')$.

BEWEIS: $\mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m_0')$ folgt direkt nach Lemma 4.3.1. $\mathfrak{Bpo}(N, m_0) \supseteq \mathfrak{Bpo}(N', m_0')$ ergibt sich folgendermaßen. Sei $bpo = (V, <, l)$ eine bzgl. (N', m_0') aktivierte BPO. Entsprechend Lemma 3.3.1 gilt für einen Schnitt C von bpo und die zu $st' \in ST$ gehörige Stelle $p_{st'} \in P'$, dass $m_0'(p_{st'}) + \sum_{v \in V \wedge v < C} (W'(l(v), p_{st'}) - W'(p_{st'}, l(v))) \geqslant \sum_{v \in C} W'(p_{st'}, l(v))$. Dies bedeutet, dass für einen Schnitt C von bpo und die zu st gehörige Stelle $p_{st} \in P$ der folgende Zusammenhang gilt:

$$
\begin{aligned}
& m_0(p_{st}) + \sum_{v \in V \wedge v < C} (W(l(v), p_{st}) - W(p_{st}, l(v))) \\
= \; & st_0 + \sum_{v \in V \wedge v < C} ({}^\circ st(l(v)) - st^\circ(l(v))) \\
= \; & \sum_{st' \in ST} \lambda_{st'} \cdot (st_0' + \sum_{v \in V \wedge v < C} ({}^\circ st'(l(v)) - st'^\circ(l(v)))) \\
= \; & \sum_{st' \in ST} \lambda_{st'} \cdot (m_0'(p_{st'}) + \sum_{v \in V \wedge v < C} (W'(l(v), p_{st'}) - W'(p_{st'}, l(v)))) \\
\geqslant \; & \sum_{st' \in ST} \lambda_{st'} \cdot (\sum_{v \in C} W'(p_{st'}, l(v))) \\
= \; & \sum_{v \in C} (\sum_{st' \in ST} \lambda_{st'} \cdot st'^\circ(l(v))) \\
= \; & \sum_{v \in C} (st^\circ(l(v))) \\
= \; & \sum_{v \in C} W(p_{st}, l(v)).
\end{aligned}
$$

Für einen Schnitt C von bpo und die zu $st' \in ST$ gehörige Stelle $p_{st'} \in P$ gilt offensichtlich $m_0(p_{st'}) + \sum_{v \in V \wedge v < C} (W(l(v), p_{st'}) - W(p_{st'}, l(v))) \geqslant \sum_{v \in C} W(p_{st'}, l(v))$. Somit ist bpo bzgl. (N, m_0) aktiviert. □

BEISPIEL: *Das Netz in Abbildung 42 zeigt Beispiele für nicht-negative Linearkombinationen von Stellentripeln. Die Stellen* $p_2 + 2p_3$ *und* $2p_4$ *ergeben sich als entsprechende Linearkombinationen der anderen Stellen. Umgekehrt ergibt sich* p_4 *auch als* $0{,}5 \cdot 2p_4$. *Durch das Weglassen beispielsweise der Stelle* $p_2 + 2p_3$ *ändert sich das Ablaufverhalten des Netzes dementsprechend nicht.* *Beispiel*

Stellentripel, welche sich als nicht-negative Linearkombination einer Menge von Stellentripeln ergeben, schränken also das Verhalten eines Netzes gegenüber der ursprünglichen Menge von Stellentripeln

nicht zusätzlich ein. Die zentrale Syntheseidee ist also, ein Erzeugendensystem von zulässigen Stellentripeln zu betrachten, so dass jedes zulässige Stellentripel eine Linearkombination des Erzeugendensystems ist. Werden die Stellentripel eines solchen Erzeugendensystems zur Beschriftungsmenge von $\mathcal{L}$ hinzugefügt, so entsteht eine sog. Erzeugendensystem-Repräsentation des gesättigt zulässigen Netzes. Eine Erzeugendensystem-Repräsentation weist dasselbe Ablaufverhalten wie das gesättigt zulässige Netz auf.

Definition 4.4.9 (Erzeugendensystem-Repräsentation)

Definition 4.4.9 (Erzeugendensystem-Repräsentation)
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge. Sei ST eine Menge von zulässigen Stellentripeln von $\mathcal{L}$, derart, dass jedes zulässige Stellentripel von $\mathcal{L}$ eine nicht-negative Linearkombination von ST ist. Das markiertes S/T-Netz (N, m_0), dessen Menge von atomaren Teilnetzen der Menge aller von Stellentripeln aus ST definierten atomaren Netzen entspricht, ist eine Erzeugendensystem-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$.

In den Ausführungen zu Beginn des Unterabschnittes haben wir dargelegt, dass es für ein endliches $\mathcal{L}$ immer eine endliche Erzeugendensystem-Repräsentation gibt.
Es ist mit Lemma 4.4.9 unmittelbar klar, dass eine Erzeugendensystem-Repräsentation exakt dasselbe Ablaufverhalten wie das gesättigt zulässige Netz aufweist. Somit erfüllt es dieselbe in folgendem Lemma formulierte Minimalitätseigenschaft wie das gesättigt zulässige Netz. Diese Eigenschaft gilt im Allgemeinen für keine der Separations-Repräsentationen (vgl. Lemma 4.4.2 und Lemma 4.4.6).

Lemma 4.4.10

Lemma 4.4.10
Sei (N, m_0) eine Erzeugendensystem-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$. Dann gilt $\mathcal{L} \subseteq \mathfrak{Bpo}(N, m_0)$ und $\mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m_0')$ für alle markierten S/T-Netze (N', m_0') mit $\mathcal{L} \subseteq \mathfrak{Bpo}(N', m_0')$.

Beweis: Dies ergibt sich direkt aus Lemma 4.3.3 und Lemma 4.4.9. □

Insbesondere gilt also wieder die entscheidende Syntheseeigenschaft, dass eine Erzeugendensystem-Repräsentation entweder das Syntheseproblem positiv löst, oder aber es keine positive Lösung gibt.

Satz 4.4.3

Satz 4.4.3
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge und sei (N, m_0) eine Erzeugendensystem-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$. Falls $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$, so gilt $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N', m_0')$ für alle markierten S/T-Netze (N', m_0').

Beweis: Dies ergibt sich direkt aus Korollar 4.3.2 und Lemma 4.4.9. □

Das Syntheseproblem kann also dadurch gelöst werden, dass eine Erzeugendensystem-Repräsentation berechnet wird und dann getestet wird, ob deren Ablaufsemantik mit dem spezifizierten Ablaufverhalten übereinstimmt. Entsprechende Testverfahren werden im nächsten Abschnitt diskutiert.

BEISPIEL: *Abbildung 73 zeigt eine Erzeugendensystem-Repräsentation der partiellen Sprache aus Abbildung 4. Dieses Netz stellt sogar eine Erzeugendensystem-Repräsentation mit einer minimalen Anzahl an Stellen dar. Das Netz enthält insbesondere alle Stellen, welche auch im Netz aus Abbildung 3 vorkommen. Außerdem hat es dasselbe Ablaufverhalten wie das Netz aus Abbildung 3. Damit weist es die von der partiellen Sprache aus Abbildung 4 spezifizierte Ablaufsemantik auf. Es ist also ein Lösungsnetz für das Syntheseproblem.* *Beispiel*

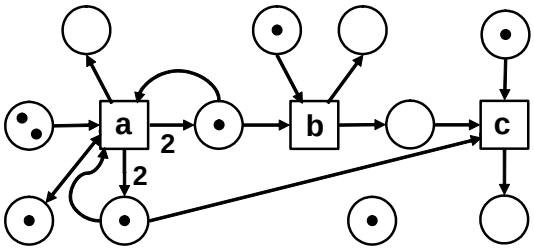

Abbildung 73: Erzeugendensystem-Repräsentation der partiellen Sprache aus Abbildung 4.

Zur effektiven Berechnung einer Erzeugendensystem-Repräsentation verwenden wir wieder die verschiedenen Regionenkonzepte des letzten Abschnittes. Dementsprechend führen wir den Begriff des Erzeugendensystems für die Regionendefinitionen der BPO-Transitions-Region und der Markenfluss-Region ein. Der Begriff der Linearkombination wird für Vektoren wie üblich verwendet, d.h. ein Vektor $\mathbf{r}$ ist eine nicht-negative Linearkombination einer Menge von Vektoren R, falls eine Familie von nicht-negativen reellen Zahlen $(\lambda_{\mathbf{r}'})_{\mathbf{r}'\in R}$ existiert, wobei nur endliche viele Zahlen der Familie ungleich Null sind, so dass $\mathbf{r} = \sum_{\mathbf{r}'\in R} \lambda_{\mathbf{r}'} \cdot \mathbf{r}'$. Für Funktionen verwenden wir eine analoge Begriffsbildung. Es gilt der folgende Zusammenhang für Regionen.

LEMMA 4.4.11 *Lemma 4.4.11*
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T.
Sei R eine Menge von BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$. Dann ist eine nicht-negative Linearkombination $r \in \mathbb{N}^{2m+1}$ von R wieder eine BPO-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$.
Sei R eine Menge von Markenfluss-Regionen von $\mathcal{L}$. Dann ist eine Linearkombination $r : \bigcup_{(V,<,l)\in\mathcal{L}} <^* \to \mathbb{N}$ von R wieder eine Markenfluss-Region von $\mathcal{L}$.

BEWEIS: Das Resultat folgt aufgrund der Linearität von (BPOT) bzw. (ANF), (ZU) und (AB). □

Wichtig ist bei diesem Lemma jeweils die Einschränkung, dass die Linearkombinationen nur Werte aus $\mathbb{N}$ annehmen. Es ist auch zu beachten, dass bei Markenfluss-Regionen beliebige Linearkombinationen zugelassen sind. Nun definieren wir ein Erzeugendensystem auf der Regionenebene wie folgt.

DEFINITION 4.4.10 (ERZEUGENDENSYSTEM FÜR REGIONEN) *Definition 4.4.10 (Erzeugendensystem für Regionen)*
Sei $\mathcal{L}$ eine partielle Sprache mit endlicher Beschriftungsmenge T. Eine Menge R von BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ bzw. Markenfluss-Regionen von $\mathcal{L}$ heißt Erzeugendensystem, falls jede BPO-Transitions-Region von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ bzw. Markenfluss-Region von $\mathcal{L}$ eine nicht-negative Linearkombination von R ist.

Mit folgendem Lemma ergibt sich, dass durch ein solches Erzeugendensystem eine Erzeugendensystem-Repräsentation gegeben ist.

Lemma 4.4.12

LEMMA 4.4.12
Sei R *eine Menge von BPO-Transitions-Regionen von* $\mathcal{L}$ *bzgl.* $T = \{t_1, \ldots, t_m\}$. *Sei weiter* $\mathbf{r} = \sum_{\mathbf{r}' \in R} \lambda_{\mathbf{r}'} \cdot \mathbf{r}' \in \mathbb{N}^{2m+1}$ *eine nicht-negative Linearkombination von* R. *Dann gilt* $\overrightarrow{\mathbf{r}} = \sum_{\mathbf{r}' \in R} \lambda_{\mathbf{r}'} \cdot \overrightarrow{\mathbf{r}'}$.
Entsprechend gilt für eine Menge R *von Markenfluss-Regionen von* $\mathcal{L}$ *und eine nicht-negative Linearkombination* $r : \bigcup_{(V,<,l) \in \mathcal{L}} <^* \to \mathbb{N} = \sum_{r' \in R} \lambda_{r'} \cdot r'$ *von* R, *dass* $\overrightarrow{r} = \sum_{r' \in R} \lambda_{r'} \cdot \overrightarrow{r'}$.

BEWEIS: Für BPO-Transitions-Regionen folgt die Behauptung direkt aus den Definitionen. Für Markenfluss-Regionen muss beachtet werden, dass für jeden Knoten v einer $*$-Erweiterung einer BPO aus $\mathcal{L}$ für die Marken-Zuflüsse und Marken-Abflüsse $Zu_r(v) = \sum_{r' \in R} \lambda_{r'} \cdot Zu_{r'}(v)$ und $Ab_r(v) = \sum_{r' \in R} \lambda_{r'} \cdot Ab_{r'}(v)$ gilt. □

Korollar 4.4.1

KOROLLAR 4.4.1
Sei R *ein Erzeugendensystem der Menge der BPO-Transitions-Regionen von* $\mathcal{L}$ *bzgl.* $T = \{t_1, \ldots, t_m\}$ *bzw. Markenfluss-Regionen von* $\mathcal{L}$ *und sei* ST *die Menge aller zu einer Region aus* R *korrespondierenden zulässigen Stellentripel. Dann ist das markiertes S/T-Netz* (N, m_0), *dessen Menge von atomaren Teilnetzen der Menge aller von Stellentripeln aus* ST *definierten atomaren Netzen entspricht, eine Erzeugendensystem-Repräsentation des gesättigt zulässigen Netzes von* $\mathcal{L}$.

BEWEIS: Dieses Korollar lässt sich für beide Regionendefinitionen auf dieselbe Weise begründen. Nach Satz 4.3.2 bzw. Satz 4.3.5 gibt es für jedes zulässige Stellentripel eine Region, zu der das Stellentripel korrespondiert. Diese Region lässt sich als nicht-negative Linearkombination von R darstellen. Dann lässt sich nach Lemma 4.4.12 das betrachtete Stellentripel als eine entsprechende nicht-negative Linearkombination von ST darstellen. Folglich ist (N, m_0) eine Erzeugendensystem-Repräsentation. □

Der Vollständigkeit halber sei noch erwähnt, dass im Falle von BPO-Transitions-Regionen aufgrund der eins-zu-eins-Korrespondenz zwischen Regionen und zulässigen Stellentripeln auch die Umkehrung der letzten zwei Resultate gilt. Bei Markenfluss-Regionen ist dies allerdings nicht der Fall, da es für ein zulässiges Stellentripel viele verschiedene mögliche Markenfluss-Verteilungen geben kann.

Beispiel

BEISPIEL: *Ein minimales endliches Erzeugendensystem der Menge der BPO-Transitions-Regionen der partiellen Sprache aus Abbildung 4 bzgl. der Transitionsreihenfolge* a, b, c *ist durch die Vektoren* $(0,1,0,0,0,0,0)$, $(1,0,0,0,0,1,0)$, $(0,0,1,0,0,0,0)$, $(1,0,0,0,0,0,1)$, $(2,0,0,0,1,0,0)$, $(1,2,0,0,1,1,0)$, $(0,0,1,0,0,0,1)$, $(1,1,0,0,1,0,0)$, $(1,2,0,0,1,0,1)$, $(1,0,0,0,0,0,0)$ *und* $(0,0,0,1,0,0,0)$ *gegeben. Es definiert die Erzeugendensystem-Repräsentation aus Abbildung 73. Aufgrund der eins-zu-eins-Korrespondenz zwischen BPO-Transitions-Regionen und zulässigen Stellentripeln definiert ein minimales Erzeugendensystem auf der Ebene der BPO-Transitions-Regionen immer auch eine minimale Erzeugendensystem-Repräsentation.*
Ein minimales Erzeugendensystem der Menge der Markenfluss-Regionen der partiellen Sprache aus Abbildung 4 besteht aus 21 Regionen. Eine zugehörige Erzeugendensystem-Repräsentation enthält die Stellen, welche in Abbildung 73 dargestellt sind und 9 weitere Stellen. Es existiert hierbei eine Stelle, welche von zwei verschiedenen Regionen definiert wird.

Wir betrachten nun eine endliche partielle Sprache. Sowohl für BPO-Transitions-Regionen als auch für Markenfluss-Regionen lässt sich dann ein endliches Erzeugendensystem durch die Berechnung eines endlichen Erzeugendensystems eines polyedrischen Kegels berechnen. Hierzu werden die linear algebraischen Charakterisierungen der verschiedenen Regionen aus dem letzten Abschnitt verwendet.

LEMMA 4.4.13 *Lemma 4.4.13*
Sei $\mathcal{L}$ eine endliche partielle Sprache.

- *Sei die Beschriftungsmenge $T = \{t_1, \ldots, t_m\}$ geordnet und sei $R \subseteq \mathbb{N}^{2m+1}$. Die Menge R ist genau dann ein endliches Erzeugendensystem von BPO-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$, wenn R ein ganzzahliges endliches Erzeugendensystem des spitzen rationalen polyedrischen Kegels $\mathbf{A}_{\mathcal{L}}^{BPOT} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ ist.*
- *Sei eine Nummerierung der Menge $\bigcup_{(V,<,l)\in\mathcal{L}} <^* = \{e_1, \ldots, e_n\}$ der Kanten aller $*$-Erweiterungen der BPOs aus $\mathcal{L}$ gegeben und sei R eine Menge von globalen Markenfluss-Funktionen von L. Die Menge R ist genau dann ein endliches Erzeugendensystem von Markenfluss-Regionen von $\mathcal{L}$, wenn die Menge aller zu einer globalen Markenfluss-Funktion aus R bzgl. $\{e_1, \ldots, e_n\}$ korrespondierenden Vektoren ein ganzzahliges endliches Erzeugendensystem des spitzen rationalen polyedrischen Kegels $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ ist.*

BEWEIS: Das Resultat ergibt sich mit Lemma 4.3.8 bzw. Lemma 4.3.14 aus den entsprechenden Definitionen. □

Wichtig ist hierbei, dass die Lösungsmengen der beiden Ungleichungssysteme $\mathbf{A}_{\mathcal{L}}^{BPOT} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ und $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ entsprechend den Ausführungen in Abschnitt 3.4 einen spitzen rationalen polyedrischen Kegel definieren. Die Spitzheit wird jeweils durch die $\mathbf{x} \geqslant \mathbf{0}$-Restriktion sichergestellt. Ein ganzzahliges endliches Erzeugendensystems eines spitzen rationalen polyedrischen Kegels lässt sich, wie in Abschnitt 3.4 diskutiert, durch das Lösen des Extremalstrahl-Aufzählungs-Problems berechnen. Dabei wird sogar das bis auf die Wahl der Vektoren auf den Extremalstrahlen eindeutige minimale ganzzahlige endliche Erzeugendensystem erzeugt. Entsprechend dem letzten Lemma ist durch ein ganzzahliges endliches Erzeugendensystem von $\mathbf{A}_{\mathcal{L}}^{BPOT} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ bzw. $\mathbf{A}_{\mathcal{L}}^{MAR} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$ ein endliches Erzeugendensystem für die Menge der Regionen von $\mathcal{L}$ des betrachteten Regionentyps gegeben. Ein solches definiert nach Korollar 4.4.1 dann eine endliche Erzeugendensystem-Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$. Diese lässt sich dadurch konstruieren, dass für jede Region des Erzeugendensystems das korrespondierende zulässige Stellentripel zur durch die Beschriftungsmenge von $\mathcal{L}$ gegebenen Transitionsmenge hinzugefügt wird. Somit ergibt sich in Pseudo-Code der folgende Algorithmus zur Konstruktion einer endlichen Erzeugendensystem-Repräsentation.

ALGORITHMUS 4.4.3 (BERECHNUNG ERZEUGENDENSYSTEM-REPRÄSENTATION) *Algorithmus 4.4.3 (Berechnung Erzeugendensystem-Repräsentation)*

```
1  Eingabe: Partielle Sprache L
2  Ausgabe: Erzeugendensystem—Repräsentation von L
3  BEGIN
```

```
4    A_L^BPOT bzw. A_L^MAR :=
       L.regionenMatrix(BPOT bzw. MAR);
5    R := ganzzahligesErzeugendensystem
       (A_L^BPOT · x ⩾ 0, x ⩾ 0 bzw. A_L^MAR · x = 0, x ⩾ 0);
6    (N, m_0) := (∅, L.beschriftungsMenge(), ∅, ∅);
7    FORALL r ∈ R DO
8        (N, m_0).fuegeHinzu(r⃗);
9    ENDFORALL
10   RETURN (N, m_0);
11 END
```

Das Problem an diesem Algorithmus ist, dass die Größe eines minimalen endlichen Erzeugendensystems eines polyedrischen Kegels im schlechtesten Fall exponentiell in der Größe des entsprechenden Ungleichungssystems sein kann. In unserem Fall können die Extremalstrahlen des Kegels berechnet werden. Auch wenn deren Anzahl bei realen Beispielen meist nicht exponentiell wächst, so erfordert im schlechtesten Fall immer noch die Berechnung der Extremalstrahlen eine exponentielle Anzahl von Berechnungsschritten in der Anzahl der Strahlen (siehe Abschnitt 3.4). Somit kann es vorkommen, dass die Laufzeit des Algorithmus sehr problematisch ist. Ist die Anzahl der berechneten Extremalstrahlen exponentiell in der Größe der Eingabe $\mathcal{L}$, so ergibt sich auch eine entsprechend große Anzahl an Stellentripeln, welche zu dem zu synthetisierenden Netz hinzugefügt werden. Typischerweise liegt die Anzahl der Extremalstrahlen eines polyedrischen Kegels aber in einem praktisch noch verwendbaren Rahmen. Auch die Berechnung lässt sich in vielen Fällen in polynomieller Zeit in der Anzahl der Extremalstrahlen durchführen.

Beispiel

BEISPIEL: *Wenn wir Algorithmus 4.4.3 unter Verwendung eines entsprechenden Verfahrens zur Aufzählung der Extremalstrahlen eines Kegels auf die partielle Sprache aus Abbildung 4 anwenden, so ergeben sich die minimalen Erzeugendensysteme von Regionen, welche wir im letzten Beispiel diskutiert haben. Im Falle von BPO-Transitions-Regionen berechnet der Algorithmus somit das Netz aus Abbildung 73.*

In folgender Bemerkung werden Möglichkeiten, den betrachteten Algorithmus im Hinblick auf die Laufzeit und auf die Größe der konstruierten Erzeugendensystem-Repräsentation zu optimieren, diskutiert.

Bemerkung 4.4.4 (Optimierungsvorschläge)

BEMERKUNG 4.4.4 (OPTIMIERUNGSVORSCHLÄGE)
Die folgenden Optimierungen von Algorithmus 4.4.3 sind möglich.

- *Zuerst einmal spielt natürlich das verwendete Verfahren zum Auffinden eines ganzzahligen endlichen Erzeugendensystems für das betrachtete Ungleichungssysteme eine entscheidende Rolle für die Laufzeit des Algorithmus. Hier sollen, wie schon ausgeführt, Verfahren zur Lösung des Extremalstrahl-Aufzählungs-Problems verwendet werden. Diese stellen die effizienteste Möglichkeit zur Berechnung eines entsprechenden Erzeugendensystems dar und garantieren die Berechnung eines minimalen Erzeugendensystems. Der letztere Punkt ist für die Größe der resultierenden Erzeugendensystem-Repräsentation wichtig. Mögliche Lösungsverfahren für das Extremalstrahl-Aufzählungs-Problem wurden in Abschnitt 3.4 diskutiert. Es ist auch hier schwer zu sagen, welches Lösungsverfahren sich für unsere Zwecke am besten eignet. Für diese Fragestellung können allgemeine Erfahrungswerte über*

die verschiedenen Verfahren zu Rate gezogen werden, es ist aber auch sinnvoll, spezielle Tests durchzuführen. Entsprechend den Ausführungen in Abschnitt 3.4 spielt insbesondere der Grad der Degeneriertheit der Ungleichungssysteme eine wichtige Rolle. Normalerweise müssen wir hierbei degenerierte Systeme erwarten.

- *Daneben hat die Größe des betrachteten Ungleichungssystems einen wesentlichen Einfluss auf den Algorithmus. Diese ist entscheidend für die Laufzeit der Verfahren zur Lösung des Extremalstrahl-Aufzählungs-Problems. Außerdem ist die obere Schranke für die Anzahl der Extremalstrahlen, und damit auch für die Anzahl der Stellen des konstruierten Netzes, von der Größe des Ungleichungssystems abhängig. Die tatsächliche Anzahl der Extremalstrahlen ist aber nur von dem betrachteten Kegel abhängig. Daher bringt zwar eine Darstellung desselben Kegels durch ein kleineres Ungleichungssystem, z.B. durch das Entfernen redundanter Ungleichungen, in diesem Punkt keine Vorteile, ein Ungleichungssystem, welches einen anderen Kegel definiert, z.B. indem BPO-Transitions-Regionen anstelle von Markenfluss-Regionen betrachtet werden, kann aber zu einer Verbesserung führen. Insgesamt ist es sicherlich sinnvoll, eine Regionendefinition mit einer möglichst kleinen entsprechenden* $\mathbf{A}$*-Matrix zu wählen. Darüber hinaus können auch noch Optimierungsverfahren aus Abschnitt 4.3 genutzt werden, um die Größe der betrachteten* $\mathbf{A}$*-Matrix so weit wie möglich zu reduzieren.*

- *Typischerweise enthält eine Erzeugendensystem-Repräsentation viele implizite Stellen. Deswegen ist es hier besonders wichtig, in dem konstruierten Netz nach impliziten Stellen zu suchen und solche zu entfernen. In vielen Fällen kann die Netzgröße hierdurch deutlich reduziert werden. Entsprechende Verfahren sind in Bemerkung 4.4.2 beschrieben.*

- *Eine weitere interessante Möglichkeit besteht darin, nach der Berechnung der Erzeugendensystem-Repräsentation noch eine Schrittseparations-Repräsentation oder eine BPO-Separations-Repräsentation mit dem Vorgehen der letzten beiden Unterabschnitte zu erzeugen. Hierbei kann mithilfe der Erzeugendensystem-Repräsentation eine Verbesserung der Algorithmen der letzten beiden Unterabschnitte vorgenommen werden. Anstatt für jede falsche Fortsetzung ein Ungleichungssystem zu lösen, kann versucht werden, direkt ein bzgl. der falschen Fortsetzung entsprechend separierendes Stellentripel der Erzeugendensystem-Repräsentation zu finden. Gibt es ein solches, so wird es zum Netz hinzugefügt. Gibt es kein solches, so lässt sich leicht zeigen, dass es dann allgemein kein bzgl. der falschen Fortsetzung separierendes Stellentripel gibt. Die Existenz eines solchen Stellentripels lässt sich effizient prüfen, da hierzu nur entsprechende Aktiviertheitstests für die Stellentripel der Erzeugendensystem-Repräsentation nötig sind. Insgesamt besteht der Vorteil, über eine Erzeugendensystem-Repräsentation hinaus eine Schrittseparations-Repräsentation oder eine BPO-Separations-Repräsentation zu berechnen, darin, dass für die Separations-Repräsentationen die Beziehungen aus Lemma 4.4.1 bzw. Lemma 4.4.5 gelten. Dies bedeutet, dass wir, falls eine falsche Fortsetzung nicht entsprechend separierbar ist, direkt eine negative Antwort auf das Syntheseproblem geben können. Ist dies nicht möglich, so ist entweder* $\mathcal{L}$ *nicht schritt- bzw. fortsetzungsabgeschlossen oder das Syntheseproblem hat eine positive Antwort. Außerdem enthalten die so entstehenden*

Separations-Repräsentationen typischerweise auch wesentlich weniger Stellen als die ursprüngliche Erzeugendensystem-Repräsentation.

Beispiel

BEISPIEL: *Abbildung 73 zeigt, dass selbst eine minimale Erzeugendensystem-Repräsentation typischerweise viele implizite Stellen enthält. So können von den* 11 *Stellen des Netzes alle Stellen bis auf die vier Stellen, welche auch in dem Netz aus Abbildung 3 vorkommen, weggelassen werden, ohne dass sich das Verhalten ändert. Das Beispiel macht also deutlich, dass die Entfernung impliziter Stellen entsprechend des dritten Optimierungsvorschlages bei einer Erzeugendensystem-Repräsentation tatsächlich besonders wichtig ist.*

Die vorgestellten Optimierungsvorschläge erhalten die im schlechtesten Fall exponentielle Abhängigkeit der Anzahl der Extremalstrahlen der entsprechenden Kegel von der Größe der Eingabe $\mathcal{L}$. Daher bleibt auch bei Verwendung aller Optimierungsvorschläge im schlechtesten Fall die exponentielle Laufzeit des Algorithmus und die exponentielle Größe der resultierenden Netze bestehen.

4.5 ÜBEREINSTIMMUNGSTEST

Im letzten Abschnitt haben wir Algorithmen zur Erzeugung verschiedener Repräsentationen des gesättigt zulässigen Netzes einer endlichen partiellen Sprache $\mathcal{L}$ eingeführt. All diese Repräsentationen sind insofern zu Synthesezwecken geeignet, dass sie entweder ein positives Lösungsnetz für das Problem darstellen, oder aber das Syntheseproblem eine negative Antwort besitzt. Nach der Berechnung einer dieser Repräsentationen ist allerdings in den meisten Fällen nicht klar, ob die Repräsentation das Syntheseproblem positiv löst oder nicht. Einzig in der Situation, dass bei der Berechnung einer Separations-Repräsentation eine falsche Fortsetzung nicht verhindert werden kann, lässt sich direkt folgern, dass die Separations-Repräsentation nicht das spezifizierte Verhalten aufweist und es somit generell keine positive Lösung des Syntheseproblems gibt. Ansonsten ist es zur Lösung des Syntheseproblems erforderlich, nach der Berechnung einer der Repräsentationen des gesättigt zulässigen Netzes zu prüfen, ob das berechnete Netz das spezifizierte Verhalten aufweist oder nicht. Im positiven Fall stellt das Netz eine Lösung des Syntheseproblems dar. Im negativen Fall hat das Syntheseproblem eine negative Antwort. Für eine derartige Überprüfung stellen wir in diesem Abschnitt verschiedene mögliche Algorithmen vor, um in der betrachteten Situation die spezifizierte partielle Sprache und das Ablaufverhalten des synthetisierten Netzes auf Übereinstimmung zu testen.
Hierzu gibt es im Wesentlichen zwei mögliche Vorgehensweisen. Den Ausgangspunkt bildet eine der verschiedenen Repräsentationen (N, m_0) und die gegebene endliche partielle Sprache $\mathcal{L}$. Es soll getestet werden, ob $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$. Für alle Repräsentationen gilt $\mathfrak{Bpo}(N, m_0) \supseteq PS(\mathcal{L})$. Daher genügt es, entweder zu testen, ob jede bzgl. (N, m_0) aktivierte BPO in $PS(\mathcal{L})$ enthalten ist, oder umgekehrt zu testen, ob keine BPO, die nicht in $PS(\mathcal{L})$ enthalten ist, bzgl. (N, m_0) aktiviert ist. Die erste Möglichkeit bezeichnen wir als optimistischen Übereinstimmungstest, die zweite als pessimistischen Übereinstimmungstest. Algorithmen für derartige Tests werden in den folgenden Unterabschnitten diskutiert.

4.5.1 *Optimistischer Übereinstimmungstest*

Bei dem optimistischen Übereinstimmungstest werden im Prinzip alle bzgl. der berechneten Repräsentation des gesättigt zulässigen Netzes einer endlichen partiellen Sprache $\mathcal{L}$ aktivierten BPOs berechnet. Im Allgemeinen kann aber das Ablaufverhalten eines endlichen markierten S/T-Netzes unendlich sein. Daher wird in einem ersten Schritt des optimistischen Übereinstimmungstests die betrachtete Repräsentation derart modifiziert, dass die Endlichkeit des Ablaufverhaltens garantiert ist. Hierzu benötigen wir die folgenden zwei Lemmata.

Lemma 4.5.1 *Lemma 4.5.1*

Sei $\mathcal{L}$ eine endliche partielle Sprache mit Beschriftungsmenge T. Für jede Transition $t \in T$ und jede BPO $bpo = (V, <, l) \in \mathcal{L}$ betrachten wir die Anzahl der mit t beschrifteten Knoten $n^t_{bpo} = |V|_l(t)$ von bpo. Weiter betrachten wir die endliche obere Schranke $n^t = max\{n^t_{bpo} \mid bpo \in \mathcal{L}\}$ für die Anzahl der mit t beschrifteten Knoten einer BPO der endlichen partiellen Sprache $\mathcal{L}$. Nun gilt, dass das Stellentripel st^t mit $st^t_0 = n^t$, $^\circ st^t = \emptyset$, $st^t{}^\circ(t') = 0$ für alle $t' \in T \setminus \{t\}$ und $st^t{}^\circ(t) = 1$ zulässig bzgl. $\mathcal{L}$ ist.

Beweis: In dem atomaren Netz (N_{st^t}, m_{st^t}) sind per Definition alle Schrittfolgen $\sigma = \tau_1 \dots \tau_n$ mit $(\tau_1 + \dots + \tau_n)(t) \leqslant n_t$ aktiviert. Damit sind alle BPOs $bpo = (V, <, l)$ mit $|V|_l(t) \leqslant n_t$ aktiviert. Aus den Voraussetzungen folgt somit, dass $\mathcal{L} \subseteq \mathfrak{Bpo}(N_{st^t}, m_{st^t})$. □

Eine zu st^t gehörige Stelle beinhaltet n^t Marken, hat einen leeren Vorbereich und nur die Transition t in ihrem Nachbereich. Wird also st^t zu einem Netz hinzugefügt, so wird dadurch garantiert, dass die Transition t in dem Netz höchstens n^t-mal schalten kann.

Lemma 4.5.2 *Lemma 4.5.2*

Sei $\mathcal{L}$ eine endliche partielle Sprache mit Beschriftungsmenge T und sei (N, m_0), $N = (P, T, W)$, ein markiertes S/T-Netz, welches für jedes $t \in T$ das vom Stellentripel st^t definierte atomare Netz als Teilnetz enthält. Dann ist $\mathfrak{Bpo}(N, m_0)$ endlich.

Beweis: Sei $bpo = (V, <, l) \in \mathfrak{Bpo}(N, m_0)$. Dann gilt $|V|_l(t) \leqslant n^t$ für alle $t \in T$. Wäre dies nicht der Fall, so würde für $\sigma(bpo') = \tau_1 \dots \tau_n$, $bpo' \in Slin(bpo)$, gelten, dass $(\tau_1 + \dots + \tau_n)(t) > n^t$. Da (N, m_0) das Stellentripel st^t enthält, würde folgen, dass $\sigma(bpo')$ und damit auch bpo nicht aktiviert ist. Aus $|V|_l(t) \leqslant n^t$ für alle $t \in T$ lässt sich nun folgern, dass $|V| \leqslant \sum_{t \in T} n^t$. Die Anzahl aller BPOs über T mit einer durch eine Schranke begrenzten Anzahl an Knoten ist endlich. Damit ist auch $\mathfrak{Bpo}(N, m_0)$ endlich. □

Die Idee ist nun, zu einer gegebenen Repräsentation des gesättigt zulässigen Netzes die zulässigen Stellentripel st^t für $t \in T$ hinzuzufügen. Per Definition gilt, dass durch das Hinzufügen zulässiger Stellentripel zu einer Schrittseparations-Repräsentation bzw. BPO-Separations-Repräsentation bzw. Erzeugendensystem-Repräsentation wieder eine Schrittseparations-Repräsentation bzw. BPO-Separations-Repräsentation bzw. Erzeugendensystem-Repräsentation entsteht. Für die derart modifizierte Repräsentation (N, m_0) ist sichergestellt, dass sie endliches Ablaufverhalten hat. Normalerweise haben die berechneten Repräsentationen aber ohnehin endliches Ablaufverhalten, so dass eine entsprechende Modifikation gar nicht notwendig ist.

So ist das Ablaufverhalten einer Erzeugendensystem-Repräsentation immer endlich. Dies liegt daran, dass nach Lemma 4.5.1 die Stellentripel st^t zulässig sind und somit im gesättigt zulässigen Netz enthalten sind. Dieses hat also nach Lemma 4.5.2 endliches Ablaufverhalten. Nach Lemma 4.4.10 und Lemma 4.3.3 hat schließlich eine Erzeugendensystem-Repräsentation dasselbe Ablaufverhalten wie das gesättigt zulässige Netz. Für die zwei Separations-Repräsentationen gilt nach Lemma 4.4.2 bzw. Lemma 4.4.6 und Lemma 4.3.3, dass sie in dem Fall, dass alle entsprechenden falschen Fortsetzungen separiert werden können, auch dasselbe Ablaufverhalten wie das gesättigt zulässige Netz aufweisen. In all diesen Fällen ist also ein endliches Ablaufverhalten garantiert. In der Situation schließlich, dass sich bei der Berechnung einer der Separations-Repräsentationen nicht alle falschen Fortsetzungen ausschließen lassen, ist ein Übereinstimmungstest eigentlich gar nicht mehr notwendig, da dann nach Lemma 4.4.1 bzw Lemma 4.4.5 ohnehin $PS(\mathcal{L}) \neq \mathfrak{Bpo}(N, m_0)$ gilt und somit eine negative Antwort auf das Syntheseproblem gegeben werden kann.

Beispiel

BEISPIEL: *Für die partielle Sprache aus Abbildung 4 entspricht die Stelle p_1 des Netzes aus Abbildung 3 dem Stellentripel st^a, da maximal zwei a-Ereignisse in einer der BPOs vorkommen. Die Stelle verhindert, dass a mehr als zweimal schalten kann. Die Stelle p_3 entspricht dem Stellentripel st^b. Wie im Beispiel dieses Netzes ist es normalerweise nicht nötig, Stellentripel st^t zu einem gegebenen Repräsentationsnetz hinzuzufügen, da ein Repräsentationsnetz meist ohnehin endliches Verhalten hat. Unendliches Verhalten kann nur entstehen, wenn bestimmte falsche Fortsetzungen nicht ausgeschlossen werden können. Aber selbst dann ist das Verhalten wie im Beispiel der partiellen Sprache aus Abbildung 54 und der zugehörigen Schrittseparations-Repräsentation aus Abbildung 57 üblicherweise endlich.*

In jedem Fall können wir im Weiteren davon ausgehen, dass ein Repräsentationsnetz (N, m_0) mit endlichem $\mathfrak{Bpo}(N, m_0)$ gegeben ist. Es soll nun entschieden werden, ob $\mathfrak{Bpo}(N, m_0) \subseteq PS(\mathcal{L})$. Dies ist genau dann der Fall, wenn $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$. Dies ist wiederum äquivalent dazu, dass das Syntheseproblem eine positive Antwort besitzt.
Da $\mathfrak{Bpo}(N, m_0)$ endlich ist, kann es berechnet werden. Um dann zu testen, ob eine BPO $bpo \in \mathfrak{Bpo}(N, m_0)$ auch in $PS(\mathcal{L})$ enthalten ist, ist es algorithmisch nötig, für jede BPO $bpo' \in PS(\mathcal{L})$ zu prüfen, ob bpo isomorph zu bpo' ist. Dies liegt daran, dass wir formal isomorphe BPOs nicht unterschieden, algorithmisch aber nur Repräsentanten betrachten können. Im Prinzip müssen wir also $\mathfrak{Bpo}(N, m_0)$ berechnen und dann für jedes $bpo \in \mathfrak{Bpo}(N, m_0)$ prüfen, ob bpo isomorph zu einer BPO aus $PS(\mathcal{L})$ ist. Um dies effizient durchzuführen, können wir berücksichtigen, dass eine BPO $bpo' \in PS(bpo)$ hierbei ignoriert werden kann. Es genügt in dieser Situation, nur bpo zu betrachten, da in dem Fall, dass $bpo' \notin PS(\mathcal{L})$ gilt, auch $bpo \notin PS(\mathcal{L})$ folgt. Es ist also ausreichend, eine Teilmenge von $\mathfrak{Bpo}(N, m_0)$ zu betrachten, welche die Repräsentationssprache $Rep(\mathfrak{Bpo}(N, m_0))$ enthält. Eine solche Menge von BPOs ist entsprechend Satz 3.3.1 durch die endliche Menge der Prozess-BPOs maximaler Länge von (N, m_0) gegeben. Diese lässt sich über die endliche Menge von Prozessnetzen maximaler Länge von (N, m_0) berechnen. Da $\mathfrak{Bpo}(N, m_0) \supseteq PS(\mathcal{L})$, ist eine Prozess-BPO maximaler Länge niemals ein echtes Präfix einer BPO aus $PS(\mathcal{L})$. Daher kann durch einen Test, ob eine solche Prozess-BPO in $Seq(\mathcal{L})$ enthalten ist, geprüft werden, ob sie in $PS(\mathcal{L})$ enthalten ist. Insgesamt genügt

es also zu prüfen, ob jede Prozess-BPO maximaler Länge von (N, m_0) isomorph zu einer BPO aus $\mathrm{Seq}(\mathcal{L})$ ist. In Pseudo-Code ergibt sich der folgende Algorithmus.

Algorithmus 4.5.1 (Optimistischer Übereinstimmungstest)

Algorithmus 4.5.1 (Optimistischer Übereinstimmungstest)

```
1   Eingabe: Repräsentationsnetz (N, m_0) und partielle
      Sprache L
2   Ausgabe: Wahrheitswert, ob Bpo(N, m_0) = PS(L) gilt
3   BEGIN
4       FORALL t ∈ T DO
5           (N, m_0).fuegeHinzu(st^t);
6       ENDFORALL
7       ProzessBPOs :=
          (N, m_0).prozessBPOsMaximalerLaenge();
8       FORALL bpo ∈ ProzessBPOs DO
9           IF Seq(L).beinhaltetNicht(bpo) THEN
10              RETURN false;
11          ENDIF
12      ENDFORALL
13      RETURN true;
14  END
```

Beispiel: *Im Beispiel des Netzes aus Abbildung 3 und der partiellen Sprache aus Abbildung 4 kann, wie gesagt, das Hinzufügen der Stellentripel st^t übersprungen werden. Die Prozess-BPOs maximaler Länge des Netzes lassen sich dann beispielsweise durch die Berechnung der Prozessnetze maximaler Länge konstruieren. Dies ist in Abbildung 48 illustriert. Für jede der abgebildeten Prozess-BPOs muss dann geprüft werden, ob es sich um eine Sequentialisierung einer der beiden BPOs bpo_1 oder bpo_2 aus Abbildung 4 handelt. Die erste BPO entspricht hierbei bpo_1, die zweite entspricht bpo_2 und die dritte sequentialisiert bpo_2. Somit ergibt der Übereinstimmungstest eine positive Antwort.*
Für die partielle Sprache aus Abbildung 54 und das zugehörige Repräsentationsnetz aus Abbildung 57 ergibt der optimistische Übereinstimmungstest eine negative Antwort. Dies liegt daran, dass die totale Prozess-BPO, welche zur Schaltfolge bcaa korrespondiert, weder eine Sequentialisierung der ersten noch der zweiten BPO aus Abbildung 54 ist.

Beispiel

Es gibt verschiedene Algorithmen zur Berechnung der Menge der maximalen Prozess-BPOs eines S/T-Netzes. Insbesondere wurden etliche Algorithmen zur Konstruktion einer sog. Entfaltung vorgeschlagen [91, 175, 169, 139, 141, 92, 140]. Aus einer Entfaltung lässt sich dann die Menge der maximalen Prozessnetze und daraus die Menge der maximalen Prozess-BPOs gewinnen. Im Allgemeinen kann die Anzahl der (maximalen) Prozess-BPOs exponentiell in der Größe des S/T-Netzes sein. Dementsprechend erfordert die Berechnung aller (maximaler) Prozess-BPOs im schlechtesten Fall einen exponentiellen Aufwand. Die Anzahl der (maximalen) Prozess-BPOs spielt dann auch im Weiteren eine wichtige Rolle, da für jede Prozess-BPO Isomorphietests durchgeführt werden. In unserer speziellen Situation erwarten wir allerdings, dass die Anzahl der maximalen Prozess-BPOs von (N, m_0) ungefähr mit der Größe von $\mathcal{L}$ übereinstimmt. Dies liegt daran, dass im Falle, dass es eine positive Lösung für das Syntheseproblem gibt,

$\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$ gilt und im negativen Fall normalerweise immer noch $\mathfrak{Bpo}(N, m_0) \approx PS(\mathcal{L})$ gilt. Natürlich enthält die Menge der Prozess-BPOs in den meisten Fällen zumindest einige Sequentialisierungen, dies kann aber auch auf die Spezifikation $\mathcal{L}$ zutreffen. Wird also der Aufwand zur Berechnung und Betrachtung aller (maximaler) Prozess-BPOs im Verhältnis zur Eingabe $\mathcal{L}$ betrachtet, so ist er häufig gutartig.
Dennoch bleibt auch dann noch das Problem, dass entsprechende Isomorphietests durchgeführt werden müssen. Für eine Prozess-BPO eines der berechneten Prozesse und eine BPO aus $\mathcal{L}$ kann direkt geprüft werden, ob die Prozess-BPO isomorph zu einer Sequentialisierung der BPO aus $\mathcal{L}$ ist. Diese Prüfung stellt ein spezielles Graphisomorphieproblem dar. Es wird allgemein angenommen, dass Graphisomorphieprobleme eine eigene Komplexitätsklasse zwischen P und NP bilden. Das übliche Verfahren zur Lösung von Graphisomorphieproblemen besteht in der Verwendung von Backtracking-Algorithmen, welche eine Menge von möglichen Isomorphismen konstruieren. Für jeden solchen möglichen Isomorphismus wird geprüft, ob es tatsächlich ein Isomorphismus ist. In unserem Fall kann auch genau so vorgegangen werden, wobei berücksichtigt werden muss, dass nicht geprüft wird, ob es sich um einen exakten Isomorphismus handelt, sondern, ob die entsprechende Sequentialisierungseigenschaft erfüllt ist, d.h. es können in der einen BPO auch mehr Kanten vorhanden sein. Dies ist allerdings nicht aufwändiger. Insgesamt hängt die Effizient eines solchen Graphisomorphietests davon ab, in wie weit sich die Menge möglicher Isomorphismen im Rahmen des Backtracking-Algorithmus durch ein effizientes „branch-and-bound"-Verfahren einschränken lässt. Hierzu sind für spezielle Typen von Graphen verschiedenste effiziente Strategien möglich. Generell lassen sich Graphisomorphieprobleme in vielen Fällen effizient lösen.
In folgender Bemerkung werden einige Möglichkeiten, um den betrachteten Algorithmus möglichst effizient zu gestalten, diskutiert.

Bemerkung 4.5.1 (Optimierungsvorschläge)

BEMERKUNG 4.5.1 (OPTIMIERUNGSVORSCHLÄGE)
Die folgenden Optimierungen von Algorithmus 4.5.1 sind möglich.

- *Eine wesentliche Rolle für die Effizienz von Verfahren zur Berechnung von Prozess-BPOs spielt natürlich die Größe des betrachteten Netzes (N, m_0). Hier ist es wichtig, im Rahmen der Überlegungen des letzten Abschnittes ein Verfahren zur Berechnung einer Repräsentation des gesättigt zulässigen Netzes einschließlich entsprechender Optimierungen zu wählen, welches ein möglichst kleines Netz erzeugt. Es ist zu beachten, dass auch die Anzahl der maximalen Prozess-BPOs verschiedener Repräsentationsnetze variieren kann. Dies hat einen wesentlichen Einfluss auf die Laufzeit des Übereinstimmungstests, da für jede Prozess-BPO entsprechende Isomorphietests notwendig sind.*

- *Durch das Hinzufügen der Stellentripel st^t wird das zu untersuchende Netz vergrößert. Wir haben gezeigt, dass dieser Schritt mit einer entsprechend genaueren Betrachtung des zuvor berechneten Repräsentationsnetzes in jedem Fall übersprungen werden kann.*

- *Entscheidend ist der verwendete Algorithmus zur Berechnung aller maximaler Prozess-BPOs des betrachteten S/T-Netzes. In diesem Kontext haben wir in [34] zwei neue Entfaltungs-Algorithmen vorgestellt, welche sich besonders gut zur direkten Berechnung entsprechender Prozess-*

BPOs von S/T-Netzen eignen (vgl. Bemerkung 4.3.3). Wir haben gezeigt, dass sie in diesem Rahmen im Vergleich zu klassischen Entfaltungs-Verfahren [91, 175, 169, 139, 141, 92, 140] wesentlich schneller sind und einen geringeren Speicherverbrauch aufweisen. Diese Verfahren verwenden das in Unterabschnitt 4.3.3 eingeführte Konzept des Markenfluss-Prozesses. Insbesondere wird also durch die Betrachtung von Markenflüssen von der Individualität von Marken in Entfaltungen bzw. Prozessnetzen abstrahiert. Auf diese Weise kann die Berechnung von Markenfluss-Prozessen und über die Korrespondenz von Markenfluss-Prozessen und Prozessnetzen dann auch die Berechnung von Prozess-BPOs effizient durchgeführt werden.

- *Da der Algorithmus abbricht, sobald einer der Tests in Zeile 9 positiv ist, spielt die Reihenfolge der Abarbeitung der Prozess-BPOs in der Schleife eine Rolle. Hier kann mit entsprechenden Heuristiken versucht werden, eine Reihenfolge zu finden, welche früh zu einem Abbruch führt. Insbesondere kann es sinnvoll sein, noch nicht sofort alle Prozess-BPOs zu berechnen, sondern erst einmal nur solche zu betrachten, welche nach der Heuristik Kandidaten für einen Abbruch sind.*

- *Schließlich ist die Effizienz des verwendeten Verfahrens für die notwendigen Isomorphietests sehr wichtig. Hier sollte, wie oben vorgeschlagen, ein Backtracking-Algorithmus verwendet werden. Dabei lässt sich beispielsweise die folgende für BPOs sehr restriktive „branch-and-bound"-Strategie, um möglichst wenig mögliche Isomorphismen zu berücksichtigen, verwenden. Wir identifizieren geeignete Äquivalenzklassen von Knoten in den zwei zu betrachtenden BPOs und berücksichtigen dann nur solche Isomorphismen, welche Knoten einer Äquivalenzklasse der ersten BPO auf Knoten der zweiten BPO, welche einer entsprechenden Äquivalenzklasse angehören, abbilden. Allein aufgrund der Betrachtung der Knotenanzahlen in den Äquivalenzklassen lässt sich dann häufig schon eine negative Antwort für den Isomorphietest geben. Auch sonst müssen bei geschickter Wahl der Äquivalenzklassen typischerweise nur ganz wenige mögliche Isomorphismen betrachtet werden. Die Äquivalenzklassen können beispielsweise die Beschriftungen eines Knotens sowie die Beschriftungen der Knoten im Vor- und im Nachbereich des Knotens berücksichtigen. Die Betrachtung der Vor- und Nachbereiche ist für BPOs besonders effektiv, da BPOs transitiv sind und daher häufig viele Kanten besitzen. Es sind auch Verfahren denkbar, welche die azyklische Struktur von BPOs ausnutzen, um die zu überprüfenden Isomorphismen einzuschränken bzw. um festzustellen, dass für ein Paar von BPOs kein entsprechender Isomorphismus existiert. Schließlich ist es durch eine geeignete Sortierung der BPOs aus $\mathfrak{L}$ nach entsprechenden Kriterien außerdem möglich, die Isomorphietests für jede Prozess-BPO durch ein binäres Suchverfahren auf bestimmte BPOs aus $\mathfrak{L}$ zu beschränken.*

- *Weiter ist der optimistische Übereinstimmungstest schneller, wenn aus $\mathfrak{L}$ möglicherweise vorhandene Präfixe und Sequentialisierungen entfernt werden. Durch das Weglassen von Präfixen reduziert sich die Menge $\mathrm{Seq}(\mathfrak{L})$ und somit auch die Anzahl der erforderlichen Isomorphietests. Auch das Weglassen von Sequentialisierungen reduziert die Anzahl der erforderlichen Isomorphietests, da es nicht notwendig ist, Sequentialisierungen explizit zu betrachten, d.h. algorithmisch wird $\mathfrak{L}$ und nicht $\mathrm{Seq}(\mathfrak{L})$ verwendet. Generell genügt es auch hier, nur die BPOs aus $\mathrm{Rep}(\mathfrak{L})$ anstatt aller BPOs aus $\mathfrak{L}$ zu berücksichtigen.*

Beispiel

BEISPIEL: *Wird ein gewöhnlicher Entfaltungsalgorithmus auf das Netz aus Abbildung 3 angewandt, so entsteht ein Verzweigungsprozess mit 32 Ereignissen. Eine Markenfluss-Entfaltung des Netzes besitzt dahingegen nur 8 Ereignisse. Dies zeigt das Einsparungspotential der Verfahren aus [34] im Rahmen des dritten Optimierungsvorschlages.*
Betrachten wir die Isomorphietests für die Prozess-BPOs aus Abbildung 48 und die partielle Sprache aus Abbildung 4, so wird unmittelbar klar, dass allein durch die Berücksichtigung der Anzahlen der Knoten der BPOs ausgeschlossen werden kann, dass die erste Prozess-BPO eine Sequentialisierung von bpo_2 *ist und dass die zweite und dritte Prozess-BPO eine Sequentialisierung von* bpo_1 *ist. Bei der Überprüfung, ob nun beispielsweise die dritte Prozess-BPO eine Sequentialisierung von* bpo_2 *ist, ziehen wir entsprechend des fünften Optimierungsvorschlages nur Bijektionen zwischen den Knotenmengen in Betracht, welche gleich-beschriftete Knoten aufeinander abbilden. Außerdem sollen noch die Vor- und Nachbereiche der Knoten berücksichtigt werden. Da das zweite* a*-Ereignis von* bpo_2 *einen* a*-Knoten im Vorbereich hat, muss dies auch für einen entsprechenden* a*-Knoten einer Sequentialisierung gelten. Damit kommt nur der zweite* a*-Knoten der Prozess-BPO in Frage. Somit werden durch das vorgeschlagene „branch-and-bound"-Vorgehen bis auf eine Bijektion alle möglichen bijektiven Abbildungen zwischen den Knotenmengen der beiden BPOs ausgeschlossen. Diese verbleibende Bijektion muss dann genauer betrachtet werden. Sie definiert in diesem Fall auch tatsächlich eine entsprechende Sequentialisierungsbeziehung.*

Obwohl der optimistische Übereinstimmungstest unter Verwendung geeigneter Optimierungen in vielen Fällen sehr effizient ist, ergibt sich im schlechtesten Fall aufgrund der Berechnung aller Prozess-BPOs maximaler Länge des betrachteten Netzes und der Isomorphietests eine exponentielle Laufzeit.

4.5.2 *Pessimistischer Übereinstimmungstest*

Die zweite Möglichkeit, um $\mathfrak{Bpo}(N, m_0) \subseteq \mathrm{PS}(\mathcal{L})$ für die berechnete Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}$ zu prüfen, ist, für jede nicht in $\mathrm{PS}(\mathcal{L})$ enthaltene BPO zu testen, ob sie in (N, m_0) aktiviert ist. Das Problem hierbei ist, dass es unendlich viele solche BPOs gibt. Daher soll eine endliche Menge $\mathrm{PS}(\mathcal{L})^c$ definiert werden, welche alle BPOs im Komplement von $\mathrm{PS}(\mathcal{L})$ repräsentiert.
Die endliche Menge $\mathrm{PS}(\mathcal{L})^c$ soll also derart definiert sein, dass $\mathrm{PS}(\mathcal{L})^c \cap \mathrm{PS}(\mathcal{L}) = \emptyset$ gilt und folgende zentrale Eigenschaft erfüllt ist. Falls keine BPO aus $\mathrm{PS}(\mathcal{L})^c$ in (N, m_0) aktiviert ist, dann ist gar keine BPO, welche nicht in $\mathrm{PS}(\mathcal{L})$ enthalten ist, in (N, m_0) aktiviert, d.h. das Ablaufverhalten von (N, m_0) enthält nur BPOs aus $\mathrm{PS}(\mathcal{L})$. Da $\mathrm{PS}(\mathcal{L})^c$ endlich ist, kann effektiv geprüft werden ob keine BPO aus $\mathrm{PS}(\mathcal{L})^c$ in (N, m_0) aktiviert ist, indem die Aktiviertheit jeder einzelnen BPO aus $\mathrm{PS}(\mathcal{L})^c$ getestet wird. In dem Fall, dass alle BPOs aus $\mathrm{PS}(\mathcal{L})^c$ nicht in (N, m_0) aktiviert sind, lässt sich aus der zentralen Eigenschaft von $\mathrm{PS}(\mathcal{L})^c$ folgern, dass $\mathfrak{Bpo}(N, m_0) \subseteq \mathrm{PS}(\mathcal{L})$ und somit $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$ gilt. Falls eine BPO aus $\mathrm{PS}(\mathcal{L})^c$ in (N, m_0) aktiviert ist, so folgt $\mathfrak{Bpo}(N, m_0) \neq \mathrm{PS}(\mathcal{L})$, da eine solche BPO in $\mathfrak{Bpo}(N, m_0)$ enthalten ist, aber wegen $\mathrm{PS}(\mathcal{L})^c \cap \mathrm{PS}(\mathcal{L}) = \emptyset$ nicht in $\mathrm{PS}(\mathcal{L})$ enthalten ist. Insgesamt zeigen diese Überlegungen somit, wie basierend auf einer geeigneten Definition einer Repräsentation $\mathrm{PS}(\mathcal{L})^c$ des Komplements von $\mathrm{PS}(\mathcal{L})$ ein effektiver Übereinstimmungstest mög-

lich ist. Es bleibt noch, eine entsprechende Definition einer solchen Repräsentationsmenge $PS(\mathcal{L})^c$ zu entwickeln.
Ein zu den Ausführungen des letzten Absatzes sehr ähnlicher Übereinstimmungstest wurde auch schon in [153] vorgeschlagen. Dieser Übereinstimmungstest lässt sich auf die Ideen dieses Abschnittes zurückführen. Er ergibt sich durch die folgende Definition der Menge $PS(\mathcal{L})^c$. Eine BPO ist genau dann in der Menge $PS(\mathcal{L})^c$, wenn sie

- durch Weglassen einer Gerüstkante von einer BPO aus $PS(\mathcal{L})$ mit minimaler Ordnung entsteht oder
- die Form $bpo; bpo'$ hat, wobei $bpo \in PS(\mathcal{L})$ und $bpo' = (V', <', l')$ mit $|V'| = 1$, und nicht in $PS(\mathcal{L})$ enthalten ist.

Wir zeigen jedoch nun, dass die Betrachtung dieser BPOs nicht ausreicht, um die diskutierte zentrale Eigenschaft für $PS(\mathcal{L})^c$ sicherzustellen. In folgendem Beispiel sind für die Definition von $PS(\mathcal{L})^c$ aus [153] alle BPOs aus $PS(\mathcal{L})^c$ nicht in (N, m_0) aktiviert und dennoch gilt $\mathfrak{Bpo}(N, m_0) \neq PS(\mathcal{L})$.

Beispiel

BEISPIEL: *Wir betrachten die partielle Sprache $\mathcal{L}$ aus Abbildung 74 links. Es ist zu beachten, dass $PS(\mathcal{L})$ nicht fortsetzungsabgeschlossen ist und es somit kein Netz mit dem spezifizierten Ablaufverhalten geben kann. Es lässt sich zeigen, dass das Netz aus Abbildung 74, Mitte, dasselbe Verhalten wie das gesättigt zulässige Netz von $\mathcal{L}$ hat. Somit hat das Netz dasselbe Verhalten wie eine Erzeugendensystem-Repräsentation (eine solche wird in [153] betrachtet). Außerdem stellt es eine Schrittseparations-Repräsentation und eine BPO-Separations-Repräsentation dar. Wir können also annehmen, dass dieses Netz die für einen Übereinstimmungstest zu betrachtende Repräsentation (N, m_0) ist. Es lässt sich leicht nachprüfen, dass für die Definition von $PS(\mathcal{L})^c$ aus [153] alle BPOs aus $PS(\mathcal{L})^c$ nicht in (N, m_0) aktiviert sind. Dennoch ist die BPO aus Abbildung 74 rechts in (N, m_0) aktiviert, obwohl sie nicht in $PS(\mathcal{L})$ enthalten ist.*

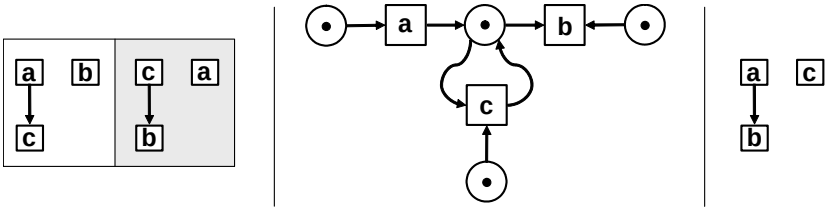

Abbildung 74: Gegenbeispiel zu Satz 6.2.26 in [153].

Das Beispiel zeigt also, dass die Definition von $PS(\mathcal{L})^c$ aus [153] nicht geeignet gewählt ist. Insbesondere stellt das Beispiel ein Gegenbeispiel zu Satz 6.2.26 in [153] dar. Dies bedeutet, dass das für den Übereinstimmungstest aus [153] zentrale Ergebnis nicht gilt. Somit ist der dort vorgestellte Übereinstimmungstest nicht korrekt.
Diese Überlegungen zeigen, dass bei der Wahl von $PS(\mathcal{L})^c$ sehr sorgfältig vorgegangen werden muss. Vor allem besteht die Gefahr, die Menge $PS(\mathcal{L})^c$ zu klein zu wählen. Das Problem an der Definition von $PS(\mathcal{L})^c$ aus [153] ist, dass nur das vollständige Anhängen eines Knotens an eine BPO aus $PS(\mathcal{L})$ nicht ausreicht. Es muss auch die Möglichkeit betrachtet werden, einen Knoten nur an einen Teil einer BPO aus $PS(\mathcal{L})$ anzuhängen.

Beispiel

BEISPIEL: *Beispielsweise lässt sich die BPO aus Abbildung 74 rechts konstruieren, indem ein* b*-Ereignis nur an das* a*-Ereignis des aus einem* a*- und einem* c*-Ereignis bestehenden Präfixes der zweiten BPO der partiellen Sprache aus Abbildung 74 angehängt wird. Sie ist in diesem Fall also in der Menge* $\mathrm{PS}(\mathcal{L})^c$ *enthalten. Ein Test zeigt dann die Aktiviertheit der BPO in dem betrachteten Netz. Somit ergibt sich bei einem entsprechenden Übereinstimmungstest korrekterweise eine negative Antwort.*

Die Idee zur Definition von $\mathrm{PS}(\mathcal{L})^c$ ist daher, jede BPO aus $\mathrm{PS}(\mathcal{L})$ entsprechend allen Möglichkeiten um einen Knoten zu erweitern, d.h. ein Knoten kann an ein beliebiges Präfix der BPO angehängt werden. Falls eine solche BPO nicht in $\mathrm{PS}(\mathcal{L})$ enthalten ist, so soll sie in $\mathrm{PS}(\mathcal{L})^c$ enthalten sein. Wir stellen im Folgenden eine entsprechende Definition von $\mathrm{PS}(\mathcal{L})^c$ vor und entwickeln darauf aufbauend einen zugehörigen Übereinstimmungstest.

Definition 4.5.1 (Repräsentation des Komplements)

DEFINITION 4.5.1 (REPRÄSENTATION DES KOMPLEMENTS)
Sei $\mathcal{L}$ *eine endliche partielle Sprache mit Beschriftungsmenge* T. *Wir definieren* $\mathrm{PS}(\mathcal{L})^c = \{\mathrm{bpo} \notin \mathrm{PS}(\mathcal{L}) \mid \mathrm{bpo} = (V' \cup \{v_t\}, <' \cup <_t, l' \cup (v_t, t))$ *wobei* $\mathrm{bpo}' = (V', <', l') \in \mathrm{PS}(\mathcal{L}), t \in T, v_t \notin V'$ *und* $<_t = V'' \times \{v_t\}$ *für ein Präfix* $\mathrm{bpo}'' = (V'', <'', l'')$ *von* $\mathrm{bpo}'\}$ *als Repräsentation des Komplements des von* $\mathcal{L}$ *spezifizierten Ablaufverhaltens.*

Beispiel

BEISPIEL: *Wir betrachten die partielle Sprache* $\mathcal{L}$ *aus Abbildung 4. Abbildung 75 zeigt die durch Anhängen eines Ereignisses an die links dargestellte BPO aus* $\mathrm{PS}(\mathcal{L})$ *entstehenden Elemente von* $\mathrm{PS}(\mathcal{L})^c$. *Beispielsweise ergibt sich das erste Element durch Anhängen eines* a*-Ereignisses an das leere Präfix der BPO. Insgesamt enthält* $\mathrm{PS}(\mathcal{L})^c$ *natürlich viele weitere BPOs. Diese ergeben sich durch entsprechende Betrachtungen für die weiteren BPOs aus* $\mathrm{PS}(\mathcal{L})$.

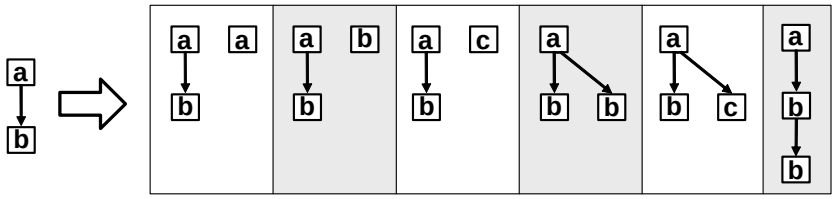

Abbildung 75: Konstruktion von $\mathrm{PS}(\mathcal{L})^c$.

Folgendes Lemma zeigt, dass diese Definition von $\mathrm{PS}(\mathcal{L})^c$ für unsere Zwecke geeignet ist.

Lemma 4.5.3

LEMMA 4.5.3
Sei $\mathcal{L}$ *eine endliche partielle Sprache mit Beschriftungsmenge* T *und sei* (N, m_0), $N = (P, T, W)$, *ein markiertes S/T-Netz mit* $\mathfrak{Bpo}(N, m_0) \supseteq \mathrm{PS}(\mathcal{L})$. *Falls* $\mathrm{PS}(\mathcal{L})^c \cap \mathfrak{Bpo}(N, m_0) = \emptyset$, *so folgt* $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$.

BEWEIS: Angenommen es gibt eine BPO $\mathrm{bpo} \in \mathfrak{Bpo}(N, m_0)$, so dass $\mathrm{bpo} \notin \mathrm{PS}(\mathcal{L})$. Dann gibt es ein bzgl. $\lhd$ maximales (möglicherweise auch leeres) Präfix $\mathrm{bpo}' = (V', <', l')$ von bpo mit $\mathrm{bpo}' \in \mathrm{PS}(\mathcal{L})$. Da bpo' ein echtes Präfix von bpo ist, gibt es ein weiteres Präfix bpo_e von bpo, welches bpo' um einen Knoten erweitert. Es gibt also ein Präfix von bpo der Form $\mathrm{bpo}_e = (V' \cup \{v_t\}, <' \cup <_t, l' \cup (v_t, t))$, wobei $t \in T, v_t \notin V'$ und $<_t = V'' \times \{v_t\}$ für ein Präfix $\mathrm{bpo}'' = (V'', <'', l'')$ von bpo'. Aufgrund der Maximalitätseigenschaft von bpo' und $\mathrm{bpo}' \lhd \mathrm{bpo}_e$ folgt, dass $\mathrm{bpo}_e \notin \mathrm{PS}(\mathcal{L})$. Damit gilt, dass $\mathrm{bpo}_e \in \mathrm{PS}(\mathcal{L})^c$. Da

bpo $\in \mathfrak{Bpo}(N, m_0)$ und $bpo_e \in$ Pref(bpo), folgt außerdem $bpo_e \in \mathfrak{Bpo}(N, m_0)$. Es folgt $bpo_e \in PS(\mathcal{L})^c \cap \mathfrak{Bpo}(N, m_0)$ und somit ergibt sich ein Widerspruch zu $PS(\mathcal{L})^c \cap \mathfrak{Bpo}(N, m_0) = \emptyset$. □

In einem entsprechenden Übereinstimmungstest muss also für jede BPO aus $PS(\mathcal{L})^c$ getestet werden, ob sie in der betrachteten Repräsentation (N, m_0) aktiviert ist. Ist dies für eine BPO der Fall, so ergibt sich aus $PS(\mathcal{L})^c \cap PS(\mathcal{L}) = \emptyset$, dass $\mathfrak{Bpo}(N, m_0) \neq PS(\mathcal{L})$. Ist dies dahingegen für keine BPO der Fall, so folgt nach Lemma 4.5.3 der Zusammenhang $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$. In Pseudo-Code ergibt sich der folgende Algorithmus.

Algorithmus 4.5.2 (Pessimistischer Übereinstimmungstest)

Algorithmus 4.5.2 (Pessimistischer Übereinstimmungstest)

```
1   Eingabe: Repräsentationsnetz (N, m0) und partielle
      Sprache L
2   Ausgabe: Wahrheitswert, ob Bpo(N, m0) = PS(L) gilt
3   BEGIN
4       PS(L)^c := L.repraesentationDesKomplements();
5       FORALL bpo ∈ PS(L)^c DO
6           IF bpo.istAktiviertIn(N, m0) THEN
7               RETURN false;
8           ENDIF
9       ENDFORALL
10      RETURN true;
11  END
```

Beispiel

Beispiel: *Für die partielle Sprache $\mathcal{L}$ aus Abbildung 4 und das zugehörige Repräsentationsnetz aus Abbildung 3 wird bei dem pessimistischen Übereinstimmungstest zuerst, wie im letzten Beispiel dargestellt, die Menge $PS(\mathcal{L})^c$ berechnet. Dann wird für jedes Element aus $PS(\mathcal{L})^c$, also insbesondere diejenigen aus Abbildung 75, geprüft, ob es in dem betrachteten Netz aktiviert ist. Dies ist für kein Element der Fall. Somit resultiert eine positive Ausgabe des Tests.*

Die Komplexität des pessimistischen Übereinstimmungstests hängt insbesondere von der Größe der Menge $PS(\mathcal{L})^c$ ab. Zuerst einmal muss diese Menge entsprechend ihrer Definition konstruiert werden. Dies ist aufwändig, da jedes Präfix jeder BPO aus $PS(\mathcal{L})$ betrachtet werden muss, d.h. schon die Konstruktion von $PS(\mathcal{L})^c$ erfordert exponentiellen Zeitaufwand in der Größe der Eingabe $\mathcal{L}$. Auch die Anzahl an BPOs in $PS(\mathcal{L})^c$ ist im schlechtesten Fall exponentiell in der Größe der Eingabe $\mathcal{L}$. Das Problem dabei ist, dass für jede BPO aus $PS(\mathcal{L})^c$ getestet werden muss, ob sie in $\mathfrak{Bpo}(N, m_0)$ enthalten ist oder nicht. Es ist also für jede BPO aus $PS(\mathcal{L})^c$ ein entsprechender Aktiviertheitstest erforderlich. Ein solcher kann entsprechend Definition 3.3.8 bzw. Lemma 3.3.1 mit exponentiellem Aufwand in der Größe der BPO durchgeführt werden. In [156] haben wir gezeigt, wie ein entsprechender Aktiviertheitstest auch mit polynomiellem Aufwand möglich ist. Die in [156] vorgestellten Algorithmen für einen polynomiellen Aktiviertheitstest verwenden das Konzept der Markenflüsse. Genauer gesagt wird die Frage, ob eine BPO eine Markenfluss-BPO ist oder nicht, in ein entsprechendes Flussoptimierungsproblem übersetzt. Dieses kann dann mithilfe entsprechender Flussoptimierungsalgorithmen in polynomieller Zeit gelöst werden [97].

Insgesamt können also die einzelnen Aktiviertheitstests des Algorithmus mit polynomiellem Aufwand durchgeführt werden, u.U. sind aber exponentiell viele solche Tests nötig.
In folgender Bemerkung werden einige Möglichkeiten, um den betrachteten Algorithmus möglichst effizient zu gestalten, diskutiert.

Bemerkung 4.5.2 (Optimierungsvorschläge)

BEMERKUNG 4.5.2 (OPTIMIERUNGSVORSCHLÄGE)
Die folgenden Optimierungen von Algorithmus 4.5.2 sind möglich.

- *Ein erster Einfluss auf die Effizienz des Algorithmus lässt sich durch die Wahl eines geschickten Verfahrens zur Berechnung von* $\mathrm{PS}(\mathcal{L})^c$ *nehmen.*
- *Dann ist insbesondere die Größe von* $\mathrm{PS}(\mathcal{L})^c$ *wesentlich für die Komplexität des Algorithmus. Es kann sich hier lohnen, alternative Definitionen von* $\mathrm{PS}(\mathcal{L})^c$ *in Betracht zu ziehen, welche auch die am Anfang des Unterabschnittes diskutierten Anforderung an eine Repräsentation des Komplements von* $\mathrm{PS}(\mathcal{L})$ *erfüllen. Wir vermuten, dass es hier möglich ist, kleinere Mengen* $\mathrm{PS}(\mathcal{L})^c$ *zur Repräsentation des Komplements von* $\mathrm{PS}(\mathcal{L})$ *sinnvoll zu verwenden. Beispielsweise kann im Falle von zwei BPOs, wobei die erste BPO ein Präfix einer Sequentialisierung der zweiten ist, die zweite BPO aus* $\mathrm{PS}(\mathcal{L})$ *weggelassen werden, da diese nur aktiviert sein kann, wenn dies auch für die erste gilt.*
- *Weiter ist die Effizienz des verwendeten Verfahrens für die notwendigen Aktiviertheitstests sehr wichtig. Hier sollten insbesondere die verschiedenen in [156] diskutierten polynomiellen Testverfahren in Betracht gezogen werden. Dort werden zwei Basisverfahren sowie etliche Verbesserung der Basisverfahren vorgestellt. In unserer speziellen Situation kann auch noch über zusätzliche Weiterentwicklungen der Verfahren nachgedacht werden. Beispielsweise werden in Algorithmus 4.5.2 einige sehr ähnliche BPOs getestet; in diesem Fall sollte es möglich sein, Zwischenergebnisse vorheriger Tests zur Effizienzverbesserung bei späteren Tests wiederzuverwenden.*
- *Die Aktiviertheitstests gehen natürlich um so schneller vonstatten, je kleiner die betrachtete Repräsentation des gesättigt zulässigen Netzes ist. Im Rahmen der Überlegungen des letzten Abschnittes kann ein Verfahren zur Berechnung einer Repräsentation einschließlich entsprechender Optimierungen gewählt werden, welches ein möglichst kleines Netz erzeugt.*
- *Da der Algorithmus abbricht, sobald einer der Tests in Zeile 6 positiv ist, spielt die Reihenfolge der Abarbeitung der BPOs aus* $\mathrm{PS}(\mathcal{L})^c$ *in der Schleife eine Rolle. Hier kann wiederum mit Heuristiken versucht werden, eine Reihenfolge zu finden, welche früh zu einem Abbruch führt. Dabei kann es sinnvoll sein, nicht sofort die ganze Menge* $\mathrm{PS}(\mathcal{L})^c$ *zu berechnen, sondern sukzessive BPOs aus* $\mathrm{PS}(\mathcal{L})^c$ *zu berechnen und dann unmittelbar zu testen. Im Falle eines Abbruchs kann so die Berechnung der ganzen Menge* $\mathrm{PS}(\mathcal{L})^c$ *vermieden werden.*
- *In der wichtigen Situation, dass eine Schrittseparations-Repräsentation bzw. BPO-Separations-Repräsentation* $(N, \mathfrak{m}_0)$ *betrachtet wird, welche alle falschen Fortsetzungen verhindert (ansonsten ist eigentlich kein Übereinstimmungstest notwendig), ist nach Lemma 4.4.1 bzw. Lemma 4.4.5 sichergestellt, dass genau dann* $\mathfrak{Bpo}(N, \mathfrak{m}_0) = \mathrm{PS}(\mathcal{L})$ *gilt, wenn* $\mathrm{PS}(\mathcal{L})$ *schrittabgeschlossen bzw. fortsetzungsabgeschlossen ist.*

In diesem Fall stellt also ein Algorithmus, welcher die entsprechende Eigenschaft von $\mathrm{PS}(\mathcal{L})$ *prüft, einen geeigneten Übereinstimmungstest dar. Eine Prüfung der Eigenschaft kann sehr ähnlich zum pessimistischen Übereinstimmungstest durchgeführt werden. Es werden auch alle BPOs* $\mathrm{bpo} \in \mathrm{PS}(\mathcal{L})^c$ *betrachtet. Anstelle des Tests, ob* bpo *in* (N, m_0) *aktiviert ist, tritt nun aber ein Test, ob* $\mathrm{Slin}(\mathrm{bpo}) \subseteq \mathrm{Slin}(\mathrm{Pref}(\mathcal{L}))$ *bzw. ob* $\{\mathrm{bpo}\}^{\Pi}_{co} \subseteq \mathcal{L}^{\Pi}_{co}$ *gilt. Analog zum pessimistischen Übereinstimmungstest wird im Fall eines positiven Tests direkt* `false` *zurückgegeben. Auch hier wird die Ausgabe* `true` *erzeugt, wenn alle Tests negativ verlaufen sind. Die Korrektheit dieses Übereinstimmungstests lässt sich mit einer zum Beweis von Lemma 4.5.3 ähnlichen Argumentation herleiten. Es ist zu beachten, dass dieser Test vollkommen unabhängig vom synthetisierten Netz* (N, m_0) *abläuft. Es geht in der betrachteten Ausgangssituation nur noch darum, ob die partielle Sprache* $\mathrm{PS}(\mathcal{L})$ *schrittabgeschlossen bzw. fortsetzungsabgeschlossen ist. Die hierfür notwendigen Teilmengen-Tests, welche die Aktiviertheitstests des pessimistischen Übereinstimmungstests ersetzen, können auf Grund der Größe der zu betrachtenden Mengen ebenfalls aufwändig sein.*

Beispiel

BEISPIEL: *Im Hinblick auf den zweiten Optimierungsvorschlag lässt sich schon bei den in Abbildung 75 illustrierten BPOs sehen, dass häufig Präfixe und Sequentialisierungen in* $\mathrm{PS}(\mathcal{L})^c$ *vorkommen. So ist die sechste dargestellte BPO eine Sequentialisierung der vierten und diese ist wiederum eine Sequentialisierung der zweiten. Auch die fünfte BPO ist eine Sequentialisierung der dritten BPO. Somit könnten die zweite, dritte und vierte BPO weggelassen werden.*
Nun betrachten wir den dritten Optimierungsvorschlag. Bei einem Aktiviertheitstest entsprechend Lemma 3.3.1 müssen beispielsweise für die BPO aus Abbildung 62 2^n *Schnitte überprüft werden. Wird eine BPO* $\mathrm{bpo} = (V, <, l)$ *entsprechend [156] in ein Flussnetzwerk übersetzt und dann die Aktiviertheit mit einem Flussoptimierungsverfahren getestet, so lässt sich eine polynomielle Komplexität erzielen. Bei Verwendung des klassischen Ford-Fulkerson-Algorithmus [97] ergibt sich beispielsweise eine Laufzeit von* $O(|V\| < |^2)$. *Mit fortschrittlicheren Flussoptimierungsverfahren sind sogar noch geringere Komplexitäten erzielbar [156].*

Im schlechtesten Fall hat der pessimistische Übereinstimmungstest auch bei Verwendung geeigneter Optimierungen aufgrund der exponentiellen Größe von $\mathrm{PS}(\mathcal{L})^c$ eine exponentielle Laufzeit. Mit einem effizienten Einsatz von Optimierungsverfahren vor allem zur Reduktion der Menge $\mathrm{PS}(\mathcal{L})^c$ scheint aber in vielen Fällen eine gutartige Komplexität des pessimistischen Übereinstimmungstests erzielbar zu sein.

4.6 IMPLEMENTIERUNG

In diesem Kapitel stellen wir das Werkzeug VipTool vor, welches die in den letzten Abschnitten entwickelten Verfahren zur Synthese eines Stellen/Transitions-Netzes aus einer endlichen partiellen Sprache unterstützt.
Obwohl es generell viele mögliche Anwendungsfelder für Petrinetzsynthese gibt, existiert, wie schon in der Einleitung diskutiert, bisher nur wenig Werkzeugunterstützung. In dem Werkzeug Petrify [58] ist ein Verfahren zur Synthese von beschrifteten elementaren Netzen aus Transitionssystemen implementiert. Petrify ist speziell für den Bereich des Hardwareentwurfs maßgeschneidert. Es können asynchro-

ne Kontroll-Schaltungen aus Signal-Transitions-Graphen synthetisiert werden, wobei sowohl die asynchrone Kontroll-Schaltung als auch der Signal-Transitions-Graph als Petrinetz gegeben sind. Das Process-Mining-Werkzeug ProM [2] verwendet Syntheseverfahren, um automatisch Prozessmodelle in der Form von Petrinetzen aus Ereignis-Logs zu generieren. Die Ereignis-Logs spezifizieren dabei das Verhalten eines gesuchten Prozessmodells. Allerdings enthält ProM keine reinen Syntheseverfahren für Petrinetze. Synet [49] ist ein Software-Paket zur Synthese von beschränkten S/T-Netzen aus Transitionssystemen. Es unterstützt insbesondere Verfahren zur Erzeugung sog. verteilter Petrinetze. Diese bestehen aus verteilten Komponenten, die über asynchronen Nachrichtenaustausch kommunizieren. Synet wurde im Bereich des Designs von Kommunikationsprotokollen angewendet. Jedoch weisen die Autoren von Synet ausdrücklich darauf hin, dass Synet als vorläufiger Prototyp, keinesfalls aber als stabiles und robustes Werkzeug, betrachtet werden sollte. Ein aktuell neu entwickeltes Werkzeug zur Synthese von Petrinetzen heißt Genet [52]. Es wurde von der selben Arbeitsgruppe, welche auch Petrify entwickelt hat, implementiert. In Genet sind einige Erweiterungen gegenüber den in Petrify verwendeten Verfahren integriert. Insbesondere werden S/T-Netze und nicht mehr elementare Netze aus Transitionssystemen synthetisiert. Unter Anwendungsaspekten ist Petrify wie ProM auf das Gebiet des Process-Mining fokussiert. Es enthält hierfür spezielle Anpassungen der implementierten Syntheseverfahren. Unser Werkzeug VipTool unterstützt nun als einzige Software die Synthese von Stellen/Transitions-Netzen aus partiellen Sprachen.

4.6.1 *Hintergrund*

VipTool wurde ursprünglich an der Universität Karlsruhe im Rahmen des Forschungsprojekts VIP (**V**erifikation von **I**nformationssystemen durch die Evaluierung **p**artiell geordneter Abläufe) als ein Werkzeug zur Modellierung, Simulation, Validierung und Verifikation von Geschäftsprozessen mithilfe von halbgeordneten Abläufen von Petrinetzen entwickelt [99]. Die Hauptfunktionalität der ursprünglichen Version von VipTool ist die Berechnung der einzelnen maximalen Prozessnetze und einer gesamten Entfaltung eines gegebenen Petrinetzes zu Visualisierungs- und Analysezwecken [99]. Dabei soll eine mithilfe eines Tiefensuch-Verfahrens durchgeführte „on the fly"-Berechnung garantieren, dass auch bei sehr komplexen Netzen zumindest noch einige Prozessnetze erzeugt werden können. Durch den Fokus auf einzelne Prozessnetze unterscheidet sich VipTool von verwandten Werkzeugen, welche nur ganze Netzentfaltungen z.B. im Rahmen des Model-Checking berechnen so wie PUNF (`homepages.cs.ncl.ac.uk/victor.khomenko/tools/tools.html`).
Das ursprüngliche VipTool ist in der hauptsächlich als Skriptsprache verwendeten Programmiersprache Python geschrieben. Später wurde eine neue Version von VipTool in Java implementiert [76]. Diese Implementierung war nun modular aufgebaut und folgte den wesentlichen Prinzipien der Objektorientierung. Wichtig ist, dass etliche Fehler und Probleme der ursprünglichen Version behoben werden konnten. Den bedeutsamsten Fortschritt der überarbeiteten Version von VipTool stellten allerdings die neuen Funktionalitäten dar [76]. Diese beinhalten u.a. einen komfortableren Editor, Exportfunktionalitäten, und eine verbes-

serte „on the fly"-Berechnung von Prozessnetzen. Die Hauptverbesserung lag in der Unterstützung eines neuen, in [71, 76] vorgestellten, schrittweisen Validierungs- und Designkonzepts für Systeme in der Form von Petrinetzen durch die Betrachtung von Prozessnetzen. Erst soll das Petrinetz validiert werden, indem die Prozessnetze untersucht werden. Dann können bestimmte Arten von Spezifikationen in das Petrinetzmodell integriert werden. Zu Validierungszwecken wird anschließend zwischen denjenigen Prozessnetzen unterschieden, welche die Spezifikation erfüllen, und denjenigen, welche sie nicht erfüllen. Daraufhin soll dann zuerst geprüft werden, ob die Spezifikation tatsächlich die gewünschten Eigenschaften modelliert. Falls notwendig wird die Spezifikation dementsprechend angepasst. Danach soll das Petrinetzmodell derart verändert werden, dass es die endgültige Spezifikation erfüllt.
Diese VipTool-Version wurde sukzessive durch weitere kleinere Funktionalitäten erweitert. Eine wesentliche Neuerung wurde schließlich in [24] vorgestellt. VipTool wurde um Verfahren zum Testen, ob eine gegebene BPO in einem Petrinetz aktiviert ist, ergänzt. Auch einige weiterführende Fragestellungen wie die Frage nach einem zu einer aktivierten BPO gehörigen Prozessnetz, die Frage, ob eine aktivierte BPO sogar minimal aktiviert ist, und die Frage nach Gründen dafür, dass eine BPO nicht aktiviert ist, werden durch die implementierten Verfahren beantwortet. Den Verfahren liegen die polynomiellen Algorithmen zum Testen von Aktiviertheit aus [133, 156] zu Grunde. Somit erlauben sie eine effiziente Verifikation von Petrinetzmodellen. Insbesondere ergänzen sie aber auch die bisherigen Validierungsfunktionalitäten von VipTool sehr gut. In der Situation, dass es aus Komplexitätsgründen nicht möglich ist, alle Prozessnetze eines Petrinetzes zu berechnen, ist es in obigem Validierungskontext sinnvoll, direkt spezielle Abläufe auf Ausführbarkeit zu testen. Außerdem erlauben die neuen Verfahren genaue Analysen, warum bestimmte Abläufe in einem Petrinetzmodell nicht ausführbar sind. Diese Informationen können im Kontext des vorgeschlagenen Validierungsansatzes wichtige Einblicke bringen.
Im Rahmen dieser Arbeit wurden nun noch Verfahren zur Synthese von S/T-Netzen aus partiellen Sprachen in VipTool integriert. Wir haben die verschiedenen in diesem Kapitel vorgestellten Syntheseverfahren implementiert. Somit lässt sich nun sagen, dass VipTool alle im Kontext von halbgeordneten Abläufen von Petrinetzen denkbaren Funktionalitäten abdeckt. Es können ausgehend von einem Netz alle Abläufe durch entsprechende Entfaltungsfunktionalitäten berechnet werden. Es kann weiter ausgehend von einem Netz und einzelnen Abläufen mit entsprechenden Testfunktionalitäten die Ausführbarkeit der Abläufe geprüft werden. Schließlich lässt sich mit den neuen Synthesefunktionalitäten nun auch ausschließlich ausgehend von einer Menge von Abläufen ein Netz mit dem entsprechenden Ablaufverhalten synthetisieren. In diesem Sinne runden die Synthesefunktionalitäten von VipTool die bisherigen Funktionalitäten aus theoretischer Sicht sehr gut ab. Alle Hauptfunktionalitäten von VipTool sind noch einmal in Abbildung 76 illustriert.
Auch aus Anwendungssicht ergänzen sich die verschiedenen Funktionalitäten von VipTool sehr gut. Wir haben schon dargestellt, inwiefern das Zusammenspiel der Entfaltungs- und der Testfunktionalitäten zur Validierung und Verbesserung von Petrinetzmodellen äußerst fruchtbar ist. Allerdings fehlt bei dem vorgestellten Validierungs- und Designansatz

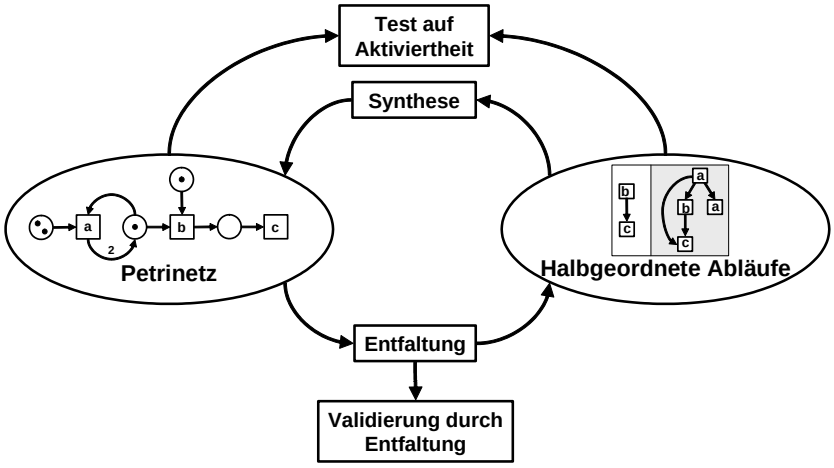

Abbildung 76: Funktionalitäten von VipTool.

der Schritt der Generierung eines Ausgangs-Petrinetzes. Der klassische Ansatz, um ein erstes Prozessmodell zu konstruieren, beginnt u.a. mit der Identifizierung von Aktivitäten, Ereignissen und Ressourcen des Prozesses. Aus derartigen Informationen wird dann direkt der Kontrollfluss des Prozesses modelliert. Die Validität des Modells wird dann im Nachhinein geprüft, indem das Verhalten des Prozessmodells mit dem Verhalten des tatsächlichen Prozesses verglichen wird. Mit den neuen Syntheseverfahren können wir nun einen anderen Weg gehen. Das Verhalten des tatsächlichen Prozesses stellt nun den Ausgangspunkt dar. Abläufe des Prozesses werden gesammelt und formalisiert. Dann wird daraus automatisch ein Prozessmodell mittels Synthesealgorithmen erzeugt. Der Vorteil ist, dass die Hauptaufgabe eines Modellierers bei diesem Vorgehen das Identifizieren und Modellieren der Abläufe des betrachteten Prozesses ist. Eine Modellierung auf dieser Ebene einzelner Prozessinstanzen ist typischerweise einfacher als das Modellieren des kompletten Prozesses. Außerdem können die Abläufe verteilt von Fachexperten modelliert werden. Ein auf diese Weise gewonnenes Ausgangsmodell kann nun mit den weiteren Funktionalitäten von VipTool validiert und weiterentwickelt werden. Auch hier können bei der Anpassung des Petrinetzes an eine veränderte Spezifikation Syntheseverfahren noch einmal eine Rolle spielen.
Im Rahmen der Implementierung der Syntheseverfahren dieser Arbeit wurde VipTool noch einmal restrukturiert. Auf Grund der inzwischen sehr hohen Anzahl an Funktionalitäten haben wir die Architektur von VipTool neu gestaltet. VipTool hat nun eine sehr flexible offene Plug-In-Architektur, welche wir im nächsten Unterabschnitt kurz beschreiben werden.
Seit der Implementierung der Algorithmen aus diesem Kapitel sind noch zwei weitere wesentliche neue Funktionalitäten zu VipTool hinzugekommen. Zum einen wurden die Entfaltungsverfahren um sehr effiziente Verfahren zur Berechnung von Markenflussprozessen erweitert (vgl. Bemerkung 4.3.3). Diese werden inzwischen, wie in Unterabschnitt 4.5.1 dargestellt, auch im Rahmen des optimistischen Übereinstimmungstests eingesetzt. Zum anderen haben wir Verfahren zur Synthese von S/T-Netzen aus termbasierten Repräsentationen unendlicher par-

tieller Sprachen in VipTool implementiert. Auf diese Verfahren und deren Implementierung wird in Abschnitt 5.1 näher eingegangen.
Für weitere Hintergrundinformationen zu dem Werkzeug VipTool sei schließlich noch auf ein Wiki, welches wir für VipTool eingerichtet haben, verwiesen. Es kann unter der Adresse `viptool.ku-eichstaett.de` gefunden werden. Hier sind alle wichtigen und aktuellen Informationen zu VipTool gesammelt. Außerdem kann VipTool von dieser Seite frei heruntergeladen werden.
Insgesamt lässt sich zusammenfassen, dass VipTool umfassende Funktionalitäten, welche sich mit Fragestellungen im Zusammenhang mit halbgeordneten Abläufen von Petrinetzen befassen, unterstützt. Es lassen sich die wichtigen theoretischen Probleme im Rahmen der Modellierung von Abhängigkeiten und Nebenläufigkeiten von halbgeordneten Abläufen von Petrinetzen lösen, aber auch viele praktische Aspekte im Rahmen der Modellierung von nebenläufigen Prozessen sind berücksichtigt. Da halbgeordnete Abläufe häufig als die geeignetste und intuitivste Darstellung von Verhalten von Petrinetzen betrachtet werden und sich sehr gut zur formalen Modellierung von Szenarien eignen, kann man sagen, dass VipTool eine wichtige Lücke im Bereich von Petrinetzwerkzeugen füllt. Es gibt zwar auch viele andere Petrinetzwerkzeuge, welche sich mit Modellierung und Analyse beschäftigen. Dabei spielt aber Halbordnungsverhalten ausschließlich im Rahmen von auf Entfaltungen basierenden Model-Checking-Verfahren eine Rolle.

4.6.2 *Architektur*

In diesem Unterabschnitt diskutieren wir detailliert die neue Architektur von VipTool. Das aktuelle VipTool ist ein plattformunabhängiges in Java entwickeltes Software-Werkzeug. Seine Vorgängerversionen [99, 76, 24] waren allein stehende Anwendungen. Aufgrund der gestiegenen Anzahl an Funktionalitäten und um generell die Feature-Erweiterbarkeit zu verbessern, haben wir VipTool im Rahmen einer offenen Plug-In-Architektur neu gestaltet. VipTool verwendet nur Standard Java-Bibliotheken und eine mit dem Java Swing Widget Werzeugsatz entwickelte graphische Benutzeroberfläche. Die Implementierung folgt strikt fortgeschrittenen objektorientierten Paradigmen. Bei der Entwicklung der neuen VipTool-Version haben wir professionelle Softwareentwicklungs-Standards wie Design Patterns, Coding Konventionen, ein Bug-Tracking System und eine ausführliche Dokumentation verwendet. Im folgenden Absatz wird ein Gesamtüberblick über die neue Architektur von VipTool gegeben.
Die VipTool Kern-Plattform bietet eine flexible Plug-In-Technologie. Der sog. Erweiterungs-Manager innerhalb der Kern-Plattform dient als Grundlage für die Integration von Plug-Ins. Genauer gesagt stellt er Funktionen zur Verfügung, um zur Laufzeit dynamisch Plug-Ins zu laden. Darüber hinaus beinhaltet die Kern-Plattform Module zur Festlegung und Implementierung der grundlegenden GUI-Elemente, für Projektmanagement-Funktionalitäten, zur Organisation des Scheduling der verschiedenen Anwendungen und zur Festlegung systemweiter Konfigurationen. Die Entwicklung von Plug-Ins wird durch die VipTool SDK Bibliothek unterstützt. Das VipTool SDK stellt geeignete Schnittstellen zur Anbindung von Plug-Ins zur Verfügung. Dies teilen sich in Schnittstellen, welche von den Plug-Ins implementiert werden,

und Schnittstellen, welche die Verbindung zur VipTool Kern-Plattform arrangieren, auf. Des Weiteren beinhaltet das VipTool SDK einige zentrale Unterstützungsklassen. Neben der Kern-Plattform und dem SDK umfasst VipTool natürlich noch die verschiedenen funktionalen Komponenten. Diese sind in einer offenen Plug-In-Architektur organisiert. Plug-Ins können, wie erläutert, mithilfe des SDKs sehr einfach programmiert werden. Sie lassen sich über den Erweiterungs-Manager an die Kern-Plattform anbinden. Die Einbindung der Plug-Ins in die GUI der Kern-Plattform wird über XML-Spezifikationen bewerkstelligt. Dies bedeutet, dass XML-Dateien verwendet werden, um die GUI von VipTool durch geeignetes Hinzufügen und Erweitern von Menüs, Knöpfen und ähnlichen Komponenten, welche für die entsprechenden Plug-Ins benötigt werden, zu konfigurieren. Damit ergibt sich eine einfache Erweiterbarkeit und eine gute Skalierbarkeit von VipTool. Abbildung 77 zeigt die Architektur von VipTool, wobei die funktionalen Plug-In-Komponenten in Paketen mit homogenen Funktionalitäten zusammengefasst sind. Abhängigkeiten der funktionalen Komponenten sind durch Pfeile dargestellt.

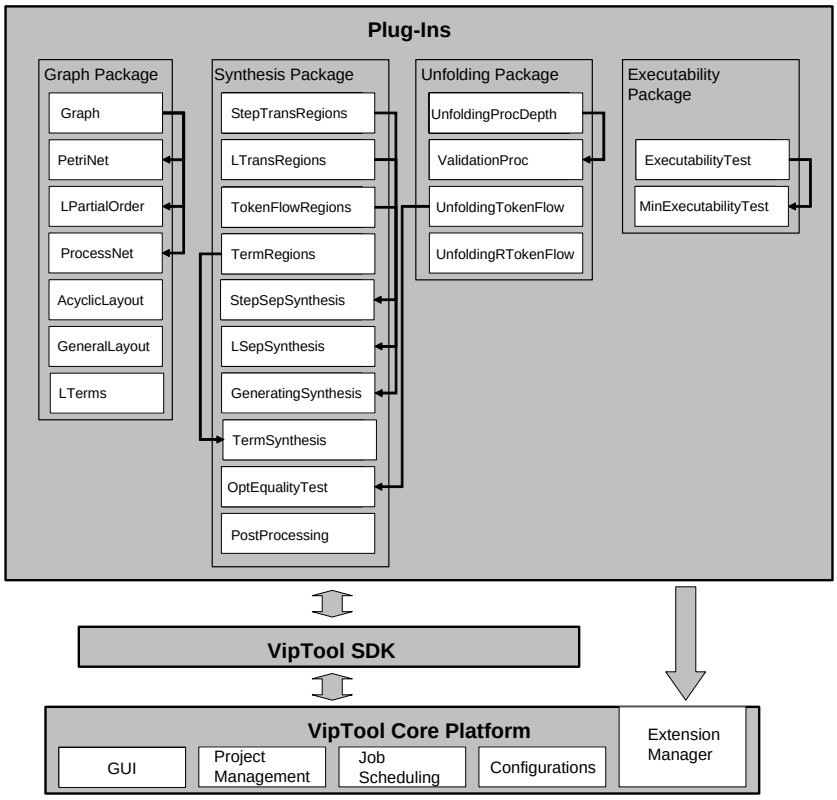

Abbildung 77: Architektur von VipTool.

Die im letzten Unterabschnitt beschriebenen Schlüsselfunktionalitäten früherer VipTool-Versionen wurden extrahiert und als unabhängige Plug-Ins in das neue VipTool reimplementiert. Vollständig neue Plug-Ins sind alle Synthese-Funktionalitäten, die neuen Entfaltungs-Funktionalitäten und eine Reihe von Plug-Ins, welche Klassen zum Darstellen und Bearbeiten von Graphen enthalten. Da die Anzahl der in VipTool verwendeten und unterstützten Graph- und Petrinetzklassen gewachsen ist, haben wir versucht, Graphen so allgemein wie möglich

in einem Graph-Plug-In eines Graph-Paketes zu definieren. Funktionen zum Editieren und zur Benutzerinteraktion für die verschiedenen zu unterstützenden Graphtypen haben wir als zusätzliche Plug-Ins des Graph-Paketes implementiert. Die Plug-Ins, welche die algorithmischen Funktionalitäten zur Verfügung stellen, haben wir in ein Synthese-Paket, ein Entfaltungs-Paket und ein Ausführbarkeits-Paket gegliedert. Algorithmische Plug-Ins können mit dem Scheduling-System der Kern-Plattform in verschiedenen Threads ausgeführt werden. Zur Kommunikation von Plug-Ins mit der Kern-Plattform sind beliebige Kommunikations-Protokolle zugelassen. Spezielle Schnittstellen des VipTool SDK unterstützen hierbei die Implementierung von nützlichen Standard-Kommunikations-Protokollen für algorithmische Plug-Ins. Diese umfassen beispielsweise Status-Meldungen, Fortschrittsanzeigen und Logging-Informationen. Zur Manipulation und zum Austausch von Daten können gemeinsame in vordefinierten Java-Klassen festgelegte Datenstrukturen und entsprechende Schnittstellen der Plug-Ins verwendet werden. Häufig wird hier aber der einfachere Weg eingeschlagen, ganze Dateien zu manipulieren und auszutauschen. Um die Integration von Komponenten zu vereinfachen, welche unabhängig oder kooperativ in verschiedenen Entwickler-Teams erstellt werden, unterstützt VipTool hier einen komfortablen Datenaustausch zwischen Plug-Ins und der Kern-Plattform über XML-Dokumente. Beispielsweise werden PNML-Dateien zur Speicherung und zum Austausch von Petrinetzen verwendet. Das Format PNML (Petri Net Mark-up Language) ist ein weit verbreitetes und anerkanntes Standard-Austauschformat für Petrinetze[220]. Alle relevanten XML-Dateien werden durch die Kern-Plattform in Workspaces und Projekten, welche beliebige Unterordner-Strukturen erlauben, verwaltet. Die Kern-Plattform bietet hierbei für den Benutzer ein Fenster mit einem Projektbaum, um ein einfaches Dateimanagement zu ermöglichen. Dieses ist insbesondere für einen flexiblen und einfachen Aufruf von funktionalen Plug-Ins wichtig, da die verschiedenen Funktionalitäten von VipTool heterogene und teilweise sehr komplexe Eingabe- und Ausgabetypen erfordern. Insgesamt unterstützt VipTool in umfassender Art und Weise einen einfachen XML-basierten Austausch von Daten zwischen den Plug-Ins. Die Verwendung einheitlicher Datenstrukturen in den Plug-Ins ist daher nicht nötig. Somit wird die Erweiterbarkeit, Skalierbarkeit, Komponierbarkeit und Wiederverwendbarkeit von VipTool-Funktionalitäten gefördert.

4.6.3 *Implementierungsdetails*

In diesem Unterabschnitt sollen die Implementierungen der einzelnen Plug-Ins von VipTool beschrieben werden (sieh Abbildung 77). Wir erläutern kurz die genaue Funktionalität und wichtige Besonderheiten jedes Plug-Ins. Das Synthese-Paket enthält insbesondere die in diesem Kapitel in vorgestellten Synthese-Algorithmen. Daher beschreiben wir die Plug-Ins des Synthese-Paketes etwas detaillierter, d.h. wir gehen darauf ein, inwieweit die in diesem Kapitel vorgestellten Pseudocode-Algorithmen auch eins-zu-eins in den Implementierungen umgesetzt sind, und welche diskutierten Optimierungen in den entsprechenden Plug-Ins verwendet werden.

Graph-Paket:

- *Graph:* Das Plug-In stellt grundlegende Graphklassen und entsprechende Schnittstellen zur Verfügung. Es ist die Grundlage für die Plug-Ins PetriNet, ProcessNet und LPartialOrder.
- *PetriNet:* Das Plug-In beinhaltet Visualisierungs- und Editier-Funktionen für Petrinetze. Außerdem werden einige interaktive Funktionalitäten für Petrinetze wie das Markenspiel oder das Anzeigen von Vor- und Nachbereichen unterstützt.
- *LPartialOrder:* Das Plug-In unterstützt die Visualisierung und das Design von halbgeordneten Abläufen durch beschriftete partielle Ordnungen sowie einige weitere verwandte Funktionen wie das Berechnen der transitiven Hülle einer binären Relation oder das Berechnen der Gerüstkantenmenge einer partiellen Ordnung.
- *ProcessNet:* Das Plug-In enthält Visualisierungs-Funktionalitäten für Prozessnetze.
- *AcyclicLayout:* Das Plug-In stellt Layout-Funktionalitäten zur automatischen Anordnung von gerichteten azyklischen Graphen wie beschrifteten partiellen Ordnungen und Prozessnetzen zur Verfügung. Diese basieren auf dem Sugiyama Algorithmus [138].
- *GeneralLayout:* Das Plug-In bietet Layout-Funktionalitäten zur Anordnung allgemeiner Graphen wie Petrinetze oder Transitionssysteme. Diese basieren auf dem Spring-Embedder Verfahren von Fruchterman und Reingold und verwenden einige Optimierungen dieses Verfahrens [138].
- *LTerms:* Das Plug-In beinhaltet Visualisierungs- und Editier-Funktionen für den Typ von Termen, welcher im Rahmen der termbasierten Repräsentation unendlicher partieller Sprachen in Abschnitt 5.1 vorkommt. Insbesondere lassen sich entsprechende Terme in einer struktogrammartigen Darstellung und in einer Darstellung als UML-Aktivitätsdiagramme visualisieren.

Entfaltungs-Paket:

- *UnfoldingProcDepth:* Das Plug-In entfaltet ein Petrinetz in seine maximalen Prozessnetze. Es folgt im wesentlichen den Standard Ideen zur Berechnung einer Entfaltung. Allerdings werden die Prozessnetze „on the fly“ mithilfe eines Tiefensuchverfahrens konstruiert. Das Plug-In erzeugt auch die gesamte Entfaltung des Petrinetzes. Hierbei wird ein Standard Cut-Off-Kriterium für Verzweigungsprozesse angewendet (siehe auch [76]).
- *ValidationProc:* Das Plug-In ermöglicht die graphische Spezifikation spezieller Eigenschaften eines Petrinetzes, indem das betrachtete Petrinetz direkt editiert wird. Solche Spezifikationen stellen typischerweise spezielle Formen von unerwünschtem oder erwünschtem Verhalten des Netzes dar. Die Menge der Prozessnetze, welche mit dem UnfoldingProcDepth-Plug-In berechnet werden, lässt sich dann automatisch in die Prozessnetze, welche die Spezifikationen erfüllen, und diejenigen, welche zumindest eine der Spezifikationen nicht erfüllen, unterteilen (siehe auch [76]).

- *UnfoldingTokenFlow:* Das Plug-In ist neu hinzugekommen. Es implementiert den Entfaltungsalgorithmus aus [34], welcher alle Markenfluss-Prozesse, welche zu einem Prozessnetz korrespondieren, und daraus dann alle Prozess-BPOs berechnet. Der Algorithmus verwendet ein innovatives auf dem Konzept der Markenflüsse basierendes Entfaltungsverfahren. Dieses ist im Falle von S/T-Netzen Standard-Entfaltungs-Verfahren bzgl. Zeit- und Speicherverbrauch signifikant überlegen [34]. Es wird auch die gesamte Markenfluss-Entfaltung eines Petrinetzes berechnet. Hierzu haben wir Cut-Off-Kriterien für klassische Verzweigungsprozesse auf Markenfluss-Prozesse übertragen [34, 76].
- *UnfoldingRTokenFlow:* Auch dieses Plug-In ist neu. Es ist sehr ähnlich zu dem UnfoldingTokenFlow-Plug-In. Allerdings werden hier entsprechend Überlegungen aus [34] direkt Markenfluss-BPOs, welche einer Prozess-BPO entsprechen, berechnet, ohne alle Markenfluss-Prozesse, welche zu Prozessnetzen korrespondieren, zu betrachten. Insbesondere wird keine vollständige Markenfluss-Entfaltung erzeugt. Dadurch lässt sich viel Speicherplatz sparen. In manchen Fällen ergibt sich dabei eine bessere Laufzeit, in anderen Fällen aber auch eine schlechtere [34]. Dies liegt daran, dass einerseits gewisse Markenfluss-Prozesse gar nicht berechnet und gespeichert werden, andererseits aber dadurch manchmal gewisse Zusatzberechnungen erforderlich sind. Insbesondere sind Aktiviertheitstests, welche mit den Plug-Ins des Ausführbarkeits-Paket durchgeführt werden, nötig. Auch dieses Plug-In verwendet Cut-Off-Kriterien. Es ist zu beachten, dass bei der konkreten Implementierung des Plug-Ins eine Variante des entsprechenden in [34] vorgestellten Verfahrens verwendet wird, welche nur Markenfluss-BPOs minimaler Ordnung erzeugt.

Ausführbarkeits-Paket:

- *ExecutabilityTest:* Das Plug-In unterstützt den ursprünglichen polynomiellen Test aus [133, 156], ob eine gegebene beschriftete partielle Ordnung in einem gegebenen S/T-Netz aktiviert ist. Der Test basiert auf dem Konzept der Markenflüsse und der Theorie der Flussnetzwerke. Er testet, ob die gegebene BPO eine Markenfluss-BPO ist. Dabei verwendet er den Algorithmus von Ford-Fulkerson [97]. Das Verfahren ermöglicht auch eine detaillierte Fehleranalyse im Falle, dass die betrachtete BPO nicht aktiviert ist. Im Falle, dass sie aktiviert ist, wird über die berechneten Markenflüsse ein zugehöriges Prozessnetz konstruiert (siehe auch [24]).
- *MinExecutabilityTest:* Dieses Plug-In basiert auf dem Executability-Test-Plug-In. Es berechnet, ob eine gegebene BPO in einem gegebenen S/T-Netz minimal aktiviert ist. Hierzu wird jeweils eine Gerüstkante der BPO weggelassen und dann das Ausführbarkeits-Test-Plug-In auf die derart modifizierte BPO angewandt, d.h. das AusführbarkeitsTest-Plug-In wird wiederholt verwendet (siehe auch [24]).

Synthese-Paket:

- *StepTransRegions:* Dieses Plug-In behandelt das Ungleichungssystem zur Charakterisierung der Menge der Schritt-Transitions-Regionen einer endlichen partiellen Sprache aus Lemma 4.3.6. Es

wird also die in diesem Lemma beschriebene Matrix konstruiert. Die Implementierung verwendet den zweiten Optimierungsvorschlag aus Bemerkung 4.3.1 sowie eine Heuristik, um entsprechend dem fünften Optimierungsvorschlag die zu betrachtende Menge von Schrittfolgen zu reduzieren. Die Heuristik vermeidet direkt bei der Konstruktion der Schrittfolgen die Erzeugung von Sequentialisierungen und Präfixen.

- *LTransRegions:* Dieses Plug-In ist mit der linear algebraischen Charakterisierung der Menge der BPO-Transitions-Regionen einer endlichen partiellen Sprache aus Lemma 4.3.8 befasst. Dementsprechend wird insbesondere die in diesem Lemma beschriebene Matrix konstruiert. Hierbei wird eine Heuristik verwendet, welche im Rahmen des zweiten Optimierungsvorschlages aus Bemerkung 4.3.2 die zu betrachtende Menge an Schnittfortsetzungen reduziert, d.h. es wird versucht, möglichst nur bzgl. $\ll$ maximale Schnittfortsetzungen zu betrachten.

- *TokenFlowRegions:* Dieses Plug-In erzeugt das Ungleichungssystem zur Charakterisierung der Menge der Markenfluss-Regionen einer endlichen partiellen Sprache aus Lemma 4.3.14. Es wird also die in diesem Lemma beschriebene Matrix konstruiert. Optimierungen, wie in Bemerkung 4.3.5 vorgeschlagen, werden nicht verwendet.

- *TermRegions:* Das Plug-In behandelt die linear algebraische Charakterisierung der Menge der Markenfluss-Regionen einer partiellen Sprache in der termbasierten Repräsentation unendlicher partieller Sprachen aus Abschnitt 5.1. Es wird also die in Lemma 5.1.5 beschriebene Matrix konstruiert. Hierbei kann das Plug-In TokenFlowRegions für die sog. Repräsentations-Menge des Terms verwendet werden. Es sei angemerkt, dass das Plug-In auch eine prototypische Implementierung einer linear algebraischen Charakterisierung von BPO-Transitions-Regionen entsprechender Terme beinhaltet. Dies ist in [33] näher ausgeführt.

- *StepSepSynthesis:* Das Plug-In erzeugt entsprechend Algorithmus 4.4.1 eine Schrittseparations-Repräsentation. Es umfasst somit den konstruktiven Teil der Synthese eines S/T-Netzes aus einer endlichen partiellen Sprache. Der Algorithmus verwendet eine der diskutierten Regionendefinitionen. Dementsprechend wird das entsprechende Plug-In StepTransRegions, LTransRegions oder TokenFlowRegions zur Konstruktion einer Matrix im Rahmen einer entsprechenden linear algebraischen Charakterisierung benutzt. Die in Algorithmus 4.4.1 zu betrachtenden Zulässigkeitsprobleme werden mit dem Simplex-Algorithmus gelöst. Wie in Abschnitt 3.4 erläutert, stellt der Simplex-Algorithmus das für solche Zwecke gebräuchlichste und in den meisten Fällen auch effizienteste Lösungsverfahren dar und es existieren sehr gute freie Implementierungen. Wir verwenden die Implementierung des Lpsolve 5.5.0.12 Java-Paketes [35], welches viele Einstellungsmöglichkeiten zur Verbesserung des Simplex-Algorithmus beinhaltet. Das StepSepSynthesis-Plug-In nutzt den ersten Optimierungsvorschlag aus Bemerkung 4.4.1, welcher in vielen Fällen direkt eine negative Antwort auf das Syntheseproblem ergibt und somit einen folgenden Übereinstimmungstest unnötig

macht. Allerdings wird der Algorithmus in diesem Fall nicht, wie in Bemerkung 4.4.1 vorgeschlagen, frühzeitig abgebrochen, sondern es wird die gesamte Schrittseparations-Repräsentation berechnet. Um entsprechend dem fünften Vorschlag aus Bemerkung 4.4.1 das Lösen der zu betrachtenden Zulässigkeitsprobleme mit dem Simplex-Algorithmus zu beschleunigen, verwenden wir einige interessante Einstellungen von Lpsolve wie bestimmte Preprocessing-Algorithmen. Wir haben hier basierend auf Erfahrungswerten Einstellungen gewählt, welche für unsere Probleme sehr effizient zu sein scheinen. Schließlich können nach der Berechnung des Netzes noch entsprechend dem achten Vorschlag in Bemerkung 4.4.1 implizite Stellen entfernt werden. Entsprechende Methoden haben wir in das PostProcessing-Plug-In ausgelagert.

- *LSepSynthesis:* Auch dieses Plug-In ist mit dem konstruktiven Teil der Synthese eines S/T-Netzes aus einer endlichen partiellen Sprache befasst. Es erzeugt entsprechend Algorithmus 4.4.2 eine BPO-Separations-Repräsentation. Der Algorithmus verwendet eine der diskutierten Regionendefinitionen, d.h. es wird wie im Falle des StepSepSynthesis-Plug-Ins das entsprechende Plug-In StepTransRegions, LTransRegions oder TokenFlowRegions zur Konstruktion einer Matrix im Rahmen einer entsprechenden linear algebraischen Charakterisierung benutzt. Die zu betrachtenden Zulässigkeitsprobleme werden auch hier mit dem Simplex-Algorithmus von Lpsolve gelöst. Das Plug-In verwendet analoge Optimierungen wie das StepSepSynthesis-Plug-In (vgl. hierzu Bemerkung 4.4.3).

- *GeneratingSynthese:* Schließlich löst auch dieses Plug-In den konstruktiven Teil der Synthese eines S/T-Netzes aus einer endlichen partiellen Sprache. Es berechnet entsprechend Algorithmus 4.4.3 eine Erzeugendensystem-Repräsentation. Auch dieses Plug-In verwendet eines der Plug-Ins StepTransRegions, LTransRegions oder TokenFlowRegions zur Konstruktion einer Matrix im Rahmen einer linear algebraischen Charakterisierung der betrachteten Regionendefinition. Die in dem Algorithmus vorkommende Berechnung eines ganzzahligen endlichen Erzeugendensystems eines rationalen spitzen polyedrischen Kegels wird mit einer fortschrittlichen Version der doppelten Beschreibungsmethode durchgeführt. Wie in Abschnitt 3.4 dargestellt, ist bei degenerierten Problemen, wie sie bei uns meist auftreten, die Verwendung eines Einfügealgorithmus sinnvoll. Alle Einfügealgorithmen sind der doppelten Beschreibungsmethode sehr ähnlich. Diese ist jedoch der verbreitetste und bekannteste Algorithmus und es existieren sehr gute freie Implementierungen. Wir verwenden die Implementierung des Polco-Paketes [212] (`www.csb.ethz.ch/tools/polco`), welches etliche Verbesserungen der doppelten Beschreibungsmethode entsprechend aktuellen Erkenntnissen nutzt. Somit wird entsprechend des ersten Optimierungsvorschlages in Bemerkung 4.4.4 ein sehr gutes Verfahren zur Lösung des Extremalstrahl-Aufzählungs-Problems verwendet. Darüber hinaus legen wir einem Nutzer gerade bei der Berechnung einer Erzeugendensystem-Repräsentation die Anwendung des PostProcessing-Plug-Ins zum Entfernen impliziter Stellen entsprechend des dritten Optimierungsvorschlages nahe.

- *TermSynthesis:* Im Rahmen dieses Plug-Ins haben wir das ErzeugendensystemSynthese-Plug-In derart erweitert, dass es auch die Synthese einer Erzeugendensystem-Repräsentation aus einer termbasierten Repräsentation einer unendlichen partiellen Sprache unterstützt. Wie in Abschnitt 5.1 beschrieben, ändert sich der Algorithmus für diesen Fall nicht. Es muss nur eine geeignete Matrix im Rahmen einer linear algebraischen Charakterisierung der Menge entsprechender Regionen des betrachteten Terms zugrunde gelegt werden. Hierzu wird das Plug-In TermRegions verwendet. Somit lässt sich auch der konstruktive Teil der Synthese eines S/T-Netzes aus einer entsprechenden unendlichen partiellen Sprache lösen. Es bleibt noch anzumerken, dass dieses Plug-In auch eine prototypische Implementierung der Berechnung einer BPO-Separations-Repräsentation für entsprechende Terme beinhaltet. Details hierzu finden sich in [33].

- *OptEqualityTest:* Das Plug-In führt einen Übereinstimmungstest durch, ob das Ablaufverhalten eines mit einem der Plug-Ins StepSepSynthesis, LSepSynthesis oder GeneratingSynthese berechneten Netzes mit der spezifizierten partiellen Sprache übereinstimmt. Somit wird entschieden, ob das Syntheseproblem eine positive Antwort hat oder nicht. In ersterem Fall löst das betrachtete Netz das Syntheseproblem. Das Plug-In ist eine Implementierung des optimistischen Übereinstimmungstests aus Algorithmus 4.5.1. Die erforderliche Berechnung aller maximaler Prozess-BPOs des betrachteten Netzes wird mithilfe des Plug-Ins UnfoldingTokenFlow durchgeführt. Das Plug-In vermeidet das Hinzufügen der Hilfsstellentripel st^t, wie im zweiten Punkt von Bemerkung 4.5.1 beschrieben. Weiter wird für die im Rahmen der Fallunterscheidung des Algorithmus notwendigen Isomorphietests die im fünften Punkt von Bemerkung 4.5.1 als Beispiel vorgeschlagene „branch-and-bound"-Strategie eines Backtracking-Algorithmus angewandt.

- *PesEqualityTest:* Das Plug-In stellt eine alternative Möglichkeit zum Plug-In OptEqualityTest für einen Übereinstimmungstest dar. Das Plug-In ist eine Implementierung des pessimistischen Übereinstimmungstests aus Algorithmus 4.5.2. Leider ist es zum aktuellen Zeitpunkt noch nicht vollständig fertiggestellt. Es soll entsprechend des dritten Punktes in Bemerkung 4.5.1 das Plug-In ExecutabilityTest verwenden, um die notwendigen Aktiviertheitstests in polynomieller Zeit durchzuführen.

- *PostProcessing:* Das Plug-In enthält Verfahren, um die in Bemerkung 4.4.2 erläuterten effizient überprüfbaren hinreichenden Bedingungen für implizite Stellen zu prüfen. Auf diese Weise können mit diesem Plug-In implizite Stellen eines gegebenen Netzes entfernt werden. Das Plug-In soll insbesondere genutzt werden, um die Größe der mit den Plug-Ins StepSepSynthesis, LSepSynthesis oder GeneratingSynthese berechneten Netze zu reduzieren.

Durch die Einbindung der Plug-Ins des Graph-Paketes werden die entsprechenden Funktionalitäten in die Benutzeroberfläche von VipTool integriert. Die funktionalen Plug-Ins des Entfaltungs-, des Ausführbarkeits- und des Synthese-Paketes lassen sich, sobald sie in VipTool eingebunden sind, über Kontextmenüs ausführen. Hierzu ist ein Rechtsklick

im Projektbaum auf eine für ein Plug-In geeignete Eingabe notwendig. Die Synthese-Plug-Ins lassen sich nicht einzeln ausführen, sondern sind direkt zu entsprechenden funktionalen Einheiten zusammengefasst. Die Plug-Ins StepSepSynthesis, LSepSynthesis und GeneratingSynthese bzw. TermSynthesis sind jeweils mit einem der Plug-Ins StepTransRegions, LTransRegions und TokenFlowRegions bzw. TermRegions zusammengefügt.Teilweise ist auch noch eines der Plug-Ins OptEqualityTest oder PesEqualityTest angehängt. Da in Anwendungen häufig nur der konstruktive Teil von Syntheseverfahren von Interesse ist, haben wir allerdings meistens auf die Einbindung der Übereinstimmungstest-Plug-Ins verzichtet.
Abbildung 78 illustriert abschließend die Benutzeroberfläche von VipTool. Links lässt sich das Workspace-Konzept und das Dateimanagement von VipTool erkennen. Oben befinden sich eine Menü-Leiste und Funktionsknöpfe. Unten werden Logging-Informationen sowie der Status aller aktiven Jobs angezeigt. Zentral finden sich, aufgeteilt in entsprechenden Fenstern, in VipTool editierte Repräsentationen der zwei BPOs aus Abbildung 4 sowie ein mit VipTool aus den zwei BPOs synthetisiertes S/T-Netz.

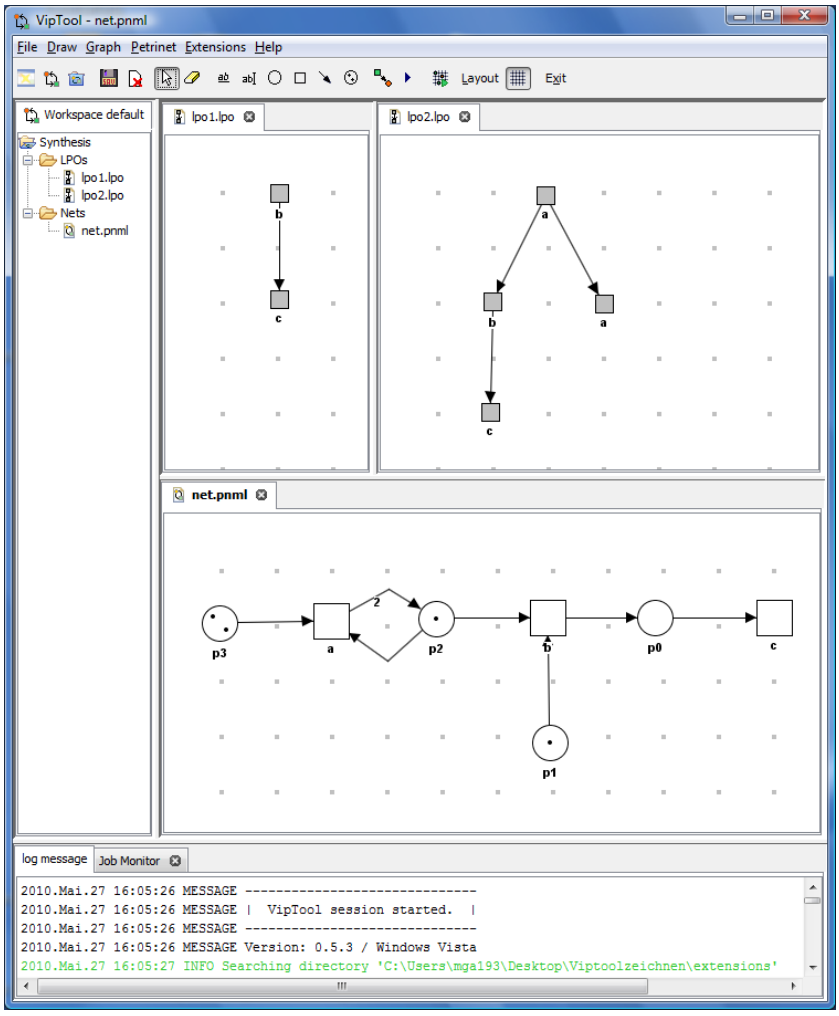

Abbildung 78: Benutzeroberfläche von VipTool.

4.7 VERGLEICH DER SYNTHESEVERFAHREN

In diesem Abschnitt sollen die verschiedenen in diesem Kapitel entwickelten Syntheseverfahren verglichen werden. Wir betrachten also die verschiedenen möglichen Instanziierungen von Algorithmus 4.2.1. Bei diesem Algorithmus können drei mögliche Regionendefinitionen, nämlich Schritt-Transitions-Regionen, BPO-Transitions-Regionen und Markenfluss-Regionen, drei mögliche Berechnungsverfahren, nämlich die Berechnung einer Schrittseparations-Repräsentation, einer BPO-Separations-Repräsentation und einer Erzeugendensystem-Repräsentation, und zwei mögliche Übereinstimmungstests, nämlich der optimistische und der pessimistische Übereinstimmungstest, gewählt werden. Wir vergleichen die verschiedenen Möglichkeiten durch theoretische Komplexitätsüberlegungen und durch experimentelle Tests.
Kurze Komplexitätsbetrachtungen haben wir auch schon jeweils in den entsprechenden Abschnitten dieses Kapitels durchgeführt. Dabei haben wir insbesondere gezeigt, dass BPO-Transitions-Regionen eine Weiterentwicklung der Schritt-Transitions-Regionen darstellen und die Betrachtung einer BPO-Separations-Repräsentation eine Weiterentwicklung der Betrachtung einer Schrittseparations-Repräsentation ist. Die Komplexität der Weiterentwicklung ist in beiden Fällen jeweils sehr ähnlich zu der ursprünglichen Vorgehensweise. Wir haben auch gezeigt, dass die Weiterentwicklungen im Allgemeinen eine leicht besser Komplexität aufweisen (aufgrund von Implementierungsdetails kann in Einzelfällen aber auch die ursprüngliche Vorgehensweise etwas effizienter sein). Daher gehen wir in den Überlegungen dieses Abschnittes nach der Diskussion von BPO-Transitions-Regionen bzw. BPO-Separations-Repräsentationen nur jeweils noch kurz auf Schritt-Transitions-Regionen bzw. Schrittseparations-Repräsentationen ein.

4.7.1 *Komplexitätsvergleich*

Zuerst vergleichen wir die verschiedenen Syntheseverfahren auf einer theoretischen Ebene. Für die verschiedenen Möglichkeiten der Synthese haben wir in diesem Kapitel verschiedenste Optimierungen diskutiert. Diese sind natürlich auch für Komplexitätsbetrachtungen von Bedeutung. Wir berücksichtigen bei den Betrachtungen in diesem Unterabschnitt die Optimierungen, welche wir auch bei den Implementierungen der Verfahren verwendet haben. Diese sind in Abschnitt 4.6 dargestellt.
Wir beginnen den Komplexitätsvergleich mit dem Vergleich der verschiedenen Regionendefinitionen. Für Komplexitätsbetrachtungen ist hier die Größe der jeweiligen Ungleichungssysteme einer entsprechenden linear algebraischen Charakterisierung entscheidend. Die Effizienz der Berechnungsverfahren zur Konstruktion eines Netzes basierend auf einer der Regionendefinitionen ist in allen Fällen wesentlich von der Größe der entsprechenden Ungleichungssysteme abhängig. Daneben spielt auch die Komplexität der Erstellung des Ungleichungssystems eine Rolle. Den zweiten Punkt des Komplexitätsvergleiches stellt dann eine genaue Diskussion der verschiedenen Berechnungsverfahren zur Synthese eines Netzes dar. Diese Berechnungsverfahren analysieren wir in Abhängigkeit von der schon zuvor diskutierten Größe der zu betrachtenden Ungleichungssysteme. Hierbei spielt neben der Laufzeit auch die Größe der erzeugten Netze eine Rolle. Zum einen ist die Konstruk-

tion kleiner Netze generell wünschenswert und zum anderen spielt die Größe der Netze für die abschließende Betrachtung der Komplexität der verschiedenen Übereinstimmungsprüfungen eine wesentliche Rolle. Wir messen die Größe eines Netzes in diesem Kapitel der Einfachheit halber durch die Anzahl der Stellen des Netzes (die Anzahl der Transitionen ist vorgegeben). Der letzte Teil des Komplexitätsvergleiches besteht schließlich aus der Betrachtung der Übereinstimmungstests in Abhängigkeit von der Größe des zu untersuchenden Netzes.
Die Komplexität eines gesamten Syntheseverfahrens wird in Abhängigkeit von der Größe der Eingabe bemessen. Die Eingabe für die Syntheseverfahren ist jeweils eine beliebige endliche partielle Sprache $\mathcal{L}$, d.h. eine endliche Menge von BPOs. Eine BPO $bpo = (V, <, l)$ wird typischerweise durch die Menge der Knoten V zusammen mit einer Beschriftung $l(v)$ für jeden Knoten $v \in V$ und der Menge der Gerüstkanten $<_g$ spezifiziert. Entsprechend ergibt sich die Größe der Eingabe als $\sum_{(V,<,l)\in\mathcal{L}}(2|V| + |<_g|)$. Häufig spielt die Anzahl der Beschriftungen $|T| \leqslant |V|$ eine Rolle. Diese kann meist als Konstante betrachtet werden. Dies liegt daran, dass die Anzahl der Beschriftungen nicht direkt von der Anzahl und der Größe der BPOs abhängig ist. In vielen realistischen Szenarien können die Transitionen des Netzes, welche ja den Beschriftungen entsprechen, von vornherein als fix angenommen werden.

Regionendefinitionen

Die Größe der Ungleichungssysteme der linear algebraischen Charakterisierungen der Menge der Schritt- und BPO-Transitions-Regionen und der Menge der Markenfluss-Regionen ergibt sich durch die Größe der Matrizen $\mathbf{A}_{\mathcal{L}}^{BPOT}$ bzw $\mathbf{A}_{\mathcal{L}}^{ST}$ und $\mathbf{A}_{\mathcal{L}}^{MAR}$. Die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ entspricht eigentlich der Anzahl der Schnittfortsetzungen von $\mathcal{L}$. Wir haben aber gezeigt, dass es genügt, nur die Menge der bzgl. $\ll$ maximalen Schnittfortsetzungen zu betrachten. Da jeder Schnitt eine Schnittfortsetzung definiert, entspricht die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ im schlechtesten Fall der Anzahl der Schnitte von $\mathcal{L}$. Allerdings können häufig etliche Schnitte vernachlässigt werden, da sie dieselbe oder eine bzgl. $\ll$ kleinere Schnittfortsetzung als ein anderer Schnitt definieren. Daher kann die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ wesentlich kleiner als die Anzahl der Schnitte von $\mathcal{L}$ sein. Die Anzahl der Schnitte von $\mathcal{L}$ ergibt sich als Summe der Anzahlen der Schnitte jeder in $\mathcal{L}$ enthaltenen BPO. Die Anzahl der Schnitte einer BPO $bpo = (V, <, l)$ kann im schlechtesten Fall exponentiell in $|V|$ und $|<_g|$ sein. Falls aber eine BPO wenig nebenläufige Ereignisse enthält, dann ist die Anzahl der Schnitte wesentlich geringer, beispielsweise entspricht sie im Falle einer totalen Ordnung der Anzahl der Knoten $|V|$. In dem entgegengesetzten Fall, dass eine BPO sehr wenig geordnete Knoten enthält, ist die Anzahl der Schnitte auch klein. In diesem Zusammenhang lässt sich leicht einsehen, dass die Anzahl der Schnitte durch $2^{|<_g|}$ beschränkt ist, d.h. ist $|<_g|$ im Extremfall 0 bzw. 1, so ist die Anzahl der Schnitte 1 bzw. 2. Dies liegt daran, dass Schnitte maximale Mengen unabhängiger Knoten sind und sich erst durch Ordnungsbeziehungen zwischen den Knoten mehrere Möglichkeiten für Schnitte ergeben. Hierbei kann sich pro Gerüstkante die Anzahl der Schnitte höchstens verdoppeln. Allerdings sind BPOs mit entsprechend wenig Kanten im Rahmen der Modellierung von Abläufen eher unrealistisch.

Die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ entspricht der Anzahl der Variablen des entsprechenden Ungleichungssystems. Diese Anzahl ist bei Transitions-Regionen durch $2|T|+1$ gegeben. Dieser Wert ist sehr klein und kann, wie erläutert, häufig sogar als Konstante betrachtet werden. Die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{ST}$ entspricht eigentlich der Anzahl der Schrittlinearisierungen von $\mathcal{L}$. Auch hier können ähnlich wie im Falle von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ häufig etliche Schrittlinearisierungen vernachlässigt werden. Von der Größenordnung her ergeben sich insgesamt analoge Komplexitätsüberlegungen für die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{ST}$ wie für die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$. Genauere Betrachtungen, wie in Abschnitt 4.3 durchgeführt, zeigen, dass die genaue Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{ST}$ im Allgemeinen größer ist als die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$. Die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{ST}$ entspricht der Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{BPOT}$, also $2|T|+1$.
Die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{MAR}$ stimmt generell ungefähr der Anzahl der Ereignisse in $\mathcal{L}$ überein. Für jede Beschriftung $t \in T$ enthält $\mathbf{A}_{\mathcal{L}}^{MAR}$ je eine Zeile für die (ZU)- und die (AB)-Restriktion des ersten und zweiten mit t beschrifteten Knotens in $\mathcal{L}$, des zweiten und dritten mit t beschrifteten Knotens in $\mathcal{L}$, usw. Außerdem enthält $\mathbf{A}_{\mathcal{L}}^{MAR}$ je eine Zeile für die (ANF)-Restriktion der ersten und zweiten BPO in $\mathcal{L}$, der zweiten und dritten BPO in $\mathcal{L}$, usw. Insgesamt ergibt sich also die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{MAR}$ als $2\sum_{t\in T}(|\bigcup_{(V,<,l)\in\mathcal{L}}\{v \in V \mid l(v) = t\}|-1)+|\mathcal{L}|-1 = 2\sum_{(V,<,l)\in\mathcal{L}}|V|-2|T|+|\mathcal{L}|-1$. Um so weniger Beschriftungen es gibt, um so größer ist diese Anzahl. Außerdem wächst die Anzahl mit der Anzahl der BPOs. Der schlechteste Fall ergibt sich also theoretisch in der (nicht sinnvollen) Situation, dass alle BPOs nur aus einem mit jeweils derselben Beschriftung versehenen Knoten bestehen. In jedem Fall ist die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{MAR}$ aber linear in der Anzahl $\sum_{(V,<,l)\in\mathcal{L}}|V|$ der Knoten von $\mathcal{L}$ (da $|\mathcal{L}| \leqslant \sum_{(V,<,l)\in\mathcal{L}}|V|$). Im schlechtesten Fall ergibt sich ein Vorfaktor von drei.
Die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{MAR}$ entspricht der Anzahl der Variablen des entsprechenden Ungleichungssystems. Diese entspricht der Anzahl $|\bigcup_{(V,<,l)\in\mathcal{L}} <^*|$ der Kanten aller $*$-Erweiterungen aller BPOs aus $\mathcal{L}$. Die Anzahl der Kanten $|<|$ einer BPO $bpo = (V,<,l)$ kann quadratisch in der Anzahl der Gerüstkanten $|<_g|$ sein. Im schlechtesten Fall ist bpo eine totale Ordnung. Dann gilt $|<_g| = |V|-1$, aber $|<| = ((|V|-1)|V|)/2$. Generell ergibt sich hier in BPOs mit vielen Ordnungsbeziehungen eine eher hohe Komplexität. Im Falle, dass $|<_g|$ klein ist ergibt sich häufig $|<| \approx |<_g|$. Im Detail hängt es aber natürlich von der genauen Verteilung der $<_g$-Kanten ab, wieviele transitive Kanten zusätzlich existieren. Darüberhinaus müssen nicht nur die $<$-Kanten berücksichtigt werden. Da die $*$-Erweiterungen der BPOs aus $\mathcal{L}$ betrachtet werden, müssen zusätzlich die Kanten von dem Quellenknoten zu jedem Knoten in V sowie die Kanten von jedem Knoten in V zu dem Senkenknoten und die Kante vom Quellen- zum Senkenknoten berücksichtigt werden. Daher kommen in jedem Fall noch $2|V|+1$ Kanten hinzu. Insgesamt kann die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{MAR}$ quadratisch in der Größe der Eingabe sein. Dies ist allerdings nur dann der Fall, wenn $\mathcal{L}$ wenig nebenläufiges Verhalten enthält, d.h. wenn die BPOs aus $\mathcal{L}$ stark geordnet sind. Falls ein gewisses Maß an Nebenläufigkeit in $\mathcal{L}$ existiert, ist die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{MAR}$ häufig linear in der Größe der Eingabe.
Zusammenfassend lässt sich sagen, dass $\mathbf{A}_{\mathcal{L}}^{BPOT}$ und $\mathbf{A}_{\mathcal{L}}^{ST}$ eine kleine meist als konstant anzunehmende Anzahl an Spalten haben, während

$\mathbf{A}_{\mathcal{L}}^{MAR}$ eine lineare und somit kleine Anzahl an Zeilen hat. Die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ und $\mathbf{A}_{\mathcal{L}}^{ST}$ kann exponentiell sein. Sie ist groß, wenn viel nebenläufiges Verhalten in $\mathcal{L}$ vorkommt. Hierbei muss beachtet werden, dass ab einem gewissen Grad an Nebenläufigkeit die Anzahl auch wieder kleiner wird. Darüber hinaus gibt es Fälle, in denen die Anzahl der Zeilen von $\mathbf{A}_{\mathcal{L}}^{BPOT}$ bzw. $\mathbf{A}_{\mathcal{L}}^{ST}$ im Verhältnis zur Anzahl der Schnitte in $\mathcal{L}$ bzw. zur Anzahl der Schrittlinearisierungen von $\mathcal{L}$ klein ist (wenn viele Schnittfortsetzungen bzw. Schrittlinearisierungen vernachlässigt werden können). In solchen Fällen kann sich auch bei viel Nebenläufigkeit eine kleine Anzahl an Zeilen ergeben. Generell hat $\mathbf{A}_{\mathcal{L}}^{ST}$ etwas mehr Zeilen als $\mathbf{A}_{\mathcal{L}}^{BPOT}$. Die Anzahl der Spalten von $\mathbf{A}_{\mathcal{L}}^{MAR}$ kann quadratisch sein. Dies ist allerdings nur für partielle Sprachen mit wenig Nebenläufigkeit der Fall.
Es lässt sich folgern, dass die Größen der Matrizen $\mathbf{A}_{\mathcal{L}}^{BPOT}$, $\mathbf{A}_{\mathcal{L}}^{ST}$ und $\mathbf{A}_{\mathcal{L}}^{MAR}$ stark von der Struktur der betrachteten partiellen Sprache $\mathcal{L}$ abhängen. Normalerweise sind $\mathbf{A}_{\mathcal{L}}^{BPOT}$ und $\mathbf{A}_{\mathcal{L}}^{ST}$ im Falle, dass $\mathcal{L}$ wenig nebenläufiges Verhalten enthält, kleiner und somit geeigneter als $\mathbf{A}_{\mathcal{L}}^{MAR}$. Falls $\mathcal{L}$ viel nebenläufiges Verhalten enthält, ist $\mathbf{A}_{\mathcal{L}}^{MAR}$ kleiner. Es kann hier aber auch Ausnahmen geben. Eine Ausnahme stellen beispielsweise Sprachen mit sehr viel Nebenläufigkeit dar. Dann weisen alle drei Matrizen eine geringe Größe auf. Generell gilt, dass die Matrix $\mathbf{A}_{\mathcal{L}}^{BPOT}$ im Allgemeinen ein wenig kleiner ist als die Matrix $\mathbf{A}_{\mathcal{L}}^{ST}$.

Berechnungsverfahren

Hier vergleichen wir die Algorithmen dieses Kapitels zur Berechnung einer Schrittseparations-Repräsentation, einer BPO-Separations-Repräsentation und einer Erzeugendensystem-Repräsentation in Abhängigkeit von der gewählten Regionendefinition. Alle drei Algorithmen beginnen mit der Konstruktion der Matrix zur linear algebraischen Charakterisierung der verwendeten Regionendefinition. Bei Schritt- und BPO-Transitions-Regionen ist der hierfür notwendige Aufwand ungefähr quadratisch in der Größe der entsprechenden Matrizen. Dies liegt daran, wie die relevanten Schrittlinearisierungen bzw. Schnittfortsetzungen berechnet werden. Insbesondere werden Vergleiche der Schrittlinearisierungen bzw. Schnittfortsetzungen durchgeführt, um bestimmte Schrittlinearisierungen bzw. Schnittfortsetzungen bei der Konstruktion der Matrix vernachlässigen zu können. Die bei Markenfluss-Regionen notwendige Aufteilung der Knoten nach Beschriftungen benötigt einen Aufwand von $|T| \sum_{(V,<,l)\in\mathcal{L}} |V|$, welcher typischerweise als linear in der Größe der Eingabe betrachtet werden kann. Die eigentliche Konstruktion der Matrix lässt sich dann in linearer Komplexität in Abhängigkeit von der Größe der Matrix durchführen. Die Größen der jeweiligen Matrizen wurden zuvor diskutiert. Insgesamt ist der Aufwand zur Konstruktion der Matrizen von der Größenordnung her im Vergleich zu den weiteren Schritten der Algorithmen zu vernachlässigen.
Wir diskutieren nun zuerst den weiteren Verlauf von Algorithmus 4.4.2 zur Berechnung einer BPO-Separations-Repräsentation. Der nächste Schritt ist die Berechnung der falschen BPO-Fortsetzungen. Um zuerst alle Präfixschritte zu berechnen, müssen alle Präfixe jeder BPO $bpo = (V, <, l) \in \mathcal{L}$ berechnet werden. Die Anzahl der Präfixe von bpo kann exponentiell in $|V|$ und $|<_g|$ sein. Im Detail gilt beispielsweise, dass eine Menge geordneter Knoten eine lineare Anzahl an Präfixen erzeugt, während eine Co-Menge von Knoten ein Präfix für jede Teil-

menge der Co-Menge erzeugt. Diese Überlegungen lassen sich induktiv auf beliebige BPOs fortsetzen. Insgesamt ist die Anzahl der Präfixe einer partiellen Sprache also groß, im schlechtesten Fall exponentiell, falls die Sprache viel nebenläufiges Verhalten enthält. Ausgehend von einem Präfixschritt π erfordert die Berechnung aller falscher BPO-Fortsetzungen (π, τ) im schlechtesten Fall exponentiellen Aufwand in der Größe des größten Schrittes τ', für welchen (π, τ') eine richtige Fortsetzung ist. Dies liegt daran, dass es nötig sein kann, alle Teilschritte von τ' jeweils um ein Ereignis zu erweitern und dann zu überprüfen, ob sich eine falsche Fortsetzung ergibt. Die Gesamtzahl der falschen BPO-Fortsetzungen hat unter der Annahme, dass die Anzahl der Beschriftungen $|T|$ konstant ist, eine obere Schranke, welche linear von der Anzahl der Präfixe von $\mathcal{L}$ abhängt.
Im nächsten Schritt des Algorithmus werden die Beschriftungen der partiellen Sprache betrachtet und als Transitionen des Netzes festgelegt. Dann wird jede falsche BPO-Fortsetzung $(\pi, \tau) \in \mathcal{L}_{ws}$ bearbeitet. Zuerst wird überprüft, ob (π, τ) in dem bis dato konstruierten Netz (N, m_0), $N = (P, T, W)$, aktiviert ist. Hierzu muss für jedes Stelle $p \in P$ überprüft werden, ob $m_0 - {}^\circ\pi + \pi^\circ \geqslant {}^\circ\tau$. Dabei ist (π, τ) durch die Knoten einer BPO aus $\mathcal{L}$ definiert, d.h. $|\pi| + |\tau| - 1$ ist durch die Knotenanzahl einer BPO beschränkt. Die Überprüfung der Ungleichung ist dementsprechend linear in der Anzahl der Knoten dieser BPO. Aber die Anzahl der nötigen Überprüfungen kann exponentiell in der Größe der Eingabe sein, da $|P|$ entsprechend groß sein kann. Normalerweise ist $|P|$ aber klein. Der Grund hierfür ist, dass eine Stelle für $(\pi, \tau) \in \mathcal{L}_{ws}$ nur dann hinzugefügt wird, wenn die Aktiviertheitsprüfung von (π, τ) positiv ist und es eine entsprechende Stelle gibt. Die Aktiviertheitsprüfung ist aber sehr häufig negativ, da eine Stelle in vielen Fällen nicht nur die betrachtete, sondern auch viele weitere falsche BPO-Fortsetzungen verhindert. Wenn diese später bearbeitet werden, ergibt sich ein negativer Aktiviertheitstest. Im schlechtesten Fall kann es aber trotzdem vorkommen, dass exponentiell viele Tests positiv sind und sich exponentiell viele Stellen ergeben.
Die Anzahl der positiven Aktiviertheitstests ist aber nun nicht nur für die Größe des Netzes bei den weiteren Aktiviertheitstests wichtig, sondern nur im positiven Fall finden die folgenden aufwändigen Schritte des Algorithmus statt. Im negativen Fall wird direkt die nächste falsche BPO-Fortsetzung bearbeitet. Der nächste Schritt im Falle eines positiven Tests ist das Lösen eines Zulässigkeitsproblems für ein Polyeder. Das Polyeder hängt von der verwendeten Regionendefinition ab. Er ist durch das Ungleichungssystem gegeben, welches von der zu Beginn des Algorithmus berechneten Matrix zusammen mit einer entsprechenden $\mathbf{b}$-Ungleichung definiert ist. Die Größe des Ungleichungssystems ergibt sich also durch die Größe dieser Matrix. Die Größen der hier in Frage kommenden Matrizen wurden, wie gesagt, schon zuvor diskutiert. Die Komplexität des Lösens der Zulässigkeitsprobleme ist natürlich stark von der Größe dieser Matrix abhängig. Wir haben verschiedene Lösungsverfahren diskutiert. Es gibt Verfahren, welche das Zulässigkeitsproblem in polynomieller Zeit in Abhängigkeit von der betrachteten Matrix lösen. Wir verwenden hier allerdings den Simplexalgorithmus, welcher im schlechtesten Fall exponentielle Laufzeit hat, in den meisten Fällen den polynomiellen Verfahren aber überlegen ist. Wie in Abschnitt 3.4 dargestellt, hat der Simplex-Algorithmus eine sehr gute durchschnittliche Laufzeit, da er norma-

lerweise nur wenige Ecken eines Polyeders besucht. Außerdem genügt es hier, Phase I des Simplex-Algorithmus durchzuführen. Wir können darüber hinaus etliche fortschrittliche Optimierungen des Algorithmus verwenden. Die Komplexität des Lösens der Zulässigkeitsprobleme ist für die Gesamtkomplexität des Algorithmus entscheidend, da dieser Schritt aufwändig ist und innerhalb der Schleife wiederholt ausgeführt wird. Der nächste Schritt hängt dann vom Ergebnis des Zulässigkeitsproblems ab. Gibt es eine zulässige Lösung, so wird eine entsprechende Stelle zu dem Netz hinzugefügt. Gibt es keine Lösung, so kann dies gespeichert werden. In diesem Fall hat das Syntheseproblem auf jeden Fall eine negative Antwort. Nach dieser Fallunterscheidung wird dann die nächste falsche BPO-Fortsetzung bearbeitet. Wenn schließlich alle falschen BPO-Fortsetzungen betrachtet wurden, wird das konstruierte Netz als Ergebnis des Algorithmus ausgegeben.
Neben der Komplexität spielt, wie schon gesagt, auch die Größe des von dem Algorithmus synthetisierten Netzes eine wichtige Rolle. Hiebei ist für uns die Anzahl der Stellen entscheidend. Die Anzahl der Stellen des Netzes hängt insbesondere davon ab, wie häufig der Test, ob eine falsche BPO-Fortsetzung in dem bis dato konstruierten Netz aktiviert ist, positiv ausfällt. Für jeden positiven Test wird eine Stelle, welche die falsche BPO-Fortsetzung verhindert, zum Netz hinzugefügt, falls es eine solche Stelle gibt. Im Falle eines negativen Testergebnisses wird keine Stelle hinzugefügt. Wir haben schon diskutiert, dass auf diese Weise eine exponentielle Anzahl an Stellen entstehen kann, die Anzahl normalerweise aber klein ist, da typischerweise wenige positive Ergebnisse des Aktiviertheitstests vorkommen. Eine genaue Einschätzung der Anzahl ist allerdings sehr schwierig. Dies liegt daran, dass der Aktiviertheitstest von den bis dato berechneten Stellen abhängt. Damit ist er von der Reihenfolge der Abarbeitung der falschen BPO-Fortsetzungen sowie von den durch den Simplexalgorithmus berechneten Stellen abhängig (der Simplexalgorithmus berechnet jeweils eine von vielen möglichen Lösungen der Zulässigkeitsprobleme). Insgesamt sollte die Anzahl der Stellen also eher gering sein, allerdings kann dies im Allgemeinen nicht garantiert werden. Wie schon erläutert, können wir nach der eigentlichen Berechnung des Netzes auch noch Verfahren, um im nachhinein implizite Stellen des Netzes zu entfernen, verwenden.
Als nächstes betrachten wir Algorithmus 4.4.1 zur Berechnung einer Schrittseparations-Repräsentation. Dieser Algorithmus ist genauso aufgebaut wie der gerade ausführlich diskutierte Algorithmus 4.4.2. Der einzige Unterschied besteht darin, dass falsche Schrittfortsetzungen anstelle von falschen BPO-Fortsetzungen verwendet werden. Daher lassen sich sehr ähnliche Komplexitätsüberlegungen wie für Algorithmus 4.4.2 durchführen. Von der Größenordnung her ergeben sich für die einzelnen Schritte von Algorithmus 4.4.1 sowie dann natürlich auch für den gesamten Algorithmus 4.4.1 dieselben Komplexitätsergebnisse wie im Falle von Algorithmus 4.4.2. Genauere Betrachtungen, wie in Abschnitt 4.4 durchgeführt, zeigen, dass der Rechenaufwand von Algorithmus 4.4.1 allerdings insgesamt etwas größer ist. Dies liegt daran, dass die Menge der falschen Schrittfortsetzungen von der Größenordnung her zwar mit der Menge der falschen BPO-Fortsetzungen übereinstimmt, aber dennoch im Allgemeinen immer etwas größer ist. Überlegungen zur Größe der von Algorithmus 4.4.1 erzeugten Netze lassen sich auch analog wie im Falle von Algorithmus 4.4.2 führen. Auch hier muss

beachtet werden, dass die Menge der falschen Schrittfortsetzungen größer ist als die Menge der falschen BPO-Fortsetzungen. Dennoch sollten die von den zwei Algorithmen erzeugten Netze normalerweise ungefähr gleich groß sein.
Als letztes diskutieren wir Algorithmus 4.4.3 zur Berechnung einer Erzeugendensystem-Repräsentation. Nach der schon angesprochenen Konstruktion der Matrix zur linear algebraischen Charakterisierung der verwendeten Regionendefinition folgt der wesentliche Schritt des Algorithmus. Es wird ein ganzzahliges endliches Erzeugendensystem eines spitzen rationalen polyedrischen Kegels berechnet. Hierzu wird das Extremalstrahl-Aufzählungs-Problem für den Kegel gelöst und somit das eindeutige minimale Erzeugendensystem konstruiert. Der Kegel ist durch das von der konstruierten Matrix definierte Ungleichungssystem gegeben. Die Größe des Ungleichungssystems ergibt sich also wieder durch die Größe dieser Matrix. Die Größen der hier in Frage kommenden Matrizen wurden schon diskutiert. Die Komplexität des Lösens des Extremalstrahl-Aufzählungs-Problems ist natürlich stark von der Größe der Matrix abhängig. Es wurden in Abschnitt 3.4 verschiedene mögliche Lösungsverfahren diskutiert. Alle Verfahren haben im schlechtesten Fall exponentielle Laufzeit sowohl in der Größe der Eingabe als auch in der Größe der Ausgabe. Wir verwenden hier eine fortschrittliche Weiterentwicklung der doppelten Beschreibungsmethode. Diese hat für typische praktische Probleme eine sehr gute Durchschnittslaufzeit. Insbesondere kann sie auch davon profitieren, dass die bei uns auftretenden Matrizen meist viele Null-Einträge beinhalten.
Nach der Berechnung des Erzeugendensystems wird die Beschriftungsmenge der partiellen Sprache betrachtet und als Menge der Transitionen des Netzes festgelegt. Schließlich wird für jedes Element des berechneten Erzeugendensystems eine korrespondierende Stelle zum Netz hinzugefügt. Das resultierende Netz wird als Ergebnis des Algorithmus ausgegeben.
Die Anzahl der Stellen des synthetisierten Netzes ergibt sich durch die Größe des berechneten Erzeugendensystems. Das Erzeugendensystem ist häufig relativ groß und kann sogar exponentiell in der Größe der betrachteten Matrix sein. Eine genauere Abschätzung ist hier schwierig, da die Größen stark variieren können. Es gibt aber Ergebnisse, welche darauf hindeuten, dass in vielen praktischen Beispielen kein exponentielles Wachstum vorliegt (siehe Abschnitt 3.4). Insgesamt sollte die Anzahl der Stellen also eher groß sein, aber sich in einem Rahmen bewegen, der für praktische Zwecke noch nutzbar ist. Auch hier können wir nach der eigentlichen Berechnung des Netzes noch Verfahren verwenden, um implizite Stellen des Netzes zu entfernen.
Zusammenfassend lässt sich sagen, dass die zwei Berechnungsverfahren zur Konstruktion einer Schrittseparations-Repräsentation bzw. einer BPO-Separations-Repräsentation im Normalfall effizienter sind als die Berechnung einer Erzeugendensystem-Repräsentation. Zwar sind auch bei den ersteren beiden Verfahren umfangreiche Berechnungsschritte notwendig, diese sind aber meist nicht so aufwändig wie das Lösen des Extremalstrahl-Aufzählungs-Problems. Hierbei ist Algorithmus 4.4.2 im Normalfall leicht effizienter als Algorithmus 4.4.1. Das Problem von Algorithmus 4.4.3 besteht darin, dass es schwierig ist, ein vollständiges Erzeugendensystem aller Regionen zu berechnen. Bei Algorithmus 4.4.2 und Algorithmus 4.4.1 werden nur einzelne ausgewählte Regionen berechnet. Entsprechend ist natürlich auch die Anzahl der berechneten

Stellen bei Algorithmus 4.4.3 normalerweise wesentlich größer. Hier wird eine Stelle für jede Region des Erzeugendensystems betrachtet. Bei Algorithmus 4.4.2 und Algorithmus 4.4.1 entstehen in manchen Fällen sehr kleine Netze, da häufig wenige Stellen ausreichen um das nicht-spezifizierte Verhalten auszuschließen. Allerdings muss noch beachtet werden, dass Algorithmus 4.4.2 und Algorithmus 4.4.1 direkt von der Struktur der partiellen Sprache abhängen. Bei Sprachen mit viel nebenläufigem Verhalten ergeben sich große Mengen an falschen Fortsetzungen, die betrachtet werden müssen. In solchen Fällen sind diese Verfahren somit aufwändiger. Dahingegen hängt Algorithmus 4.4.3 nicht direkt von der Struktur der partiellen Sprache ab. Diese spielt hier nur indirekt über das betrachtete Ungleichungssystem eine Rolle. Somit sollte bei Algorithmus 4.4.2 abgesehen von der Größe des Ungleichungssystems kein systematischer Unterschied zwischen Sprachen mit viel nebenläufigem und wenig nebenläufigem Verhalten bestehen.

Übereinstimmungstests

Nun werden noch der optimistische und der pessimistische Übereinstimmungstest verglichen. Diese sind in Algorithmus 4.5.1 und Algorithmus 4.5.2 beschrieben. Die Komplexität der beiden Algorithmen hängt insbesondere auch von der Größe des zuvor konstruierten Netzes ab. Diese Größe wurde schon im bisherigen Verlauf des Unterabschnittes diskutiert. Wir beginnen mit Algorithmus 4.5.1. Der Schritt des Hinzufügens der Stellen st^t zum betrachteten Netz wird in unserer Implementierung übersprungen. Dann folgt der aufwändige Schritt der Berechnung aller Prozess-BPOs mit maximaler Länge des betrachteten Netzes. Hier spielt nun die Größe des Netzes eine wichtige Rolle. An dieser Stelle sollten insbesondere auch die Kantengewichte und die Anfangsmarkierung des Netzes zur Größe hinzugerechnet werden, da diese für das Ablaufverhalten bedeutsam sind. Sowohl der Aufwand zur Berechnung der BPOs als auch die Anzahl der berechneten BPOs kann exponentiell in der Größe des Netzes sein. Nun verwenden wir aber, wie in Unterabschnitt 4.5.1 dargestellt, einen innovativen sehr effizienten Entfaltungs-Algorithmus zur Berechnung der BPOs. Dieser ist zwar im schlechtesten Fall immer noch exponentiell in der Größe des Netzes, aber wesentlich schneller als vergleichbare Algorithmen. Darüber hinaus ist in unserer Situation zu erwarten, dass die Anzahl der maximalen Prozess-BPOs von (N, m_0) ungefähr mit der Größe von $\mathcal{L}$ übereinstimmt, da gerade ein Netz mit dem von $\mathcal{L}$ spezifizierten Ablaufverhalten synthetisiert werden soll. Natürlich lässt sich im Allgemeinen keine genaue Übereinstimmung erwarten, da zum einen die Menge der Prozess-BPOs Sequentialisierungen enthalten kann, welche nicht in $\mathcal{L}$ gegeben sind, und es zum anderen nicht immer ein Netz mit dem Ablaufverhalten $PS(\mathcal{L})$ gibt. Dennoch ist nach diesen Überlegungen die Anzahl der berechneten BPOs in vielen Fällen linear in der Eingabe $\mathcal{L}$. In solchen Fällen ergibt sich gerade mit dem von uns verwendeten effizienten Entfaltungs-Algorithmus oft auch ein gutartiger Berechnungsaufwand.
Die Anzahl der berechneten BPOs ist nun wichtig, da jede dieser BPOs weiter verarbeitet wird. Für jede BPO wird geprüft, ob sie eine Sequentialisierung einer BPO aus $\mathcal{L}$ ist. Hierzu wird für die betrachtete BPO zusammen mit jeder BPO aus $\mathcal{L}$ ein entsprechend modifizierter Isomorphietest durchgeführt. Dabei werden die Knoten beider BPOs

jeweils in charakteristische Äquivalenzklassen eingeteilt. Dann werden die Knotenanzahlen in den entsprechenden Klassen verglichen. Stimmen diese nicht überein, so kann direkt eine negative Antwort auf den Test gegeben werden. Dies ist häufig der Fall. Gilt an dieser Stelle Übereinstimmung, so wird ein Backtrackingverfahren verwendet, welches alle möglichen modifizierten Isomorphismen testet. Im Rahmen einer „branch-and-bound"-Strategie werden aber nur die Knoten aus entsprechenden Äquivalenzklassen aufeinander abgebildet. Generell gilt, dass der Aufwand eines solchen Graphisomorphietests, bei dem sich, wie hier beschrieben, eine effektive „branch-and-bound"-Strategie finden lässt, im schlechtesten Fall zwar exponentiell in der Größe der betrachteten BPOs sein kann, normalerweise solche Fälle aber nicht vorkommen. Meistens lässt sich der Test sogar in linearer Zeit durchführen. Ergibt sich im Rahmen dieser Tests der Fall, dass eine betrachtete Prozess-BPO von keiner BPO aus $\mathcal{L}$ eine Sequentialisierung ist, so gibt der Übereinstimmungstest direkt eine negative Antwort aus. Eine positive Antwort wird ausgegeben, falls alle berechneten Prozess-BPOs abgearbeitet sind und hierbei keine negative Antwort ausgegeben wurde.
Der pessimistische Übereinstimmungstest aus Algorithmus 4.5.2 beginnt mit der Berechnung der Repräsentation $\mathrm{PS}(\mathcal{L})^c$ des Komplements von $\mathrm{PS}(\mathcal{L})$. Die Größe dieser Menge ist typischerweise exponentiell in der Größe von $\mathcal{L}$. Entsprechend erfordert auch die Berechnung von $\mathrm{PS}(\mathcal{L})^c$ exponentielle Laufzeit. Jede BPO aus $\mathrm{PS}(\mathcal{L})^c$ enthält einen Knoten mehr als eine entsprechende BPO aus $\mathrm{PS}(\mathcal{L})$. Eine polynomielle Größe von $\mathrm{PS}(\mathcal{L})^c$ ergibt sich für partielle Sprachen $\mathcal{L}$, welche sehr wenig nebenläufiges Verhalten beinhalten, da deren BPOs nur wenige Präfixe und Sequentialisierungen besitzen. In diesem Fall kann auch die Berechnung von $\mathrm{PS}(\mathcal{L})^c$ in polynomieller Zeit durchgeführt werden.
Die Größe von $\mathrm{PS}(\mathcal{L})^c$ ist nun wichtig, da im Weiteren jede BPO aus $\mathrm{PS}(\mathcal{L})^c$ untersucht wird. Es wird für jede BPO geprüft, ob sie bzgl. des betrachteten Netzes aktiviert ist. Hierfür spielt nun natürlich die Größe des Netzes eine wichtige Rolle. Wir verwenden, wie in Unterabschnitt 4.5.2 beschrieben, einen auf Flussoptimierungsalgorithmen basierenden Test, ob eine BPO in einem S/T-Netz aktiviert ist. Dieser Test erfordert polynomiellen Aufwand in der Größe des Netzes und der BPO. Resultiert bei einem der durchzuführenden Aktiviertehitstests ein positives Ergebnis, so gibt der Übereinstimmungstest direkt eine negative Antwort aus. Eine positive Antwort wird ausgegeben, falls alle BPOs aus $\mathrm{PS}(\mathcal{L})^c$ abgearbeitet sind und hierbei alle Aktiviertheitstests negativ waren.
Zusammenfassend erfordern zwar beide Algorithmen im schlechtesten Fall exponentielle Laufzeit, obige Überlegungen zeigen aber, dass ein solcher Fall beim optimistischen Übereinstimmungstest eher selten zu sein scheint. Dahingegen ergibt sich beim pessimistischen Übereinstimmungstest nur für partielle Sprachen mit wenig Nebenläufigkeit eine polynomielle Laufzeit. Wir vermuten daher, dass meist der optimistische Übereinstimmungstest zu bevorzugen ist. Allerdings hat die Größe des zuvor berechneten Netzes auf den optimistischen Übereinstimmungstest eine größere Auswirkung. Dies liegt daran, dass hier ein aufwändiger Entfaltungsalgorithmus auf dieses Netz angewandt wird. Beim pessimistischen Übereinstimmungstest wird das Netz nur im Rahmen effizienter Aktiviertheitstests benötigt. Daher hat der pessimistische Übereinstimmungstest tendenziell Komplexitätsvorteile, wenn das betrachtete Netz sehr groß ist.

4.7.2 *Experimenteller Vergleich*

Nun vergleichen wir die verschiedenen Syntheseverfahren empirisch durch experimentelle Tests. Hierzu verwenden wir die Implementierungen der verschiedenen relevanten Algorithmen in VipTool (vgl. Abschnitt 4.6). Wir testen die Laufzeit und die Größe der synthetisierten Netze. Wir beginnen mit systematischen Tests, bei welchen wir verschiedene Parameter der Eingabe, also der betrachteten partiellen Sprache, variieren. Anschließend wenden wir die Verfahren noch auf realistische Beispiele an.
Wir beschränken uns bei den Tests auf den zentralen konstruktiven Teil der Syntheseverfahren, d.h. wir betrachten die Übereinstimmungstest-Verfahren hier nicht (bisher haben wir auch nur den optimistischen Übereinstimmungstest implementiert). Weiter behandeln wir nur die Verfahren zur Berechnung einer BPO-Separations-Repräsentation und einer Erzeugendensystem-Repräsentation jeweils unter Verwendung von BPO-Transitions-Regionen und Markenfluss-Regionen, d.h. wir zeigen Tests der Plug-Ins LSepSynthesis und GeneratingSynthese jeweils sowohl zusammen mit dem Plug-In LTransRegions als auch mit dem Plug-In TokenFlowRegions. Wir haben schon zu Beginn des Abschnittes erläutert, dass sich die Berechnung einer Schrittseparations-Repräsentation sehr ähnlich wie die Berechnung einer BPO-Separations-Repräsentation verhält und dass Schritt-Transitions-Regionen sehr ähnlich zu BPO-Transitions-Regionen sind. Daher sind für die Plug-Ins StepSepSynthesis bzw. StepTransRegions analoge Testergebnisse wie für die Plug-Ins LSepSynthesis bzw. LTransRegions zu erwarten. Dies hat sich bei entsprechenden Tests auch bestätigt. Diese Tests werden aber an dieser Stelle nicht weiter dargestellt. Schließlich ist noch zu beachten, dass wir bei den folgenden Testergebnissen bewusst keine impliziten Stellen beispielsweise mit dem PostProcessing-Plug-In entfernen. Es soll hier die Größe der rein durch die verwendeten Syntheseverfahren berechneten Netze verglichen werden.
Die Testumgebung gestaltet sich folgendermaßen. Alle Tests werden auf einem Intel Core2Duo 2.66 GHz (2 CPUs) Rechner mit 2012 MB RAM durchgeführt. Als Betriebssystem wird die Microsoft Windows Vista Business-Edition verwendet. Es ist die Laufzeitumgebung von Java SE 1.6.0 installiert. Um Nachvollziehbarkeit der Testergebnisse zu gewährleisten, stellen wir alle verwendeten Eingabedaten als XML-Dateien auf der VipTool-Homepage `viptool.ku-eichstaett.de` zur Verfügung. Tatsächlich spielt nämlich bei den Syntheseverfahren die Reihenfolge, in welcher die BPOs und die einzelnen Knoten und Kanten der BPOs abgespeichert sind, eine Rolle. Dies liegt daran, dass dadurch in einigen Fällen die Reihenfolge der Abarbeitung beeinflusst wird. Die Reihenfolge der BPOs ist hierbei durch die Dateinamen festgelegt. Problematisch ist aber die Reihenfolge der Knoten und Kanten der BPOs, welche davon abhängt, in welcher Reihenfolge sie in VipTool gezeichnet werden. Daher müssen zur Reproduktion der vorgestellten Ergebnisse die bereitgestellten Dateien verwendet werden.

Systematische Tests

Bei den systematischen Tests versuchen wir insbesondere, die Abhängigkeit der Performance der Syntheseverfahren von der Größe und der Struktur der betrachteten partiellen Sprache besser zu verstehen. Wir haben umfassende Tests durchgeführt. Die Hauptergebnisse der

Tests stellen wir hier anhand von vier Testreihen vor. In der ersten Reihe betrachten wir drei total geordnete BPOs a_n, b_n, c_n. Es wird die Anzahl der sequentiell geordneten Knoten in den BPOs erhöht (siehe Abbildung 79 oben links). In der zweiten Reihe wird eine BPO abc_n, welche aus drei nebenläufigen Sequenzen von Knoten besteht, untersucht. Wiederum wird die Anzahl der Knoten in der BPO erhöht (siehe Abbildung 79 oben rechts). In der dritten Reihe werden BPOs $l_1 \dots l_n$ untersucht, welche sich nur durch die Beschriftung des letzten Knotens unterscheiden. Hierbei wird die Anzahl der betrachteten BPOs erhöht (siehe Abbildung 79 unten links). Zuletzt betrachten wir eine BPO $axbc_n$, welche nur sehr wenige Kanten beinhaltet. Wir erhöhen hierbei die hauptsächlich nebenläufig angeordneten Knoten (siehe Abbildung 79 unten rechts). Zur Illustration zeigen wir in Abbildung 80 vier S/T-Netze, welche das in den vier Testreihen verwendete Ablaufverhalten aufweisen. Diese sind also mögliche Ergebnisse der Syntheseverfahren. Sie stellen jeweils die kompakteste Repräsentation des betrachteten Verhaltens durch ein S/T-Netz dar. Insbesondere lässt sich in allen vier Testreihen das spezifizierte Verhalten exakt reproduzieren. Daher haben die synthetisierten Netze in allen Beispielen das spezifizierte Ablaufverhalten.

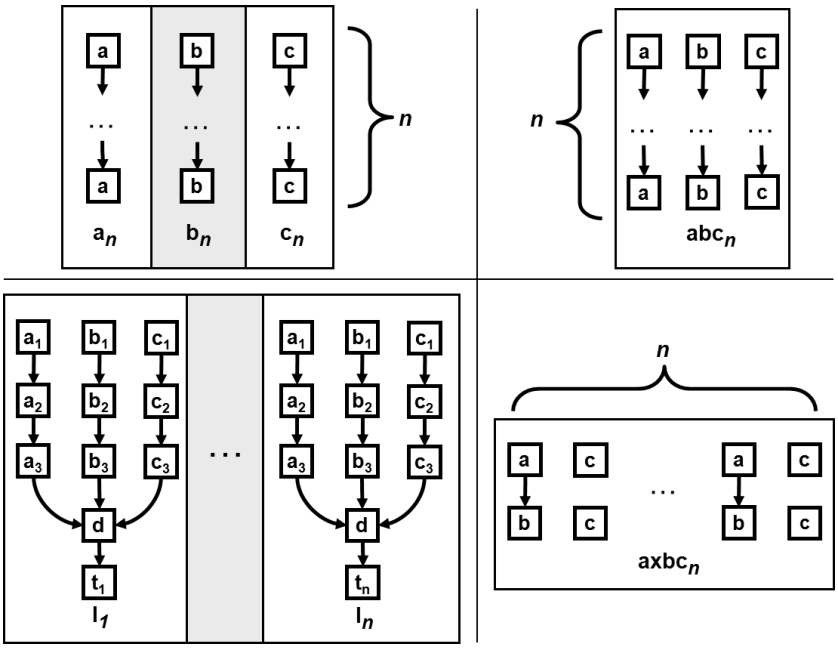

Abbildung 79: Testreihen. Oben links: Drei BPOs, welche drei alternative Sequenzen modellieren. Oben rechts: Eine BPO, welche drei nebenläufige Sequenzen modelliert. Unten links: n BPOs, welche eine Alternative von n abschließenden Transitionen modellieren. Unten rechts: Eine BPO mit hauptsächlich nebenläufigen Ereignissen.

Die Ergebnisse der Tests sind in den Tabellen der Abbildungen 81 und 82 dargestellt. Sie sind nach dem verwendeten Algorithmus aufgeteilt. In jeder Zeile werden die Ergebnisse zu einer als Eingabe verwendeten partiellen Sprache aufgeführt. Die erste Spalte gibt jeweils die betrachtete partielle Sprache an. Die weiteren Spalten werden bzgl. der verwendeten Regionendefinition unterschieden. Für eine Regionendefinition werden folgende Testergebnisse aufgeführt. Eine erste

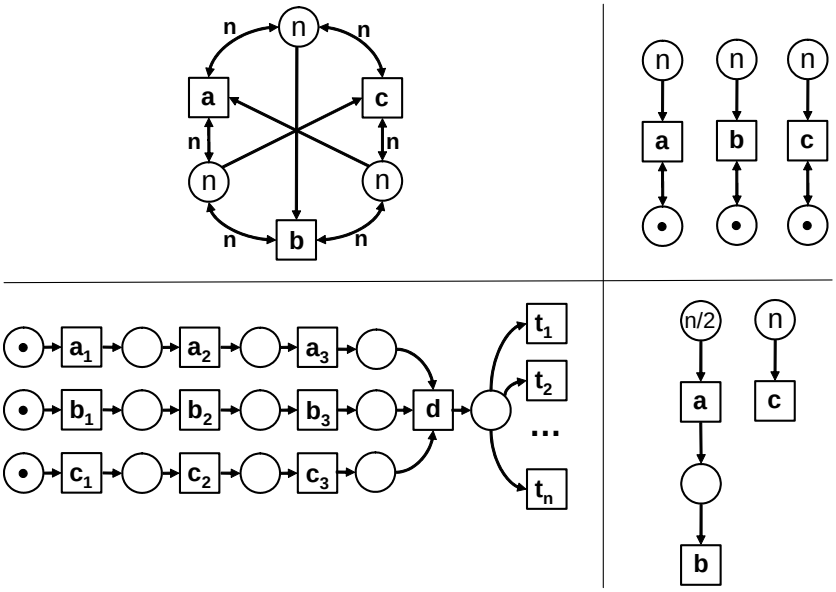

Abbildung 80: S/T-Netze, deren Ablaufsemantiken mit den entsprechenden partiellen Sprachen aus Abbildung 79 übereinstimmen.

Spalte enthält die mittlere Gesamtlaufzeit von 50 Testdurchführungen des Algorithmus in ms. In Klammern sind die Abweichungen des entsprechenden 95%-Konfidenzintervalls für den Erwartungswert der Gesamtlaufzeit in ms angegeben. Eine zweite Spalte enthält die Anzahl der durch den Algorithmus synthetisierten Stellen. Im Falle der Berechnung einer BPO-Separations-Repräsentation (Abbildung 81) wird die Gesamtlaufzeit noch einmal aufgeteilt. Eine dritte Spalte bei beiden Regionendefinitionen gibt hierbei die Durchschnittslaufzeit für die Berechnung einer zulässigen Stelle, d.h. zum Lösen eines entsprechenden Zulässigkeitsproblems, an. In der allerletzten Spalte der Tabelle ist die für beide Regionendefinitionen übereinstimmende Durchschnittslaufzeit zur Berechnung der Menge der falschen BPO-Fortsetzungen dargestellt. An den Stellen, an denen ein „-“ eingetragen ist, war eine Berechnung nicht möglich. Dies lag meist daran, dass ein Speicherüberlauf aufgetreten ist.
Im Hinblick auf die verwendete Regionendefinition ergeben sich folgende Ergebnisse bzgl. der Laufzeit der Verfahren. Die in der ersten Testreihe betrachteten partiellen Sprachen enthalten keinerlei nebenläufiges Verhalten. In dieser Testreihe ist die Laufzeit der Verfahren bei der Verwendung von BPO-Transitions-Regionen signifikant besser als bei der Verwendung von Markenfluss-Regionen. Im Falle der Berechnung einer Erzeugendensystem-Repräsentation mit Markenfluss-Regionen ergeben sich sogar gar keine Ergebnisse mehr.
In der zweiten Testreihe ergeben sich entgegengesetzte Ergebnisse. Die betrachtete partielle Sprache beinhaltet einen hohen Grad an Nebenläufigkeit. Hier ist die Verwendung von Markenfluss-Regionen besser als die Verwendung von BPO-Transitions-Regionen. Bei der Berechnung einer BPO-Separations-Repräsentation ist zwar die Laufzeit für das Lösen der einzelnen Ungleichungssysteme bei Markenfluss-Regionen signifikant besser, der Unterschied in der Gesamtlaufzeit ist bei den zwei Regionendefinitionen aber nicht besonders groß. Dies liegt daran, dass in Beispielen mit einem hohen Maß an Nebenläufigkeit der bei

Berechnung BPO-Separations-Repräsentation

Sprache	BPO-Transitions-Regionen			Markenfluss-Regionen			Erstellungszeit falsche BPO-Fortsetzungen
	Gesamtlaufzeit (ms)	Anzahl an Stellen	Zeit pro Zulässigkeitsproblem	Gesamtlaufzeit (ms)	Anzahl an Stellen	Zeit pro Zulässigkeitsproblem	
$\{a_{20}, b_{20}, c_{20}\}$	12 (±1)	3	1	455 (±3)	16	28	7
$\{a_{25}, b_{25}, c_{25}\}$	15 (±1)	3	2	853 (±5)	16	52	9
$\{a_{30}, b_{30}, c_{30}\}$	20 (±1)	3	2	1462 (±7)	17	85	14
$\{abc_9\}$	278 (±6)	6	17	209 (±5)	6	5	176
$\{abc_{11}\}$	683 (±8)	6	30	588 (±7)	9	10	498
$\{abc_{13}\}$	1499 (±9)	6	54	1301 (±9)	7	18	1171
$\{l_1 \ldots l_{20}\}$	288 (±6)	31	4	1238 (±6)	31	35	140
$\{l_1 \ldots l_{30}\}$	501 (±7)	31	6	2782 (±11)	31	80	297
$\{l_1 \ldots l_{40}\}$	834 (±8)	31	8	3194 (±12)	23	114	559
$\{axbc_4\}$	735 (±7)	3	4	729 (±7)	3	2	723
$\{axbc_6\}$	1829 (±11)	3	6	1820 (±11)	3	3	1811
$\{axbc_8\}$	163436 (±152)	3	9	163421 (±152)	3	4	163409
Apoptosis	9145 (±39)	22	97	8898 (±18)	29	65	7006
Transit	5563 (±22)	168	30	120189 (±107)	399	300	380
Prod./cons.	56370 (±78)	23	65	55024 (±75)	23	7	54854
Log file	5253 (±19)	59	68	-	-	-	1233

Abbildung 81: Testergebnisse bei Berechnung einer BPO-Separations-Repräsentation

Berechnung Erzeugendensystem-Repräsentation

Sprache	BPO-Transitions-Regionen		Markenfluss-Regionen	
	Gesamtlaufzeit (ms)	Anzahl an Stellen	Gesamtlaufzeit (ms)	Anzahl an Stellen
$\{a_{20}, b_{20}, c_{20}\}$	699 (±4)	30	-	-
$\{a_{25}, b_{25}, c_{25}\}$	721 (±4)	30	-	-
$\{a_{30}, b_{30}, c_{30}\}$	756 (±5)	30	-	-
$\{abc_3\}$	906 (±6)	10	163 (±2)	25
$\{abc_5\}$	4301 (±19)	10	621 (±5)	772
$\{abc_7\}$	23482 (±36)	10	-	-
$\{l_1, l_2\}$	1601 (±8)	74	3057 (±10)	10892
$\{l_1 \ldots l_5\}$	2015 (±10)	385	-	-
$\{l_1 \ldots l_{10}\}$	10087 (±23)	11302	-	-
$\{axbc_4\}$	566 (±5)	8	138 (±2)	10
$\{axbc_6\}$	1496 (±7)	8	166 (±2)	14
$\{axbc_8\}$	137738 (±134)	8	174 (±2)	22
Apoptosis	-	-	-	-
Transit	-	-	-	-
Prod./cons.	-	-	1109 (±6)	241
Log file	-	-	-	-

Abbildung 82: Testergebnisse bei Berechnung einer Erzeugendensystem-Repräsentation

beiden Regionendefinitionen notwendige Aufwand zur Berechnung der falschen Fortsetzungen hoch ist. An dieser Stelle ist anzumerken, dass gerade dieser Teil noch nicht Laufzeit-optimiert implementiert ist. Bei der Berechnung einer Erzeugendensystem-Repräsentation ergibt sich bei den ersten beiden Tests für Markenfluss-Regionen eine deutlich bessere Laufzeit als für BPO-Transitions-Regionen. Allerdings liefert die Verwendung von Markenfluss-Regionen beim dritten Test gar kein Ergebnis. Es scheint hier so, dass das verwendete Verfahren zur Berechnung eines Erzeugendensystems sehr sensitiv gegenüber einer stark wachsenden Anzahl an Variablen ist.
In der dritten Testreihe werden partielle Sprachen, welche aus einer großen Anzahl von relativ kleinen BPOs bestehen, betrachtet. Die einzelnen BPOs sind sehr ähnlich. Dies ist ein typischer Fall, in dem im Rahmen von BPO-Transitions-Regionen die Anzahl der zu betrachtenden Schnittfortsetzungen im Verhältnis zur Anzahl der Schnitte der Sprache klein ist. Daher ist hier die Verwendung von BPO-Transitions-Regionen effizienter als die Verwendung von Markenfluss-Regionen, obwohl die partielle Sprache einen relativ hohen Grad an Nebenläufigkeit aufweist.
Die BPOs der vierten Testreihe weisen ein sehr hohes Maß an Nebenläufigkeit auf. Die Verwendung von Markenfluss-Regionen ist hier besser als die Verwendung von BPO-Transitions-Regionen. Allerdings ergeben sich bei BPO-Transitions-Regionen bei einem derart hohen Maß an Nebenläufigkeit auch gute Ergebnisse. Allerdings ist die Be-

rechnung einer linear algebraischen Charakterisierung entsprechender BPO-Transitions-Regionen relativ aufwändig. Besonders problematisch ist bei einer solchen Sprache wiederum die Berechnung der falschen BPO-Fortsetzungen.
Im Vergleich der beiden Berechnungsverfahren zeigt sich insgesamt, dass zumeist die Berechnung einer BPO-Separations-Repräsentation deutlich effizienter ist als die Berechnung einer Erzeugendensystem-Repräsentation. Die einzige Ausnahme hierzu stellt die vierte Testreihe dar. Bei dem sehr hohen Grad an Nebenläufigkeit dieser Testreihe ist die Berechnung der Menge der falschen BPO-Fortsetzungen sehr aufwändig. Daher ergeben sich hier im Rahmen der Berechnung einer Erzeugendensystem-Repräsentation bessere Ergebnisse.
Es ist schwierig, allgemeine Aussagen über die Anzahlen der generierten Stellen zu treffen. Wir betrachten zuerst wieder die Abhängigkeit von der verwendeten Regionendefinition. Bei der Berechnung einer Erzeugendensystem-Repräsentation ergibt sich bei Markenfluss-Regionen eine deutlich höhere Anzahl an Stellen als bei BPO-Transitions-Regionen. Insbesondere das Wachstum der Anzahlen bei größer werdenden Beispielen ist bei Markenfluss-Regionen meist sehr stark, während sich bei BPO-Transitions-Regionen teilweise gar kein Wachstum ergibt. Bei der Berechnung einer BPO-Separations-Repräsentation stellt sich die Situation anders dar. Während in der ersten Testreihe bei der Verwendung von Markenfluss-Regionen signifikant mehr Stellen berechnet werden als bei der Verwendung von BPO-Transitions-Regionen, sind die Anzahlen an Stellen in der zweiten, der dritten und der vierten Testreihe meist gleich. In der ersten und zweiten Testreihe werden bei der Verwendung von BPO-Transitions-Regionen sogar die entsprechenden Netze aus Abbildung erzeugt, welche eine minimale Anzahl an Stellen enthalten. In der vierten Testreihe gilt dies für beide Regionendefinitionen. Ein Wachstum der Stellenanzahlen bei wachsenden Beispielen lässt sich in allen Fällen nicht beobachten.
Beim Vergleich der zwei Berechnungsverfahren bzgl. der Anzahlen der erzeugten Stellen schneidet das Verfahren zur Berechnung einer BPO-Separations-Repräsentation durchgängig wesentlich besser ab als das Verfahren zur Berechnung einer Erzeugendensystem-Repräsentation. Im Falle von BPO-Transitions-Regionen hält sich der Unterschied zumindest bei der zweiten und vierten Testreihe noch in Grenzen.
Insgesamt bestätigen die Testergebnisse im wesentlichen die theoretischen Überlegungen des letzten Unterabschnittes. Allerdings ergeben sich auch einige darüber hinausgehende Erkenntnisse. Außerdem vermitteln die Tests einen guten Eindruck von den Größenordnungen der Unterschiede der Syntheseverfahren.

Anwendungsbeispiele

Nun werden die Syntheseverfahren in realistischen Beispielen angewendet. Als Beispiele betrachten wir zum einen zwei Petrinetze aus realen Fallstudien und ein klassisches für Anwendungen bedeutsames Petrinetz. Für diese Petrinetze berechnen wir die Abläufe in der Form von BPOs und verwenden die resultierenden partiellen Sprachen als Eingaben für die Syntheseverfahren. Zum anderen betrachten wir eine partielle Sprache, welche aus einer realen Log-Datei eines Geschäftsprozesses generiert ist. Auf diese vier partiellen Sprachen wenden wir die Syntheseverfahren an.

Das erste Beispiel ist ein Netz, welches von der DSSZ-Gruppe aus Cottbus entwickelt wurde (siehe `www.dssz.informatik.tu-cottbus.de`). Es modelliert den biologischen Prozess der Apoptose [115]. Die Autoren haben verschiedene Versionen des Modells präsentiert. Wir verwenden die Version, welche von der Gruppe zu Analysezwecken verwendet wurde (diese Version findet sich in verschiedenen Vortragsfolien auf der angegebenen Webseite, beispielsweise „Biopathways and Petri Nets - Demonstrated for Apoptosis“, „Modularization of Biochemical Networks by T-invariants“, „Biochemically Interpreted Petri Nets - Two Open Problems“ oder „Petri Nets for Systems and Synthetic Biology“). Anstelle der Eingabe-Transitionen haben wir entsprechende Marken in das Netz eingefügt. Dabei haben wir jede Eingabesubstanz zweimal zur Verfügung gestellt, um die Abläufe vielfältiger zu gestalten. Es ergeben sich sieben BPOs mit insgesamt 167 Knoten und 13 Beschriftungen. Aufgrund der nicht-sicheren Anfangsmarkierung weisen die BPOs einen gewissen Grad an nebenläufigem Verhalten auf.
Das zweite Beispiel ist die zweite Version eines Workflows zur Modellierung eines Kommunikationsablaufs innerhalb einer von einem Berater von Deloitte und Touche initiierten Fallstudie [215] (der „transit2“-Prozess findet sich auch auf `www.petriweb.org`). Um das Verhalten des Netzes endlich zu gestalten, haben wir drei Stellen zu dem Netz hinzugefügt, welche die Anzahl der von den drei zentralen (umrandeten) Kommunikationskanälen erlaubten Wiederholungen der Nachrichtenübermittlung aufgrund von fehlerhaften Nachrichten jeweils auf eine mögliche Wiederholung beschränken. Genauer gesagt haben wir jede der drei Transitionen zum Senden einer „fun-nck“-Nachricht („s_fun_nck“) durch eine Stelle, welche nur eine Marke enthält und nur eine zu der jeweiligen Transition gerichtete Kante besitzt, beschränkt. Dann ergeben sich acht BPOs mit insgesamt 288 Knoten und 36 verschiedenen Beschriftungen. In diesem Beispiel ist der Großteil der Kommunikation geordnet, so dass die BPOs wenig nebenläufiges Verhalten aufweisen.
Für ein Beispiel mit einem hohen Maß an Nebenläufigkeit haben wir ein Netz bestehend aus drei nebenläufig operierenden Instanzen des klassischen Produzent/Konsument-Petrinetzes [22], welches in vielen Petrinetzbüchern vorkommt, betrachtet (siehe auch `www.petriweb.org`). Dieses Netz spielt im Rahmen der Modellierung von Protokollen mit Petrinetzen häufig eine Rolle, da hierbei in etlichen Fällen dem Produzent/Konsument-Petrinetz ähnliche Strukturen vorkommen (siehe z.B. [129]). Für unser Beispiel haben wir die drei Produzenten jeweils derart beschränkt, dass sie nur zweimal produzieren dürfen. Dadurch garantieren wir ein endliches Ablaufverhalten. Es ergibt sich nur eine BPO mit 21 Knoten und zwölf verschiedenen Beschriftungen.
Das letzte Beispiel ist schließlich eine Log-Datei eines Geschäftsprozesses, welche mit dem Werkzeug ProM [2] geliefert wird. Das ProM-Team verwendet dieses Beispiel zum Testen der verschiedenen Process-Mining-Algorithmen von ProM. Der Name der mit ProM gelieferten Datei ist `grouped_a22f0n00.xml`. Es ist zu beachten, dass die Syntheseverfahren dieses Kapitels das betrachtete Verhalten exakt in ein Petrinetz übersetzen. Daher versuchen sie, genau die Anzahlen der Wiederholungen des Schaltens einzelner Transitionen in Abhängigkeit von den Schaltvorgängen anderer Transitionen zu reproduzieren. Da das Log unvollständig ist, würde dies zu einem extrem komplexen Netz führen. Deswegen haben wir die Log-Datei zuerst gefiltert, indem wir

alle Prozessinstanzen, welche mehr als eine Wiederholung der Schleifen-Transitionen m und i enthalten, entfernt haben. Um dann nicht nur Sequenzen, sondern partiell geordnetes Verhalten zu berücksichtigen, haben wir den schon in Unterabschnitt 2.2.2 angesprochenen Partial-Order-Generator aus ProM verwendet, welcher eine entsprechende Menge von BPOs aus einer Standard-Log-Datei extrahiert. Daraus ergeben sich 58 BPOs mit insgesamt 841 Knoten und 22 verschiedenen Beschriftungen. Das Beispiel weist ein für Geschäftsprozesse typisches moderates Maß an Nebenläufigkeit auf.
Die Resultate der experimentellen Tests für diese vier realistischen Beispiele sind in denselben Tabellen (Abbildungen 81 und 82) wie die Resultate der systematischen Tests dargestellt. Die Ergebnisse bestätigen im Wesentlichen unsere bisher gewonnenen Erkenntnisse.
Als Fazit aller Tests lässt sich zusammenfassen, dass mit der aktuellen Implementierung normalerweise die Berechnung einer BPO-Separations-Repräsentation der Berechnung einer Erzeugendensystem-Repräsentation sowohl von der Laufzeit als auch von der Größe der resultierenden Netze her vorzuziehen ist. Einzig bei einem sehr hohen Maß an Nebenläufigkeit kann die Berechnung einer Erzeugendensystem-Repräsentation zu Vorteilen führen. Bei dem Vergleich von BPO-Transitions-Regionen und Markenfluss-Regionen hat sich gezeigt, dass das erstere Regionen-Konzept normalerweise bei einem moderaten Maß an Nebenläufigkeit besser abschneidet, während das zweite Konzept bei viel Nebenläufigkeit Vorteile aufweist. In letzterem Fall hat sich aber meist nur eine leicht bessere Laufzeit bei Markenfluss-Regionen ergeben, so dass BPO-Transitions-Regionen insgesamt etwas besser abgeschnitten haben.

5

VERALLGEMEINERUNG DER SYNTHESEPROBLEMSTELLUNG

Nachdem wir im letzten Kapitel die exakte Synthese eines S/T-Netzes aus einer endlichen partiellen Sprache ausführlich besprochen haben, diskutieren wir in diesem Kapitel, wie sich die gewonnenen Erkenntnisse auf weitere Syntheseproblemstellungen verallgemeinern lassen. Wie in der Einleitung erläutert, stellt die in Kapitel 4 betrachtete Syntheseproblemstellung im Rahmen der Synthese eines Petrinetzes aus halbgeordneten Abläufen nur eine mögliche Fragestellung dar. Wir haben vier Bausteine identifiziert, welche für eine entsprechende Problemstellung relevant sind. Wir zeigen in den folgenden Abschnitten für jeden der vier Bausteine der Problemdimension unseres Synthesebaukastens, wie wir die in Kapitel 4 erarbeiteten Ergebnisse bzgl. einer in dem betrachteten Baustein veränderten Problemstellung anpassen müssen.

Die ersten zwei Bausteine der Problemdimension beziehen sich auf die Eingabe. Wir beginnen mit der als Eingabe verwendeten Sprachklasse. Hier haben wir uns bisher auf endliche partielle Sprachen beschränkt. In diesem Kapitel wollen wir die Synthese aus unendlichen partiellen Sprachen diskutieren. Hierzu müssen wir uns auf geeignete Klassen von unendlichen partiellen Sprachen einschränken und für solche Klassen jeweils eine geeignete Art der endlichen Repräsentation von entsprechenden unendlichen Sprachen entwickeln. Als zweites diskutieren wir alternative Sprachtypen zur Darstellung von Ablaufverhalten von Petrinetzen als Eingabe für das Syntheseproblem. In Kapitel 4 haben wir partielle Sprachen unter Verwendung der Ablaufsemantik von Petrinetzen zugrundegelegt. Hier werden wir weitere Möglichkeiten der Repräsentation von Ablaufverhalten diskutieren. Dann wenden wir uns den zwei die Ausgabe betreffenden Bausteinen der Problemdimension zu. In Kapitel 4 wird ein S/T-Netz als Ausgabe gefordert. In diesem Kapitel wollen wir andere Petrinetzklassen als Ausgabe zulassen. Weiter haben wir in Kapitel 4 das Problem der exakten Synthese aus einer Sprache behandelt, d.h. wir haben gefordert, dass das Ausgabenetz exakt das spezifizierte Ablaufverhalten besitzt. In diesem Kapitel wollen wir auch andere funktionale Abhängigkeiten des Ausgabenetzes von der Eingabe diskutieren, welche als Varianten im Rahmen einer Synthesefragestellung interessant sind.

Das Kapitel ist in vier Abschnitte gegliedert. Jeder Abschnitt behandelt einen Problem-Baustein. Für jeden Baustein wird mit einer kurzen praktischen Motivation begonnen. Danach wird gezeigt, wie sich die Ergebnisse des letzten Kapitels auf in dem betrachteten Baustein veränderte Problemstellungen übertragen lassen. Zum Erhalt der Lesbarkeit der Arbeit müssen wir uns hierbei jeweils auf ausgewählte zentra-

le Aspekte beschränken. Außerdem fallen die Ausführungen dieses Kapitels weniger detailliert aus als die des letzten Kapitels.

5.1 SPRACHKLASSEN

Für etliche Anwendung reicht die Betrachtung endlicher partieller Sprachen aus. Viele reale System-Spezifikationen definieren ein endliches Ablaufverhalten des betrachteten Systems. Daneben gibt es aber auch sehr häufig System-Spezifikationen, welche ein unendliches Verhalten festlegen. In den meisten realen Systemen lässt sich gewisses Verhalten im Prinzip unendlich oft wiederholen. Auf diese Weise entsteht unendliches Ablaufverhalten. Im Rahmen einer Verhaltensspezifikation stellt sich hier die Frage, wie sich solch unendliches Verhalten endlich darstellen lässt.
Wir wollen hierzu in diesem Abschnitt eine möglichst intuitive Herangehensweise wählen. Es soll möglich sein, den angesprochenen typischen Fall von sich wiederholendem Systemverhalten direkt anzugeben. In diesem Kontext führen wir einen entsprechenden Iterationsoperator für Ablaufverhalten ein. Um iterierbares Verhalten als Teil eines gesamten Ablaufverhaltens darstellen zu können, betrachten wir darüber hinaus Verknüpfungsoperatoren für Ablaufverhalten. Wir führen Operatoren zur sequentiellen, parallelen und alternativen Verknüpfung von Ablaufverhalten ein. Formal gesprochen betrachten wir Terme, welche aus beliebigen BPOs und den genannten Verknüpfungsoperatoren einschließlich eines Iterationsoperators entstehen.
Für zwei einzelne BPOs verwenden wir den sequentiellen und den parallelen Verknüpfungsoperator für BPOs, welche wir schon in Abschnitt 3.3 eingeführt haben. Die alternative Verknüpfung bzw. Vereinigung von zwei BPOs bedeutet, dass eine partielle Sprache mit diesen beiden BPOs definiert wird. Die Semantik einer iterierten BPO ist eine entsprechende unendliche partielle Sprache, welche über den in Abschnitt 3.3 eingeführten Potenzoperator für BPOs festgelegt wird. Diese Verknüpfungsprinzipien für einzelne BPOs werden anschließend auf Terme von BPOs verallgemeinert.
Im ersten Unterabschnitt führen wir zunächst entsprechende Terme zur Darstellung von unendlichem Ablaufverhalten ein. Wir verwenden dann die Konzepte des letzten Kapitels, um ein Syntheseverfahren zur Erzeugung eines S/T-Netzes, welches das durch einen Term spezifizierte Ablaufverhalten aufweist, zu entwickeln. Wir präsentieren hierzu auch eine entsprechende Implementierung. Wir betrachten also detailliert das im letzten Kapitel diskutierte Syntheseproblem, wobei wir als Eingabe nun die Sprachklasse der durch entsprechende Terme darstellbaren partiellen Sprachen verwenden. Allerdings deckt diese Sprachklasse nicht alle zu S/T-Netzen gehörenden Ablaufverhalten ab. Wir zeigen im zweiten Unterabschnitt, dass sich nur eine bestimmte Klasse von unendlichen partiellen Sprachen, auch von S/T-Netz-Sprachen, durch Terme repräsentieren lässt. Im Rahmen dieses Problems diskutieren wir, wie sich unendliche partielle Sprachen auf allgemeinere Art und Weise endlich repräsentieren lassen. Insbesondere zeigen wir beispielhaft eine allgemeinere, allerdings auch entsprechend komplexere, Möglichkeit einer endlichen Repräsentation für unendliche partielle Sprachen und übertragen die bisher gewonnenen Syntheseergebnisse auf diese Repräsentationsmöglichkeit.

5.1.1 *Synthese aus BPO-Termen*

Eine Spezifikation einer partiellen Sprache als Eingabe für einen Synthesealgorithmus muss endlich sein. Im letzten Kapitel haben wir uns auf endliche partielle Sprachen als Eingabe beschränkt. In diesem Abschnitt betrachten wir unendliche partielle Sprachen. Für diese ist dementsprechend eine geeignete endliche Repräsentation notwendig. In diesem Unterabschnitt verwenden wir eine termbasierte endliche Repräsentation unendlicher partieller Sprachen.
Wir führen sog. BPO-Terme ein. Diese verallgemeinern klassische reguläre Ausdrücke. Klassische reguläre Ausdrücke repräsentieren sequentielle Sprachen. Anstelle von klassischen Worten treten nun aber halbgeordnete Worte in der Form von BPOs. Dementsprechend ergeben sich einige Anpassungen. Während die Grundbausteine eines regulären Ausdruckes einzelne Buchstaben sind, gehen wir bei BPO-Termen von beliebigen BPOs als Grundbausteine aus. Dies ist insbesondere notwendig, um zu gewährleisten, dass sich beliebige endliche partielle Sprachen durch BPO-Terme repräsentieren lassen. Nur mithilfe der verwendeten Verknüpfungsoperatoren lassen sich nämlich nicht alle in BPOs möglichen Abhängigkeitsbeziehungen zwischen Knoten darstellen. Zur Verknüpfung von zwei Termen verwenden wir wie bei regulären Ausdrücken einen sequentiellen und einen alternativen Kompositionsoperator. Zur Repräsentation von unendlichem Verhalten verwenden wir wie üblich einen Iterationsoperator für Terme. Gegenüber regulären Ausdrücken gibt es nun zusätzlich noch einen Operator zur parallelen Komposition von zwei Termen. Dieser macht nur im Kontext von BPOs Sinn, welche nebenläufiges, also paralleles, Verhalten darstellen können.
Die Synthese von Petrinetzen aus klassischen regulären Ausdrücken über einem endlichen Alphabet von Transitionen wurde schon in einer Reihe von Arbeiten diskutiert [63, 65, 13, 18]. Die Vorgehensweise in diesem Abschnitt hat Ähnlichkeiten zu diesen Ansätzen, insbesondere zu [63]. Die Idee der Synthese aus BPO-Termen wurde schon in [153] aufgeworfen, allerdings noch ohne etwaige technische Ausführungen.
BPO-Terme werden, wie besprochen, durch Iteration, alternative Komposition bzw. Vereinigung, parallele Komposition und sequentielle Komposition aus einem endlichen Alphabet von BPOs $\mathcal{A}$ konstruiert. Für $A \in \mathcal{A}$ schreiben wir $A = (V_A, <_A, l_A)$.

Definition 5.1.1 (BPO-Term)

Definition 5.1.1 (BPO-Term)
Die Menge der BPO-Terme über einer endlichen Menge von BPOs $\mathcal{A}$ ist folgendermaßen induktiv definiert: Die Buchstaben $A \in \mathcal{A}$ und λ sind BPO-Terme. Seien α_1 und α_2 BPO-Terme. Dann sind $\alpha = \alpha_1; \alpha_2$ (sequentielle Komposition), $\alpha = \alpha_1 + \alpha_2$ (Vereinigung), $\alpha = (\alpha_1)^*$ (Iteration) und $\alpha = \alpha_1 \parallel \alpha_2$ (parallele Komposition) BPO-Terme.

Falls jede BPO aus $\mathcal{A}$ nur aus einem Knoten besteht, so liegt einem BPO-Term ein Alphabet von Transitionen zugrunde. In diesem, in der Literatur schon betrachteten Fall definiert ein Term eine sog. „series rational sp-language“ [151, 152]. Für einen allgemeinen BPO-Term definieren wir nun auf ähnliche Weise induktiv eine zugeordnete möglicherweise unendliche partielle Sprache $\mathcal{L}(\alpha)$.

Definition 5.1.2 (Partielle Sprache eines BPO-Terms)

Definition 5.1.2 (Partielle Sprache eines BPO-Terms)
Wir definieren $\mathcal{L}(\lambda) = \{\lambda\}$ *und* $\mathcal{L}(A) = \{A\}$. *Weiter definieren wir induktiv für BPO-Terme* α_1 *und* α_2*:*
$\mathcal{L}(\alpha_1 + \alpha_2) = \mathcal{L}(\alpha_1) \cup \mathcal{L}(\alpha_2)$
$\mathcal{L}(\alpha_1 ; \alpha_2) = \{A_1 ; A_2 \mid A_1 \in \mathcal{L}(\alpha_1),\, A_2 \in \mathcal{L}(\alpha_2)\}$
$\mathcal{L}((\alpha_1)^*) = \{A_1 ; \ldots ; A_n \mid A_1, \ldots, A_n \in \mathcal{L}(\alpha_1)\} \cup \{\lambda\}$
$\mathcal{L}(\alpha_1 \parallel \alpha_2) = \{A_1 \parallel A_2 \mid A_1 \in \mathcal{L}(\alpha_1),\, A_2 \in \mathcal{L}(\alpha_2)\}$

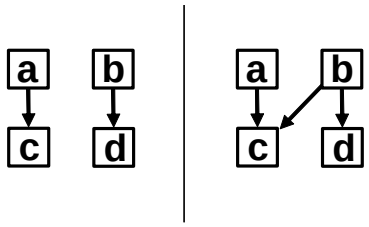

Abbildung 83: Zwei BPOs.

Zur Vereinfachung der Abbildungen betrachten wir in den Beispielen nur BPO-Terme, welche aus BPOs mit nur einem Knoten zusammengesetzt sind. Hierbei bezeichnen wir mit a die BPO, welche nur einen mit a beschrifteten Knoten enthält.

Beispiel

Beispiel: *Wir beginnen mit einem endlichen Beispiel. Die in Abbildung 83 links dargestellte BPO ist das einzige Element der partiellen Sprache des BPO-Terms* $(a;c)\|(b;d)$. *Die BPO in Abbildung 83 rechts weist eine sog. N-Form auf. Diese lässt sich nicht durch einen Term über den BPOs* a, b, c *und* d *darstellen. Allerdings kann diese BPO direkt als Baustein für einen BPO-Term verwendet werden.*
Abbildung 84 illustriert links die unendliche partielle Sprache des BPO-Terms $a\|b^*$. *Ein Netz mit einem entsprechenden Ablaufverhalten ist in der Abbildung rechts dargestellt (bei Betrachtung des Präfix- und Sequentialisierungsabschlusses). Es lassen sich aber nicht alle unendlichen partiellen Sprachen durch BPO-Terme darstellen. Dies wird im nächsten Unterabschnitt ausführlich diskutiert. Beispielsweise kann die in Abbildung 85 links illustrierte unendliche partielle Sprache nicht durch einen BPO-Term repräsentiert werden (auch wenn wir nur denselben Präfix- und Sequentialisierungsabschluss erhalten möchten), da es durch sequentielle Komposition und Iteration nicht möglich ist, eine BPO nur an einen Teil einer anderen BPO anzuhängen. Dabei ist zu beachten, dass die partielle Sprache aus Abbildung 85 der Ablaufsemantik des in der Abbildung rechts dargestellten S/T-Netzes entspricht, d.h. mit BPO-Termen lassen sich auch nicht alle Ablaufsemantiken von S/T-Netzen repräsentieren. Selbiges gilt auch für die Ablaufsemantiken von elementaren Netzen und sogar einssicheren konfliktfreien Netzen.*

Mit Definition 5.1.2 lässt sich nun das Syntheseproblem des letzten Kapitels für den Fall von BPO-Termen umformen. In diesem Unterabschnitt betrachten wir das folgende Problem.

Problemspezifikation 5.1.1 (Syntheseproblem für BPO-Terme)

Problemspezifikation 5.1.1 (Syntheseproblem für BPO-Terme)
Eingabe: *Ein BPO-Term* α.
Ausgabe: *Ein markiertes S/T-Netz* (N, m_0) *mit* $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L}(\alpha))$, *falls ein solches Lösungsnetz existiert, und eine „*`notexists`*“-Meldung sonst.*

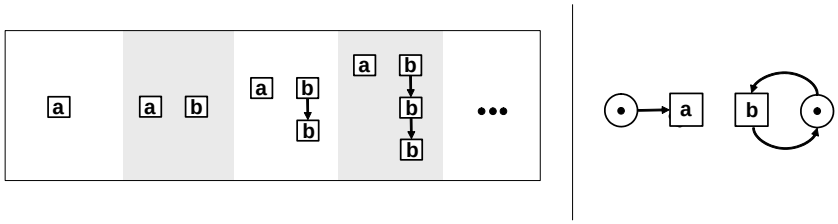

Abbildung 84: Unendliche partielle Sprache eines BPO-Terms.

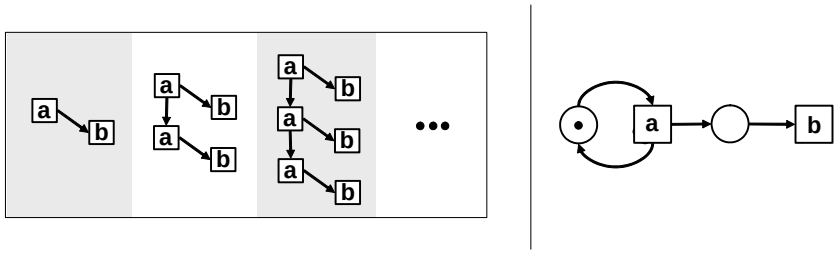

Abbildung 85: Unendliche partielle Sprache, welche nicht durch einen BPO-Term repräsentiert werden kann.

Um das Syntheseproblem zu lösen, gehen wir im Prinzip wie im letzten Kapitel vor, d.h. wir wenden die Ideen der Regionentheorie an. Allerdings ergeben sich für unendliche partielle Sprachen im Rahmen der im letzten Kapitel diskutierten linear algebraischen Charakterisierungen von Regionen jeweils unendlich viele Ungleichungen. Im Falle von Markenfluss-Regionen ergeben sich auch unendlich viele Variablen. Eine derartige direkte Übertragung des Vorgehens des letzten Kapitels ist also nicht praktikabel. Stattdessen sollen hier Regionen direkt für einen BPO-Term und nicht für die durch den Term definierte unendliche partielle Sprache betrachtet werden. Diese Regionen sollen natürlich dennoch genau die zulässigen Stellentripel der partiellen Sprache charakterisieren. In anderen Worten bedeutet dies, dass die Menge der Regionen eines BPO-Terms dieselbe Menge an Stellen definieren soll wie die Menge der Regionen der vom Term definierten partiellen Sprache. Hierbei können wir auf Regionen jedweden Typs zurückgreifen. Es ist zu beachten, dass die Regionendefinitionen des letzten Abschnittes auch tatsächlich für unendliche partielle Sprachen formuliert wurden.

Beispiel

BEISPIEL: *Wir betrachten im Weiteren den Term* $\mathfrak{b} + (\mathfrak{a}; (\mathfrak{a} \parallel \mathfrak{b})^\star)$. *Die partielle Sprache dieses Terms ist in Teil (a) von Abbildung 86 dargestellt. Es lässt sich leicht nachprüfen, dass all diese BPOs bzgl. der in Teil (b) der Abbildung dargestellten Stelle aktiviert sind, d.h. die Stelle ist bzgl.* $\mathcal{L}(\mathfrak{b} + (\mathfrak{a}; (\mathfrak{a} \parallel \mathfrak{b})^\star))$ *zulässig. Die dritte abgebildete BPO ist aber nicht bzgl. der in Teil (c) dargestellten Stelle aktiviert, d.h. diese Stelle ist nicht zulässig.*

Das Ziel ist also, zuerst eine entsprechende Definition von Regionen für BPO-Terme zu entwickeln und darauf aufbauend dann eine endliche linear algebraische Charakterisierung solcher Regionen zu finden. Aus dieser können dann wiederum Verfahren zur Berechnung einer endlichen Repräsentation des gesättigt zulässigen Netzes der partiellen Sprache des Terms abgeleitet werden. Wir führen im Folgenden beispielhaft Markenfluss-Regionen für BPO-Terme ein und zeigen, dass die Menge der Markenfluss-Regionen eines Terms dieselbe Menge an Stel-

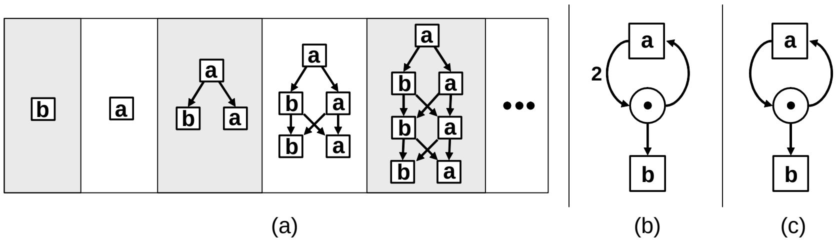

Abbildung 86: Eine partielle Sprache, eine zulässige und eine nicht-zulässige Stelle.

len definiert wie die Menge der Markenfluss-Regionen der partiellen Sprache des Terms.

Beispiel

Beispiel: *Eine Markenfluss-Region der unendlichen partiellen Sprache aus Teil (a) von Abbildung 86 ist in Abbildung 87 illustriert. Zur besseren Lesbarkeit stellen wir in diesem Abschnitt nur die echt positiven Markenflüsse dar (insbesondere werden Nicht-Gerüstkanten mit einem Markenfluss von Null weggelassen). Die zulässige Stelle aus Teil (b) von Abbildung 86 korrespondiert zu dieser Region.*

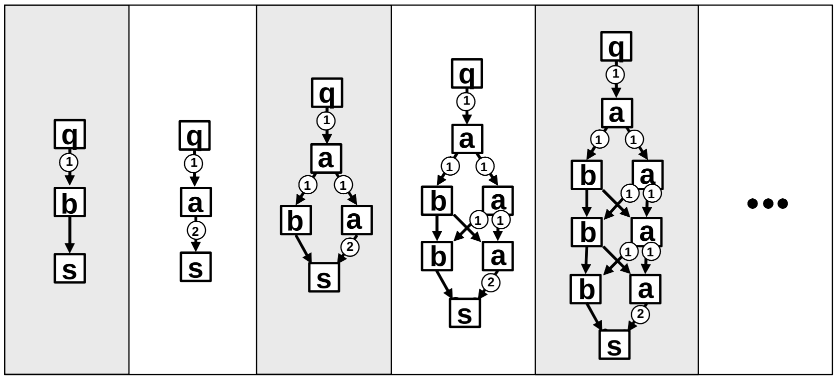

Abbildung 87: Markenfluss-Region.

Wir entwickeln eine Technik, um die Markenfluss-Regionen einer möglicherweise unendlichen partiellen Sprache $\mathcal{L}(\alpha)$ mithilfe von Markenfluss-Regionen einer endlichen partiellen Sprache darzustellen. Die endliche partielle Sprache muss hierzu die Sprache $\mathcal{L}(\alpha)$ geeignet repräsentieren. Das Problem ist, dass $\mathcal{L}(\alpha)$ aufgrund des Iterationsoperators unendlich viele BPOs enthalten kann. Eine in einem markierten Netz aktivierte BPO A kann dann und nur dann beliebig häufig hintereinander schalten, wenn sie insgesamt in jeder Stelle höchstens so viele Marken konsumiert wie sie produziert. Genau dann reduziert das Schalten von A die Anfangsmarkierung nicht. Damit ist A nach jedem Schalten von A wiederum aktiviert. Dies ist also eine notwendige und hinreichende Bedingung dafür, dass A unendlich oft schalten kann. Falls nun A in einer Markierung m iteriert schalten kann, so kann eine andere BPO B genau dann nach dem Schalten von A^n für jedes $n \in \mathbb{N}$ schalten, wenn B in m schalten kann. Dies liegt daran, dass in diesem Fall das Schalten von A die Anzahl der Marken in jeder Stelle nicht reduziert. Entsprechend diesen Überlegungen lässt sich die möglicherweise unendliche Menge $\mathcal{L}(\alpha)$ durch zwei endliche Mengen von BPOs $R(\alpha)$ und

$I(\alpha)$ darstellen. Die Menge $R(\alpha)$ heißt hierbei Repräsentationsmenge. Die Menge $I(\alpha)$ heißt Iterationsmenge. Diese Mengen können dann zur Definition von Regionen von α verwendet werden.
Die Menge $R(\alpha)$ beinhaltet grob gesagt alle BPOs aus $\mathcal{L}(\alpha)$, wobei Iterationen von α vernachlässigt werden. Die Menge $I(\alpha)$ beinhaltet dann die BPOs, welche grob gesagt zu iterierbaren Subtermen von α gehören. Beide Mengen $R(\alpha)$ und $I(\alpha)$ sind induktiv ausgehend von dem BPO-Term α definiert. Damit ergibt sich das folgende Vorgehen zur Definition einer Region von α. Um sicherzustellen, dass alle BPOs aus $R(\alpha)$ auch bzgl. der durch die Region des Terms definierten Stelle aktiviert sind, setzen wir voraus, dass eine Region von α eine Region von $R(\alpha)$ ist. Dann muss nur noch garantiert werden, dass für bestimmte BPOs ein iteriertes Schalten möglich ist. Nun ist $I(\alpha)$ derart definiert, dass es genau diese BPOs enthält. Wir fordern also, dass alle BPOs aus $I(\alpha)$ in der von der Region definierten Stelle mindestens so viele Marken produzieren wie sie konsumieren. Insgesamt stellt dies sicher, dass die von der Region von α definierte Stelle zulässig ist.

Definition 5.1.3 (Repräsentations- und Iterationsmenge)

Definition 5.1.3 (Repräsentations- und Iterationsmenge)
Die Repräsentationsmenge $R(\alpha)$ und die Iterationsmenge $I(\alpha)$ einer partiellen Sprache $\mathcal{L}(\alpha)$ sind für BPO-Terme α_1 und α_2 folgendermaßen induktiv definiert:

- $R(\lambda) = \{\lambda\}$
- $R(A) = \{A\}$ *für* $A \in \mathcal{A}$
- $R(\alpha_1 + \alpha_2) = R(\alpha_1) \cup R(\alpha_2)$
- $R(\alpha_1; \alpha_2) = \{A_1; A_2 \mid A_1 \in R(\alpha_1),\ A_2 \in R(\alpha_2)\}$
- $R((\alpha_1)^*) = R(\alpha_1) \cup \{\lambda\}$
- $R(\alpha_1 \parallel \alpha_2) = \{A_1 \parallel A_2 \mid A_1 \in R(\alpha_1),\ A_2 \in R(\alpha_2)\}$
- $I(\lambda) = \emptyset$
- $I(A) = \emptyset$ *für* $A \in \mathcal{A}$
- $I(\alpha_1 + \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$
- $I(\alpha_1; \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$
- $I((\alpha_1)^*) = I(\alpha_1) \cup R(\alpha_1)$
- $I(\alpha_1 \parallel \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$

Damit definieren wir nun eine Markenfluss-Region eines Terms α. Eine solche Region ist eine Markenfluss-Region von $R(\alpha)$, welche die Forderung, dass jede BPO aus $I(\alpha)$ in der von der Region definierten Stelle mindestens so viele Marken produziert wie sie konsumiert, erfüllt. Diese Forderung bedeutet, dass die Summe der Marken, welche von Knoten der BPO produziert werden, die Summe der Marken, welche von den Knoten konsumiert werden, übersteigt. Sei also eine BPO $bpo = (V, <, l)$ und eine Stelle p eines Netzes (N, m_0), $N = (P, T, W)$, gegeben. Dann wird die Differenz

$$Prod(bpo, p) := \sum_{t \in |V|_l} |V|_l(t)(W(t,p) - W(p,t))$$

für die von der Region definierten Stelle p betrachtet.

Definition 5.1.4 (Markenfluss-Region eines BPO-Terms)

Definition 5.1.4 (Markenfluss-Region eines BPO-Terms)
Eine Markenfluss-Region s eines BPO-Terms α ist eine Markenfluss-Region von $R(\alpha)$, welche für alle BPOs $bpo \in I(\alpha)$:

$$(IT) \qquad Prod(bpo, p_{\overrightarrow{s}}) \geqslant 0.$$

erfüllt.
Die Definition von $\overrightarrow{s}$ ergibt sich hierbei dadurch, dass s eine Markenfluss-Region von $R(\alpha)$ ist. Wir bezeichnen $\overrightarrow{s}$ als das zu s korrespondierende Stellentripel über T.

Beispiel

Beispiel: *Abbildung 88 zeigt die Repräsentationsmenge $R(b + (a;(a \parallel b)^\star))$ und die Iterationsmenge $I(b + (a;(a \parallel b)^\star))$ der partiellen Sprache $\mathcal{L}(b + (a;(a \parallel b)^\star))$ zusammen mit einer Markenfluss-Region s des BPO-Terms $b + (a;(a \parallel b)^\star)$. Für die BPO $bpo = (V, <, l) \in I(\alpha)$ ist der Wert $\sum_{t \in |V|_l} |V|_l(t) W_{\overrightarrow{s}}(t, p_{\overrightarrow{s}}) = 2$ ($\sum_{t \in |V|_l} |V|_l(t) W_{\overrightarrow{s}}(p_{\overrightarrow{s}}, t) = 2$) an einen großen ausgehenden (eingehenden) Pfeil annotiert. Es gilt $Prod(bpo, p_{\overrightarrow{s}}) = \sum_{t \in |V|_l} |V|_l(t)(W_{\overrightarrow{s}}(t, p_{\overrightarrow{s}}) - W_{\overrightarrow{s}}(p_{\overrightarrow{s}}, t)) = 0$. Zur dargestellten Region s korrespondiert die zulässige Stelle aus Teil (b) von Abbildung 86.*
Dieses Beispiel illustriert insbesondere das Prinzip der Berechnung der Repräsentations- und Iterationsmenge. Die Repräsentationsmenge wird gebildet, indem die parallelen und sequentiellen Kompositionsoperatoren in die BPO-Struktur integriert werden. Dabei werden Alternativen durch verschiedene BPOs ausgedrückt und iterierte Teile werden durch zwei verschiedene BPOs repräsentiert, wobei eine den iterierte Teil genau einmal enthält und bei einer der Teil weggelassen ist. Die Iterationsmenge ist in diesem Beispiel einfach durch die zu $a \| b$ definierte BPO gegeben. Dadurch wird angezeigt, dass diese BPO entsprechend wiederholt stattfinden kann.

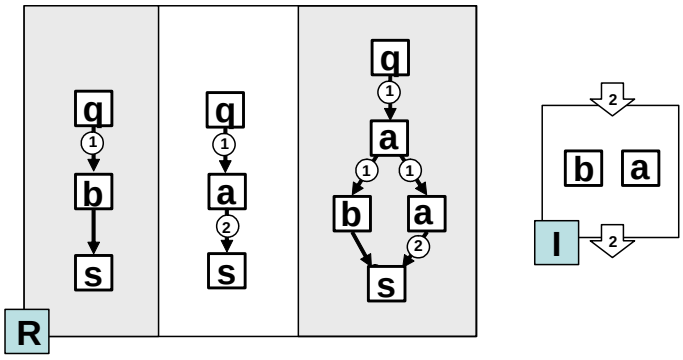

Abbildung 88: Markenfluss-Region eines BPO-Terms.

Im Weiteren zeigen wir, dass ein Stellentripel genau dann zulässig bzgl. $\mathcal{L}(\alpha)$ ist, wenn es zu einer Markenfluss-Region von α korrespondiert. Dazu zeigen wir, dass die Menge der von Regionen von α definierten Stellen mit der Menge der von Regionen von $\mathcal{L}(\alpha)$ definierten Stellen übereinstimmt. Wir beginnen mit einigen Notationen für globale Markenfluss-Funktionen. Sei r eine globale Markenfluss-Funktion von $\mathcal{L}$ und $bpo \in \mathcal{L}$, dann schreiben wir $Final(bpo, r) = Zu_r(s_{bpo})$ und $Init(bpo, r) = Ab_r(q_{bpo})$. Falls r eine Markenfluss-Region von $\mathcal{L}$ ist, schreiben wir auch $Init(p_{\overrightarrow{r}}) = Init(bpo, r)$ und $Final(bpo, p_{\overrightarrow{r}}) = Final(bpo, r)$.

Zum Beweis des gewünschten Zusammenhangs benötigen wir vier Lemmata, welche wir im folgenden herleiten. Wenn eine Markenfluss-Region r einer BPO bpo betrachtet wird, so lässt sich $\mathrm{Prod}(\mathrm{bpo}, p_{\overrightarrow{r}})$, wie im nächsten Lemma dargelegt, berechnen. Es ist hierbei zu beachten, dass $\mathrm{Prod}(\mathrm{bpo}, p)$ unabhängig von der Anfangsmarkierung von p ist.

LEMMA 5.1.1 *Lemma 5.1.1*

Sei $\mathrm{bpo} = (V, <, l)$ *eine BPO und* r *eine Markenfluss-Region von* $\{\mathrm{bpo}\}$. *Dann gilt* $\mathrm{Prod}(\mathrm{bpo}, p_{\overrightarrow{r}}) = \mathrm{Final}(\mathrm{bpo}, p_{\overrightarrow{r}}) - \mathrm{Init}(p_{\overrightarrow{r}})$.

BEWEIS: Es gilt $\sum_{v \in V^*} \mathrm{Ab}_r(v) = \sum_{e \in <^*} r(e) = \sum_{v \in V^*} \mathrm{Zu}_r(v)$. Dann folgt $\sum_{v \in V} \mathrm{Ab}_r(v) + \mathrm{Ab}_r(q_{\mathrm{bpo}}) + \mathrm{Ab}_r(s_{\mathrm{bpo}}) = \sum_{v \in V} \mathrm{Zu}_r(v) + \mathrm{Zu}_r(q_{\mathrm{bpo}}) + \mathrm{Zu}_r(s_{\mathrm{bpo}})$. Da $\mathrm{Ab}_r(s_{\mathrm{bpo}}) = 0 = \mathrm{Zu}_r(q_{\mathrm{bpo}})$, ergibt sich $\mathrm{Zu}_r(s_{\mathrm{bpo}}) - \mathrm{Ab}_r(q_{\mathrm{bpo}}) = \sum_{v \in V}(\mathrm{Ab}_r(v) - \mathrm{Zu}_r(v)) = \sum_{t \in |V|_l} |V|_l(t) \cdot (W_{\overrightarrow{r}}(t, p_{\overrightarrow{r}}) - W_{\overrightarrow{r}}(p_{\overrightarrow{r}}, t))$. □

Das folgende Lemma macht die Beziehung zwischen $\mathcal{L}(\alpha)$ und $R(\alpha)$ deutlich.

LEMMA 5.1.2 *Lemma 5.1.2*

Sei ein BPO-Term α *und eine Markenfluss-Region* s *von* α *gegeben. Dann gibt es für jede BPO* $\mathrm{bpo} \in \mathcal{L}(\alpha)$ *eine BPO* $\mathrm{bpo}^R \in R(\alpha)$, *so dass* $\mathrm{Prod}(\mathrm{bpo}, p_{\overrightarrow{s}}) \geqslant \mathrm{Prod}(\mathrm{bpo}^R, p_{\overrightarrow{s}})$.

BEWEIS: Wir führen den Beweis induktiv. Für $A \in \mathcal{A}$ gilt $\mathcal{L}(A) = \{A\} = R(A)$, d.h. wir können $A^R = A$ setzen. Angenommen die Aussage gilt für α_1 und α_2, dann ergibt sich der Induktionsschritt folgendermaßen:
ad $(+)$: Für $\mathrm{bpo} \in \mathcal{L}(\alpha_1 + \alpha_2)$ gilt $\mathrm{bpo} \in \mathcal{L}(\alpha_1)$ oder $\mathrm{bpo} \in \mathcal{L}(\alpha_2)$. Sei o.B.d.A. $\mathrm{bpo} \in \mathcal{L}(\alpha_1)$. Nach Induktionsvoraussetzung gibt es $\mathrm{bpo}^R \in R(\alpha_1) \subseteq R(\alpha_1 + \alpha_2)$, so dass $\mathrm{Prod}(\mathrm{bpo}, p_{\overrightarrow{s}}) \geqslant \mathrm{Prod}(\mathrm{bpo}^R, p_{\overrightarrow{s}})$.
ad $(;)$: Für $\mathrm{bpo} \in \mathcal{L}(\alpha_1;\alpha_2)$ gibt es $\mathrm{bpo}_1 \in \mathcal{L}(\alpha_1)$ und $\mathrm{bpo}_2 \in \mathcal{L}(\alpha_2)$, so dass $\mathrm{bpo}_1;\mathrm{bpo}_2 = \mathrm{bpo}$. Nach Induktionsvoraussetzung gibt es $\mathrm{bpo}_1^R \in R(\alpha_1)$, $\mathrm{bpo}_2^R \in R(\alpha_2)$, so dass $\mathrm{Prod}(\mathrm{bpo}_i^R, p_{\overrightarrow{s}}) \leqslant \mathrm{Prod}(\mathrm{bpo}_i, p_{\overrightarrow{s}})$, $i = 1, 2$. Es folgt $\mathrm{bpo}_1^R;\mathrm{bpo}_2^R \in R(\alpha_1;\alpha_2)$ mit $\mathrm{Prod}(\mathrm{bpo}_1^R;\mathrm{bpo}_2^R, p_{\overrightarrow{s}}) = \mathrm{Prod}(\mathrm{bpo}_1^R, p_{\overrightarrow{s}}) + \mathrm{Prod}(\mathrm{bpo}_2^R, p_{\overrightarrow{s}}) \leqslant \mathrm{Prod}(\mathrm{bpo}_1, p_{\overrightarrow{s}}) + \mathrm{Prod}(\mathrm{bpo}_2, p_{\overrightarrow{s}}) = \mathrm{Prod}(\mathrm{bpo}_1;\mathrm{bpo}_2, p_{\overrightarrow{s}})$.
ad $(\|)$: Für $\mathrm{bpo} \in \mathcal{L}(\alpha_1 \parallel \alpha_2)$ gibt es $\mathrm{bpo}_1 \in \mathcal{L}(\alpha_1)$ und $\mathrm{bpo}_2 \in \mathcal{L}(\alpha_2)$, so dass $\mathrm{bpo}_1 \parallel \mathrm{bpo}_2 = \mathrm{bpo}$. Nach Induktionsvoraussetzung gibt es $\mathrm{bpo}_1^R \in R(\alpha_1)$, $\mathrm{bpo}_2^R \in R(\alpha_2)$, so dass $\mathrm{Prod}(\mathrm{bpo}_i^R, p_{\overrightarrow{s}}) \leqslant \mathrm{Prod}(\mathrm{bpo}_i, p_{\overrightarrow{s}})$, $i = 1, 2$. Es folgt $\mathrm{bpo}_1^R \parallel \mathrm{bpo}_2^R \in R(\alpha_1 \parallel \alpha_2)$ mit $\mathrm{Prod}(\mathrm{bpo}_1^R \parallel \mathrm{bpo}_2^R, p_{\overrightarrow{s}}) = \mathrm{Prod}(\mathrm{bpo}_1^R, p_{\overrightarrow{s}}) + \mathrm{Prod}(\mathrm{bpo}_2^R, p_{\overrightarrow{s}}) \leqslant \mathrm{Prod}(\mathrm{bpo}_1, p_{\overrightarrow{s}}) + \mathrm{Prod}(\mathrm{bpo}_2, p_{\overrightarrow{s}}) = \mathrm{Prod}(\mathrm{bpo}_1 \parallel \mathrm{bpo}_2, p_{\overrightarrow{s}})$.
ad $(*)$: Für $\mathrm{bpo} \in \mathcal{L}(\alpha_1^*)$ gibt es $\mathrm{bpo}_1, \ldots, \mathrm{bpo}_n \in \mathcal{L}(\alpha_1)$, so dass $\mathrm{bpo}_1;\ldots;\mathrm{bpo}_n = \mathrm{bpo}$. Nach Induktionsvoraussetzung und (IT) gibt es für jede BPO bpo_i ($i \in \{1, \ldots, n\}$) eine BPO $\mathrm{bpo}_i^R \in R(\alpha_1) \subseteq I(\alpha_1^*)$ mit $\mathrm{Prod}(\mathrm{bpo}_i, p_{\overrightarrow{s}}) \geqslant \mathrm{Prod}(\mathrm{bpo}_i^R, p_{\overrightarrow{s}}) \geqslant 0$. Es folgt $\mathrm{bpo}_1^R \in R(\alpha_1^*)$ mit $\mathrm{Prod}(\mathrm{bpo}_1^R, p_{\overrightarrow{s}}) \leqslant \mathrm{Prod}(\mathrm{bpo}_1, p_{\overrightarrow{s}}) + \ldots + \mathrm{Prod}(\mathrm{bpo}_n, p_{\overrightarrow{s}}) = \mathrm{Prod}(\mathrm{bpo}_1;\ldots;\mathrm{bpo}_n, p_{\overrightarrow{s}})$. □

Die folgenden zwei Lemmata sind technischer Natur. Das erste zeigt, wie sich eine Markenfluss-Region bei einer sequentiellen Verknüpfung von zwei BPOs auf die zwei BPOs aufteilen lässt. Das zweite zeigt die umgekehrte Richtung, d.h. wie sich unter bestimmten Voraussetzungen zwei Markenfluss-Regionen auf zwei BPOs zu einer Markenfluss-Region auf der sequentiellen Verknüpfung der BPOs zusammenfügen lassen.

Lemma 5.1.3

LEMMA 5.1.3
Sei s *eine Markenfluss-Region von* $\{bpo_1; bpo_2\}$. *Dann gibt es eine Markenfluss-Region* $s|_{bpo_1}$ *von* $\{bpo_1\}$ *mit* $\overrightarrow{s|_{bpo_1}} = \overrightarrow{s}$ *und eine Markenfluss-Region* $s|_{bpo_2}$ *von* $\{bpo_2\}$, *so dass* $W_{\overrightarrow{s|_{bpo_2}}}(p_{\overrightarrow{s|_{bpo_2}}}, t) = W_{\overrightarrow{s}}(p_{\overrightarrow{s}}, t)$ *und* $W_{\overrightarrow{s|_{bpo_2}}}(t, p_{\overrightarrow{s|_{bpo_2}}}) = W_{\overrightarrow{s}}(t, p_{\overrightarrow{s}})$ *für jedes* $t \in T$ *und* $Init(p_{\overrightarrow{s|_{bpo_2}}}) = Init(p_{\overrightarrow{s}}) + Prod(bpo_1, p_{\overrightarrow{s}})$.

BEWEIS: Sei $bpo_1 = (V_1, <_1, l_1)$, $bpo_2 = (V_2, <_2, l_2)$ und $bpo = bpo_1; bpo_2$. Wir definieren zuerst $s|_{bpo_1}$ durch:

(i) $\forall e \in V_1 \times V_1 : s|_{bpo_1}(e) = s(e)$,

(ii) $\forall v \in V_1 : s|_{bpo_1}(q_{bpo_1}, v) = s(q_{bpo}, v)$,

(iii) $\forall v \in V_1 : s|_{bpo_1}(v, s_{bpo_1}) = \sum_{w \in V_2 \cup \{s_{bpo}\}} s(v, w)$,

(iv) $s|_{bpo_1}(q_{bpo_1}, s_{bpo_1}) = \sum_{w \in V_2 \cup \{s_{bpo}\}} s(q_{bpo}, w)$.

Da durch diese Konstruktion der Marken-Zufluss und der Marken-Abfluss von Knoten aus V_1 und von dem Quellenknoten nicht verändert werden, ist $s|_{bpo_1}$ eine Markenfluss-Region und $\overrightarrow{s}$ ist das korrespondierende Stellentripel. Nun definieren wir $s|_{bpo_2}$ durch:

(i) $\forall e \in V_2 \times V_2 : s|_{bpo_2}(e) = s(e)$,

(ii) $\forall v \in V_2 : s|_{bpo_2}(v, s_{bpo_2}) = s(v, s_{bpo})$,

(iii) $\forall v \in V_2 : s|_{bpo_2}(q_{bpo_2}, v) = \sum_{w \in V_1 \cup \{q_{bpo}\}} s(w, v)$,

(iv) $s|_{bpo_2}(q_{bpo_2}, s_{bpo_2}) = \sum_{w \in V_1 \cup \{q_{bpo}\}} s(w, s_{bpo})$.

Da durch diese Konstruktion der Marken-Zufluss und der Marken-Abfluss von Knoten aus V_2 und von dem Senkenknoten nicht verändert werden, ist $s|_{bpo_2}$ eine Markenfluss-Region und es gilt $W_{\overrightarrow{s|_{bpo_2}}}(p_{\overrightarrow{s|_{bpo_2}}}, t) = W_{\overrightarrow{s}}(p_{\overrightarrow{s}}, t)$ und $W_{\overrightarrow{s|_{bpo_2}}}(t, p_{\overrightarrow{s|_{bpo_2}}}) = W_{\overrightarrow{s}}(t, p_{\overrightarrow{s}})$ für jedes $t \in T$ und $Init(p_{\overrightarrow{s|_{bpo_2}}}) = Final(bpo_2, p_{\overrightarrow{s|_{bpo_2}}}) - Prod(bpo_2, p_{\overrightarrow{s|_{bpo_2}}}) = Final(bpo, p_{\overrightarrow{s}}) - (Prod(bpo, p_{\overrightarrow{s}}) - Prod(bpo_1, p_{\overrightarrow{s}})) = Init(p_{\overrightarrow{s}}) + Prod(bpo_1, p_{\overrightarrow{s}})$. □

Lemma 5.1.4

LEMMA 5.1.4
Seien zwei BPOs bpo_1 *und* bpo_2 *gegeben. Sei* r_1 *eine Markenfluss-Region von* $\{bpo_1\}$ *und* r_2 *eine Markenfluss-Region von* $\{bpo_2\}$, *so dass* $W_{\overrightarrow{r_2}}(p_{\overrightarrow{r_2}}, t) = W_{\overrightarrow{r_1}}(p_{\overrightarrow{r_1}}, t)$ *und* $W_{\overrightarrow{r_2}}(t, p_{\overrightarrow{r_2}}) = W_{\overrightarrow{r_1}}(t, p_{\overrightarrow{r_1}})$ *für jedes* $t \in T$ *und* $Final(bpo_1, p_{\overrightarrow{r_1}}) \geqslant Init(p_{\overrightarrow{r_2}})$. *Dann gibt es eine Markenfluss-Region* r *von* $\{bpo_1; bpo_2\}$ *mit* $\overrightarrow{r} = \overrightarrow{r_1}$.

BEWEIS: Sei $bpo_1 = (V_1, <_1, l_1)$, $bpo_2 = (V_2, <_2, l_2)$ und $bpo = bpo_1; bpo_2$. Wir definieren r durch:

(i) $\forall e \in V_1 \times V_1 : r(e) = r_1(e)$,

(ii) $\forall e \in V_2 \times V_2 : r(e) = r_2(e)$,

(iii) $\forall v \in V_1 : r(q_{bpo}, v) = r_1(q_{bpo_1}, v)$,

(iv) $\forall v \in V_2 : r(v, s_{bpo}) = r_2(v, q_{bpo_2})$.

Dann sind bzgl. r der Marken-Zufluss von Knoten aus V_1 und der Marken-Abfluss von Knoten aus V_2 konsistent zur Behauptung. Es muss noch $\mathtt{Final}(bpo_1, r_1)$ geeignet auf die Kanten $(\{q_{bpo}\} \cup V_1\}) \times (\{V_2 \cup \{s_{bpo}\})$ verteilt werden, so dass r eine Markenfluss-Region ist. Dazu muss es nur möglich sein, dass der Marken-Zufluss von jedem Knoten aus $(\{V_2 \cup \{s_{bpo}\})$ entsprechend dessen Beschriftung bedient werden kann. Dies ist offensichtlich der Fall, da $\mathtt{Final}(bpo_1, p_{\overrightarrow{r_1}}) \geqslant \mathtt{Init}(p_{\overrightarrow{r_2}})$ gilt. Per Konstruktion ist r dann also eine Markenfluss-Region mit korrespondierendem Stellentripel $\overrightarrow{r_1}$. □

Der folgende Satz zeigt nun die Korrespondenz zwischen Markenfluss-Regionen von BPO-Termen und Markenfluss-Regionen von zu BPO-Termen gehörigen partiellen Sprachen. Es wird gezeigt, dass die Menge der zu einer Markenfluss-Region eines BPO-Terms korrespondierenden Stellentripel mit der Menge der zu einer Markenfluss-Region der partiellen Sprache des Terms korrespondierenden Stellentripel übereinstimmt. Mit Satz 4.3.5 folgt somit, dass die Markenfluss-Regionen eines BPO-Terms genau die bzgl. der zugehörigen partiellen Sprache zulässigen Stellentripel definieren.

BEISPIEL: *Wir haben schon gezeigt, dass die in Abbildung 88 dargestellte Markenfluss-Region eines BPO-Terms dasselbe Stellentripel definiert wie die in Abbildung 87 illustrierte Markenfluss-Region der entsprechenden partiellen Sprache. Beide Regionen definieren die in Abbildung 86, Teil (b), dargestellte Stelle.* *Beispiel*

SATZ 5.1.1 *Satz 5.1.1*
Sei α ein BPO-Term. Es gilt:
(i) Sei s eine Markenfluss-Region von α. Dann gibt es eine Markenfluss-Region r von $\mathcal{L}(\alpha)$ mit $\overrightarrow{r} = \overrightarrow{s}$.
(ii) Sei r eine Markenfluss-Region von $\mathcal{L}(\alpha)$. Dann gibt es eine Markenfluss-Region s von α mit $\overrightarrow{s} = \overrightarrow{r}$.

BEWEIS: **(i)**: Wir beweisen die erste Aussage induktiv. Für $A \in \mathcal{A}$ gilt $R(A) = \{A\} = \mathcal{L}(A)$. Daher können wir $r = s$ wählen. Angenommen die Aussage gilt für α_1 und α_2, dann ergibt sich der Induktionsschritt folgendermaßen:
ad $(;)$: Sei s eine Markenfluss-Region von $\alpha_1;\alpha_2$. Zuerst konstruieren wir aus s eine Markenfluss-Region s_1 von α_1 und eine Markenfluss-Region s_2 von α_2. Für die Konstruktion von s_1 fixieren wir $bpo_2 \in R(\alpha_2)$. Für jede BPO $bpo_1 \in R(\alpha_1)$ gilt $bpo_1;bpo_2 \in R(\alpha_1;\alpha_2)$. In Lemma 5.1.3 wird beschrieben, wie sich eine Markenfluss-Region $s|_{bpo_1}$ von $\{bpo_1\}$ mit korrespondierendem Stellentripel $\overrightarrow{s}$ konstruieren lässt. Für jede BPO $bpo_1 \in R(\alpha_1)$ definieren wir $s_1 = s|_{bpo_1}$. Da $I(\alpha_1) \subseteq I(\alpha_1;\alpha_2)$, erfüllt s_1 die Eigenschaft (IT). Somit ist s_1 per Konstruktion eine Markenfluss-Region von α_1. Zur Konstruktion von s_2 fixieren wir $bpo_1^{min} \in R(\alpha_1)$, so dass für alle BPOs $bpo \in R(\alpha_1)$ die Beziehung $\mathtt{Prod}(bpo_1^{min}, p_{\overrightarrow{s}}) \leqslant \mathtt{Prod}(bpo, p_{\overrightarrow{s}})$ gilt. Für jede BPO $bpo_2 \in R(\alpha_2)$ gilt $bpo_1^{min};bpo_2 \in R(\alpha_1;\alpha_2)$. In Lemma 5.1.3 wird beschrieben, wie sich eine Markenfluss-Region $s|_{bpo_2}$ von $\{bpo_2\}$ mit $W^{\longrightarrow}_{s|_{bpo_2}}(p_{\longrightarrow}, t) = W_{\overrightarrow{s}}(p_{\overrightarrow{s}}, t)$ und $W^{\longrightarrow}_{s|_{bpo_2}}(t, p_{\longrightarrow}) = W_{\overrightarrow{s}}(t, p_{\overrightarrow{s}})$ für jedes $t \in T$ und $\mathtt{Init}(p_{\longrightarrow}) = \mathtt{Init}(p_{\overrightarrow{s}}) + \mathtt{Prod}(bpo_1^{min}, p_{\overrightarrow{s}})$ konstruieren lässt. Wir definieren $s_2 = s|_{bpo_2}$ für jede BPO $bpo_2 \in R(\alpha_2)$. Da $I(\alpha_2) \subseteq I(\alpha_1;\alpha_2)$, erfüllt s_2 die Eigenschaft (IT). Somit ist s_2 per Konstruktion eine

Markenfluss-Region von α_2. Nach Induktionsvoraussetzung gibt es eine Markenfluss-Region r_1 von $\mathcal{L}(\alpha_1)$ mit $\overrightarrow{r_1} = \overrightarrow{s}$ und eine Markenfluss-Region r_2 von $\mathcal{L}(\alpha_2)$ mit $\overrightarrow{r_2} = \overrightarrow{s|_{bpo_2}}$. Für $bpo \in \mathcal{L}(\alpha_1;\alpha_2)$ gibt es $bpo_1^L \in \mathcal{L}(\alpha_1)$ und $bpo_2^L \in \mathcal{L}(\alpha_2)$, so dass $bpo_1^L;bpo_2^L = bpo$. Es gilt $Final(bpo_1^L, p_{\overrightarrow{s}}) = Init(p_{\overrightarrow{s}}) + Prod(bpo_1^L, p_{\overrightarrow{s}}) \geqslant Init(p_{\overrightarrow{s}}) + Prod(bpo_1^{min}, p_{\overrightarrow{s}}) = Init(p_{\overrightarrow{r_2}})$. Somit sind die Voraussetzungen für Lemma 5.1.4 erfüllt und wir können eine Markenfluss-Region $r|_{bpo}$ von $\{bpo\}$ mit korrespondierendem Stellentripel $\overrightarrow{s}$ konstruieren. Damit lässt sich die gesuchte Markenfluss-Region durch $r = r|_{bpo}$ für jede $bpo \in \mathcal{L}(\alpha_1;\alpha_2)$ definieren.
ad $(+)$: Sei s eine Markenfluss-Region von $\alpha_1 + \alpha_2$. Da $R(\alpha_1 + \alpha_2) = R(\alpha_1) \cup R(\alpha_2)$ und $I(\alpha_1 + \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$ gilt, erhalten wir direkt Markenfluss-Regionen s_1 von α_1 und s_2 von α_2 jeweils mit korrespondierendem Stellentripel $\overrightarrow{s}$, indem wir einfach s nur auf $R(\alpha_1)$ bzw. $R(\alpha_2)$ betrachten. Nach Induktionsvoraussetzung gibt es eine Markenfluss-Region r_1 von $\mathcal{L}(\alpha_1)$ und eine Markenfluss-Region r_2 von $\mathcal{L}(\alpha_2)$ jeweils mit korrespondierendem Stellentripel $\overrightarrow{s}$. Für $bpo \in \mathcal{L}(\alpha_1 + \alpha_2)$ gilt $bpo \in \mathcal{L}(\alpha_1)$ oder $bpo \in \mathcal{L}(\alpha_2)$. Daher wird durch $r_1 \cup r_2$ eine Markenfluss-Region auf $\mathcal{L}(\alpha_1 + \alpha_2)$ mit korrespondierendem Stellentripel $\overrightarrow{s}$ festgelegt.
ad $(\|)$: Hier ist eine ähnliche Argumentation wie im Falle von ; möglich. Aus einer Markenfluss-Region s von $\alpha_1 \parallel \alpha_2$ lässt sich eine geeignete Markenfluss-Region s_1 von α_1 und eine geeignete Markenfluss-Region s_2 von α_2 konstruieren. Die genaue Konstruktion ist wie im Falle von ; wieder technisch kompliziert. Auch hier müssen die Flüsse geeignet aufgeteilt und umverteilt werden. Dabei kann wesentlich verwendet werden, dass die Markenflüsse auf parallelen Teilen einer BPO gänzlich unabhängig voneinander sind, wodurch sich eine erste natürliche Aufteilung ergibt. Auf eine genaue Konstruktionsanleitung wird an dieser Stelle aber nicht noch einmal eingegangen. Nach der Aufteilung ergeben sich wiederum nach Induktionsvoraussetzung geeignete Markenfluss-Regionen r_1 von $\mathcal{L}(\alpha_1)$ und r_2 von $\mathcal{L}(\alpha_2)$, welche sich zur gesuchten Markenfluss-Region r zusammenfügen lassen.
ad $(*)$: Sei s eine Markenfluss-Region von α_1^*. Dann definiert s direkt eine Markenfluss-Region s_1 von α_1 mit korrespondierendem Stellentripel $\overrightarrow{s}$, da $R(\alpha_1) \subseteq R(\alpha_1^*)$ und $I(\alpha_1) \subseteq I(\alpha_1^*)$. Nach Induktionsvoraussetzung gibt es eine Markenfluss-Region r_1 von $\mathcal{L}(\alpha_1)$ mit korrespondierendem Stellentripel $\overrightarrow{s}$. Für $bpo \in \mathcal{L}(\alpha_1^*)$ gibt es BPOs $bpo_i \in \mathcal{L}(\alpha_1)$, $i = 1,\ldots,n$, so dass $bpo_1;\ldots;bpo_n = bpo$. Wie im Falle von (;) wenden wir Lemma 5.1.4 an, um eine Markenfluss-Region r_2 von $\{bpo_1;bpo_2\}$ mit korrespondierendem Stellentripel $\overrightarrow{s}$ zu erhalten. Hierzu müssen wir die Voraussetzungen von Lemma 5.1.4 nachweisen. Für $bpo_1 \in \mathcal{L}(\alpha_1)$ gibt es $bpo_1^R \in R(\alpha_1)$ mit $Prod(bpo_1, p_{\overrightarrow{s}}) \geqslant Prod(bpo_1^R, p_{\overrightarrow{s}})$ (Lemma 5.1.2). Da $R(\alpha_1) \subseteq I(\alpha_1^*)$ gilt, folgt $Prod(bpo_1^R, p_{\overrightarrow{s}}) \geqslant 0$. Somit gilt $Prod(bpo_1, p_{\overrightarrow{s}}) \geqslant 0$ und $Init(bpo_1, r_1) = Init(bpo_2, r_1)$, da r_1 eine Markenfluss-Region von $\mathcal{L}(\alpha_1)$ ist. Schließlich erhalten wir $Final(bpo_1, p_{\overrightarrow{s}}) = Init(p_{\overrightarrow{s}}) + Prod(bpo_1, p_{\overrightarrow{s}}) \geqslant Init(p_{\overrightarrow{s}})$. Induktiv ergibt sich eine Markenfluss-Region r_n von $\{bpo\}$ mit korrespondierendem Stellentripel $\overrightarrow{s}$.
(ii): Da $R(\alpha) \subseteq \mathcal{L}(\alpha)$, definiert r durch die Einschränkung auf $R(\alpha)$ unmittelbar eine Markenfluss-Region s von $R(\alpha)$ mit korrespondierendem Stellentripel $\overrightarrow{r}$. Es bleibt noch, (IT) zu zeigen. Angenommen es gibt eine BPO $bpo \in I(\alpha)$ mit $Prod(bpo, p_{\overrightarrow{r}}) < 0$. Nach

Definition von $I(\alpha)$ gibt es Subterme β und γ von α, so dass $\beta = \gamma^*$ und $bpo \in R(\gamma) \subseteq I(\beta) \subseteq I(\alpha)$. Dann gilt $(bpo)^n \in \mathcal{L}(\beta)$ und $Prod((bpo)^n, p_{\overrightarrow{r}}) = n \cdot Prod(bpo, p_{\overrightarrow{r}}) \leqslant -n$ für jedes $n \in \mathbb{N}$. Damit gilt für β die folgende Eigenschaft: Für jedes n gibt es eine BPO $bpo_n \in \mathcal{L}(\beta)$, so dass $Prod(bpo_n, p_{\overrightarrow{r}}) \leqslant -n$. Es ist nun folgendes zu beachten: Falls α_1 diese Eigenschaft erfüllt, dann erfüllt sie für einen beliebigen Term α_2 auch der BPO-Term $\alpha_1 + \alpha_2$, $\alpha_1 \parallel \alpha_2$, $\alpha_1; \alpha_2$, $\alpha_2; \alpha_1$ und α_1^*. Da β ein Subterm von α ist, impliziert dies, dass auch α die betrachtete Eigenschaft erfüllt. Somit gibt es eine BPO $bpo_N \in \mathcal{L}(\alpha)$, so dass $Prod(bpo_N, p_{\overrightarrow{r}}) < -Init(p_{\overrightarrow{r}})$. Somit folgt $Final(bpo_N, p_{\overrightarrow{r}}) = Prod(bpo_N, p_{\overrightarrow{r}}) + Init(p_{\overrightarrow{r}}) < 0$, ein Widerspruch. □

Insgesamt haben wir also eine geeignete Regionendefinition für BPO-Terme gefunden. Diese lässt nun auch eine endliche linear algebraische Charakterisierung zu. Da eine Markenfluss-Region s von α eine Markenfluss-Region von $R(\alpha)$ ist, können wir die Charakterisierung aus Lemma 4.3.14 verwenden, d.h. wir betrachten die Matrix $\mathbf{A}^{MAR}_{R(\alpha)}$. Zusätzlich soll die Menge von BPOs $I(\alpha)$ noch (IT) erfüllen. Für jede BPO $bpo \in I(\alpha)$ definieren wir daher einen Zeilenvektor $\mathbf{it}_{bpo}$ einer weiteren Matrix $\mathbf{A}^{MARIT}_{I(\alpha)}$, so dass die Ungleichung $\mathbf{it}_{bpo} \cdot \mathbf{x} \geqslant 0$ die Eigenschaft $Prod(bpo, p_{\overrightarrow{s}}) \geqslant 0$ sicherstellt. Hierzu fixieren wir wie in Abschnitt 4.4 für jede Transition $t \in T$ ein Beispielereignis ν_t einer BPO aus R_α, welches mit t beschriftet ist (ein solches existiert per Konstruktion). Wir bezeichnen die Menge der in ν_t eingehenden Kanten mit E^{zu}_t und die Menge der aus ν_t ausgehenden Kanten mit E^{ab}_t. Damit ergibt sich die folgenden linear algebraische Charakterisierung der Menge aller Markenfluss-Regionen von α.

LEMMA 5.1.5 *Lemma 5.1.5*

Sei eine Nummerierung der Menge $\bigcup_{(V,<,l)\in R(\alpha)} <^*= \{e_1, \ldots, e_n\}$ der Kanten aller $*$-Erweiterungen der BPOs aus $R(\alpha)$ gegeben. Die Menge der Markenfluss-Regionen von α entspricht der Menge der zu ganzzahligen Lösungen des homogenen linearen Ungleichungssystems

$$\mathbf{A}^{MAR}_{R(\alpha)} \cdot \mathbf{x} = \mathbf{0}, \mathbf{x} \geqslant \mathbf{0},$$

$$\mathbf{A}^{MARIT}_{I(\alpha)} \cdot \mathbf{x} \geqslant \mathbf{0}$$

bzgl. $\{e_1, \ldots, e_n\}$ korrespondierenden globalen Markenfluss-Funktionen. Die Matrix $\mathbf{A}^{MARIT}_{I(\alpha)}$ setzt sich aus einem Zeilenvektor $\mathbf{it}_{bpo} = (it_{bpo,1}, \ldots, it_{bpo,n})$ für jede $bpo = (V, <, l) \in I(\alpha)$ zusammen. Die Zeilenvektoren sind gegeben durch:

$$it_{bpo,i} = \begin{cases} |V|_l(t) & \text{für } e_i \in E^{ab}_t \text{ und } e_i \notin \bigcup_{t'\in T} E^{zu}_{t'}, \\ -|V|_l(t') & \text{für } e_i \notin \bigcup_{t\in T} E^{ab}_t \text{ und } e_i \in E^{zu}_{t'}, \\ |V|_l(t) - |V|_l(t') & \text{für } e_i \in E^{ab}_t \text{ und } e_i \in E^{zu}_{t'}, \\ 0 & \text{für } e_i \notin \bigcup_{t\in T} E^{ab}_t \text{ und } e_i \notin \bigcup_{t'\in T} E^{zu}_{t'}. \end{cases}$$

BEWEIS: Sei $\mathbf{s} \in \mathbb{N}^n$ und $s = r_{\mathbf{s}}$ die bzgl. $\{e_1, \ldots, e_n\}$ korrespondierende globale Markenfluss-Funktion von $R(\alpha)$. Der Zeilenvektor $\mathbf{it}_{bpo}$ ist derart definiert, dass $\mathbf{it}_{bpo} \cdot \mathbf{s} \geqslant 0 \Leftrightarrow$ (IT) unter der Voraussetzung, dass s eine Markenfluss-Region von $R(\alpha)$ ist. Es folgt mit Lemma 4.3.14 die Behauptung. □

Somit können wir auch Markenfluss-Regionen eines BPO-Terms α durch das Lösen eines endlichen homogenen linearen Ungleichungssystems berechnen. Daher lassen sich im Prinzip Repräsentationen des gesättigt zulässigen Netzes von $\mathcal{L}(\alpha)$ unter Verwendung der linear algebraischen Charakterisierung des letzten Lemmas auf analoge Art und Weise wie im letzten Kapitel berechnen. Insbesondere kann eine Erzeugendensystem-Repräsentation mit demselben Verfahren wie in Abschnitt 4.4 konstruiert werden. Das bedeutet, dass wir eine endliche Erzeugendensystem-Repräsentation von $\mathcal{L}(\alpha)$ berechnen können, indem wir Algorithmus 4.4.3 mit der linear algebraischen Charakterisierung aus Lemma 5.1.5 verwenden. Insgesamt lässt sich so also der konstruktive Teil des in diesem Unterabschnitt aufgeworfenen Syntheseproblems lösen. Wir haben ein Verfahren entwickelt, welches ausgehend von einem BPO-Term α ein S/T-Netz erzeugt, welches das durch $\mathcal{L}(\alpha)$ gegebene Ablaufverhalten aufweist, falls es solch ein Netz gibt. Komplexitätsbetrachtungen lassen sich für diesen Algorithmus ähnlich wie in Kapitel 4 führen.

Beispiel

BEISPIEL: *Wenn wir ausgehend von dem BPO-Term* $b + (a;(a \parallel b)^\star)$*, wie beschrieben, eine Erzeugendensystem-Repräsentation berechnen, so entsteht ein Netz mit 12 Stellen. Entfernen wir nun noch implizite Stellen, so entsteht das in Abbildung 89 dargestellte S/T-Netz. Dieses Netz erlaubt die Schaltfolgen* $abb, ababb, aabbb, abababb, abaabbb, aabbabb, aababbb, \ldots$. *Diese sind nicht durch den Term spezifiziert. Wir können also folgern, dass das Syntheseproblem für diesen Term eine negative Antwort hat.*

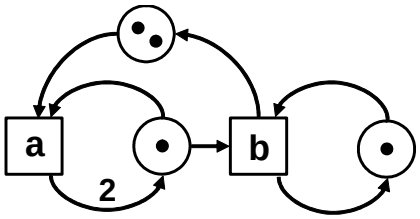

Abbildung 89: Aus einem Term synthetisiertes Netz.

Mit analogen Überlegungen, wie sie hier für Markenfluss-Regionen durchgeführt wurden, lassen sich auch Schritt-Transitions-Regionen und BPO-Transitions-Regionen für Terme definieren und linear algebraisch charakterisieren (siehe [33]). Im Gegensatz zur Berechnung einer Erzeugendensystem-Repräsentation erfordert die Berechnung einer Schrittseparations-Repräsentation bzw. einer BPO-Separations-Repräsentation allerdings einige zusätzliche Überlegungen [33], auf die wir hier nicht näher eingehen. Bisher haben wir auch noch nicht diskutiert, wie sich prüfen lässt, ob ein aus einem BPO-Term synthetisiertes Netz tatsächlich das spezifizierte Verhalten aufweist, oder ob es kein Netz mit einem solchen Verhalten gibt. Auch für einen solchen Übereinstimmungstest bieten sich analoge Vorgehensweisen wie im letzten Kapitel an. Allerdings ist hier eine Verallgemeinerung auf Terme kompliziert. In [32] haben wir einen entsprechende Verallgemeinerung des pessimistischen Übereinstimmungstests entwickelt. Der resultierende Algorithmus ist allerdings tatsächlich sehr umfangreich und aufwändig. Ein formaler Beweis der Korrektheit des Verfahrens steht noch aus. Daher gibt es noch keinen Beweis, dass das Entscheidungsproblem, ob es ein S/T-Netz gibt, welches das von einem BPO-Term spezifizierte

Verhalten aufweist, überhaupt entscheidbar ist. Die Argumentationen im Rahmen des Übereinstimmungstests in [32] sind aber sehr überzeugend, so dass wir zuversichtlich sind, dass das dort vorgestellte Verfahren das Problem entscheidet. Hier wollen wir nun aber nicht mehr weiter auf mögliche Verfahren für einen Übereinstimmungstest eingehen. Stattdessen wollen wir kurz auf die Implementierung des in diesem Abschnitt diskutierten Syntheseverfahrens in VipTool eingehen. Wir haben die Syntheseideen dieses Unterabschnittes in die Plug-Ins TermRegions und TermSynthesis von VipTool integriert. Um BPO-Terme zu spezifizieren, kann das Plug-In LTerms verwendet werden. Gegenüber den anderen Synthese-Plug-Ins bieten diese Plug-Ins zwei wesentliche praktische Fortschritte. Zum einen ist es damit erstmals möglich, Netze aus unendlichen Mengen von BPOs zu synthetisieren. Die unendlichen Mengen werden durch BPO-Terme endlich repräsentiert. Zum anderen erlauben die Kompositionsoperatoren von BPO-Termen eine modulare Spezifikation der betrachteten partiellen Sprache.
Bei der Verwendung dieser Plug-Ins beginnt der Nutzer mit der Spezifikation eines BPO-Terms. Für praktische Belange bzw. für Nutzer, welche mit den zugrundeliegenden Formalismen nicht vertraut sind, wird eine intuitive Nutzer-Schnittstelle, welche die graphische Spezifikation des Verhaltens eines BPO-Terms unterstützt, zur Verfügung gestellt. Zuerst müssen die dem Term zugrundeliegenden BPO-Bausteine wie in VipTool üblich entworfen werden. Dann gibt es einen speziellen Editor zur Erzeugung von BPO-Termen. Neben einer textuellen Repräsentation der Terme unterstützt der Editor auch zwei graphische Sichtweisen (siehe Abbildung 90).
Die erste Sicht illustriert die induktive Komposition der BPOs in einer natürlichen Weise durch das Bilden einer Blockstruktur (ähnlich zu [104]), bei der die Kompositionsoperatoren zwischen den entsprechenden Blöcken visualisiert werden. Dieser Ansatz ähnelt der Visualisierung von Algorithmen durch Struktogramme. Die Struktur von BPO-Termen wird nachgebildet, indem Blöcke entsprechend der Kompositionsoperatoren verknüpft und eingebettet werden. Diese Block-Visualisierung bietet einen klar strukturierten Überblick über einen Term. Sie unterstützt den Nutzer, BPO-Terme konsistent aufzubauen. Es können immer nur vorhandene Blöcke zu größeren Blöcken komponiert werden. Dabei lässt sich von unnötigen Details durch das Verstecken des genauen Inhalts eines Blockes abstrahieren.
Die zweite Sicht übersetzt die Term-Struktur in ein UML-Aktivitätsdiagramm. Diese sind für viele Nutzer geläufiger und intuitiv verständlich. Die dem Term zugrunde liegenden BPOs bilden die Aktivitäten des Diagramms. In diesem Sinne müssen sie als abstrakte Aktivitäten interpretiert werden, welche durch eine in Form einer BPO gegebenen Verhaltensbeschreibung verfeinert werden. Der sequentielle Kompositionsoperator legt die Pfadabhängigkeiten zwischen den BPO-Aktivitäten fest. Der parallele bzw. alternative Kompositionsoperator bestimmt balancierte parallele bzw. alternative Verzweigungen und Zusammenführungen. Der Iterationsoperator ist als zyklische Struktur implementiert. Es ist zu beachten, dass BPO-Terme bei diesem Vorgehen nur eine eingeschränkte Klasse von Aktivitätsdiagrammen erzeugen.
Ausgehend von einem BPO-Term kann mit VipTool nun wie im Falle von Mengen von BPOs ein Netz synthetisiert werden. Es wird, wie in diesem Unterabschnitt beschrieben, ein S/T-Netz aus einer Term-

Spezifikation berechnet. Neben der hier beschriebenen Berechnung einer Erzeugendensystem-Repräsentation ist auch ein hier nicht beschriebenes noch prototypisches Verfahren zur Berechnung einer BPO-Separations-Repräsentation implementiert. Letzteres Verfahren ist in [33] erklärt.
Abbildung 90 zeigt einen Screenshot von VipTool. Er zeigt die zwei graphischen Sichten für den in diesem Abschnitt betrachteten Beispiel-Term sowie ein aus dem Term synthetisiertes Netz.

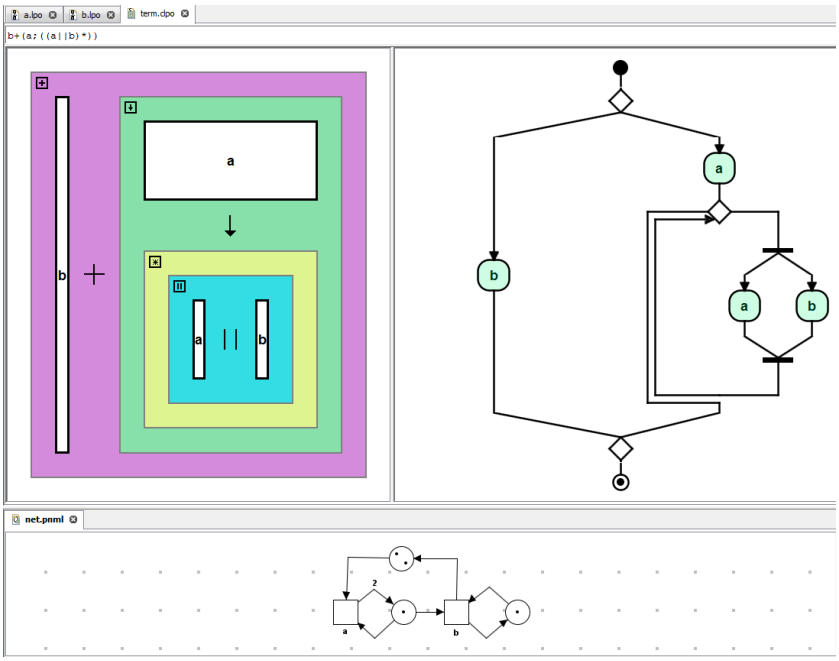

Abbildung 90: Screenshot von VipTool.

Die Verwendung einer Term-Spezifikation für ein Petrinetz-Modell erweitert den in Kapitel 2 beschriebenen Modellierungsansatz. Wiederum sollen zuerst einzelne Szenarien spezifiziert werden, da dies wesentlich einfacher und intuitiver ist als sofort das Gesamtsystem zu betrachten. Im Gegensatz zu 2 müssen die Szenarien nun aber nicht mehr das vollständige System abdecken, sondern es können auch Szenarien eines gewissen Teils des Systems spezifiziert werden. Dieser modulare Ansatz ist häufig sinnvoll, da einzelne Anwender einerseits oft ohnehin nur einen bestimmten Ausschnitt des Systems kennen und andererseits Szenarien des Gesamtsystems häufig sehr komplex und groß sein können. Insbesondere kann die Anzahl der Szenarien des Gesamtsystems exponentiell von der Anzahl der zu betrachtenden Alternativen abhängig sein oder im Falle von Schleifen sogar unendlich sein. Diese Probleme können durch eine modulare Spezifikation von Szenarien mithilfe von BPO-Termen gelöst werden. Diese erlauben es, den Kontrollfluss des Systems teilweise in die Spezifikation zu integrieren. Die einzelnen modularen Szenarien müssen bei diesem Ansatz natürlich händisch geeignet in Beziehung gesetzt werden. Hierzu werden die Kompositionsoperatoren verwendet, wobei der Iterationsoperator auch die Spezifikation von sich wiederholenden Teilszenarien erlaubt. Aus einem BPO-Term lässt sich schließlich automatisch ein integriertes Prozessmodell in der Form eines Petrinetzes synthetisieren.

5.1.2 *Allgemeinere unendliche partielle Sprachen*

Wie schon erwähnt, gibt es partielle Sprachen einschließlich Ablaufsemantiken von S/T-Netzen, welche nicht durch BPO-Terme repräsentiert werden können (siehe Abbildung 85). Der Hauptgrund hierfür ist, dass es mit dem Iterationsoperator nicht möglich ist, BPOs nur an einen Teil einer vorhergehenden BPO anzuhängen. Bei der Iteration von BPOs wird jede BPO immer an die vollständige vorherige BPO angehängt, wodurch jeweils eine Synchronisations-Barriere entsteht. In diesem Unterabschnitt zeigen wir formal, dass aus diesem Grund tatsächlich nicht alle Ablaufsemantiken von S/T-Netzen durch BPO-Terme dargestellt werden können. Weiter führen wir ein Beispiel einer allgemeineren wiederum termbasierten endlichen Repräsentation für unendliche partielle Sprachen ein. Diese Repräsentation verallgemeinert BPO-Terme. Sie erlaubt es, BPOs iterativ an sog. Schnittstellen vorhergehender BPOs anzuhängen, wodurch die vollständige Synchronisations-Barriere vermieden werden kann. Wir erweitern also BPO-Terme um derartige Schnittstellen für die sequentielle Komposition und die Iteration. Weiter zeigen wir, wie sich die Regionenkonzepte des letzten Unterabschnittes für diese Verallgemeinerung anpassen lassen.
An dieser Stelle ist es angebracht kurz zu erwähnen, dass die Möglichkeit mit einer bestimmten Art von endlicher Repräsentation, wie beispielsweise BPO-Termen, eine möglichst große Klasse von unendlichen partiellen Sprachen, insbesondere solchen von S/T-Netzen, darzustellen, aus theoretischer Sicht sicherlich sehr wichtig ist. Aus praktischer Anwendungssicht sind allerdings andere Kriterien wie intuitive und einfache Spezifikationsmöglichkeiten oder Komplexitätsaspekte wichtiger. Bzgl. der Mächtigkeit genügt es wenn ausreichend „relevante“ Sprachen repräsentiert werden können. In dieser Hinsicht sind BPO-Terme normalerweise ausreichend, da sie eine intuitive Möglichkeit zur Darstellung von sich wiederholendem Verhalten erlauben. Bei den beiden erwähnten anwendungsorientierten Kriterien, also Nutzerfreundlichkeit und Komplexität, schneiden BPO-Terme sehr gut ab. Wie im letzten Unterabschnitt gezeigt, erlauben BPO-Terme eine intuitive graphische Spezifikation von Verhalten. Aus Komplexitätssicht gibt es bei BPO-Termen für unendliche Sprachen kaum Einbußen gegenüber vergleichbaren endlichen Sprachen. Im Hinblick auf diese beiden anwendungsorientierten Kriterien schneidet die allgemeinere Term-Spezifikation dieses Unterabschnittes schlechter ab. Allerdings haben wir hier bewusst eine Repräsentationsmöglichkeit gewählt, die immer noch eine intuitive graphische Spezifikation erlaubt und eine zufriedenstellende Komplexität aufweist.
Zum formalen Nachweis, dass es Ablaufsemantiken von S/T-Netzen gibt, welche nicht durch BPO-Terme dargestellt werden können, betrachten wir die in Abbildung 91 und Abbildung 92 illustrierten partiellen Sprachen.

Beispiel

BEISPIEL: *In den beiden Beispielen aus Abbildung 91 und Abbildung 92 ist das Ablaufverhalten des jeweils dargestellten S/T-Netzes durch die dargestellte partielle Sprache gegeben. Es ist hierbei interessant, dass das Beispiel-Netz aus Abbildung 91 unbeschränkt ist, während das Netz in Abbildung 92 einsicher ist.*

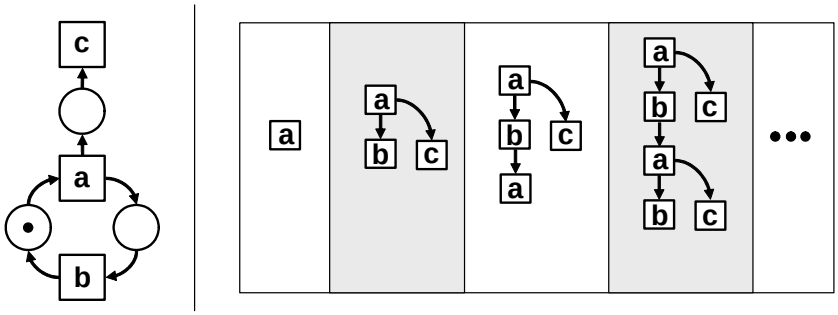

Abbildung 91: Ablaufverhalten eines unbeschränkten Netzes, welches nicht durch einen BPO-Term repräsentiert werden kann.

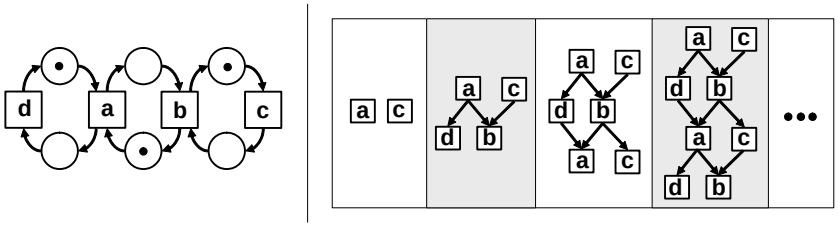

Abbildung 92: Ablaufverhalten eines einssicheren Netzes, welches nicht durch einen BPO-Term repräsentiert werden kann.

Lemma 5.1.6

LEMMA 5.1.6
Die in den Abbildungen 91 und 92 illustrierten unendlichen partiellen Sprachen können nicht durch einen BPO-Term repräsentiert werden.

BEWEIS: Zuerst betrachten wir die partielle Sprache aus Abbildung 91. Wir nehmen an, dass es einen BPO-Term α gibt, welcher diese Sprache erzeugt. Dann gilt notwendiger Weise $I(\alpha) \neq \emptyset$, da es sich um eine unendliche Sprache handelt. Daraus folgt, dass α einen Subterm der Form β^* beinhaltet. Offensichtlich kann dieser Subterm keinen mit c beschrifteten Knoten enthalten, da jedes Vorkommen eines mit c beschrifteten Knotens in einer BPO aus $\mathcal{L}$ maximal ist. Damit sind die einzigen Kandidaten für β die Terme $a; b$ und Potenzen davon. Diese führen offensichtlich nicht zu einem unendlich häufigen Vorkommen von mit c beschrifteten Knoten in $\mathcal{L}$, ein Widerspruch.
Für die partielle Sprache aus Abbildung 92 lässt sich ähnlich argumentieren. Wiederum enthält α einen Subterm der Form β^*. In diesem Fall sind die einzigen Kandidaten für β die Terme $a; d, c; b, a; (d \parallel b)$, $(a \parallel c); b$ und Potenzen dieser Terme. In der partiellen Sprache kommen alle Beschriftungen unendlich oft vor, ein Widerspruch. □

Wir erweitern BPO-Terme nun durch die Einführung der verfeinerten Operatoren der partiellen Iteration ^{*X} und der partiellen sequentiellen Komposition $;_X$ für BPOs und für BPO-Terme. Diese erlauben es, eine BPO nur an Teile einer vorhergehenden BPO anzuhängen. Terme, welche diese neuen Operatoren verwenden, heißen verallgemeinerte BPO-Terme. Durch das Subskript X wird jeweils eine bestimmte Art von Schnittstelle angegeben, die spezifiziert, zu welchen Teilen einer vorangehenden BPO eine nachfolgende BPO angehängt werden kann. Auf der Ebene von BPOs ist solch eine Schnittstelle durch eine Teil-BPO pre der vorausgehenden BPO gegeben. Formal definieren wir ausge-

hend von BPOs $bpo_1 = (V_1, <_1, l_1)$, $bpo_2 = (V_2, <_2, l_2)$ und einer Teil-BPO $pre = (V_1', <_1 |_{V_1' \times V_1'}, l_1|_{V_1'})$, $V_1' \subseteq V_1$, von bpo_1, die folgende Kompositionsregel:

$$bpo_1 ;_{pre} bpo_2 = (V_1 \cup V_2, (<_1 \cup <_2 \cup (V_1' \times V_2))^+, l_1 \cup l_2)$$

Die Teil-BPO pre repräsentiert den Teil von bpo_1, an welchen bpo_2 angehängt wird.
Für Terme sind Schnittstellen durch eine Menge von BPOs X gegeben. Eine BPO aus X wird dann im Rahmen einer sequentiellen Komposition als eine Teil-BPO einer vorausgehenden BPO interpretiert. Die BPOs aus X heißen Schnittstellen-BPOs. Grob gesagt beschreiben die Schnittstellen-BPOs die Teile von BPOs, welche zum Anhängen weiterer BPOs verwendet werden.

DEFINITION 5.1.5 (VERALLGEMEINERTER BPO-TERM) *Definition 5.1.5 (Verallgemeinerter BPO-Term)*
Die Menge der verallgemeinerten BPO-Terme über einem endlichen Alphabet von BPOs $\mathcal{A}$ ist folgendermaßen induktiv definiert: Die BPOs $A \in \mathcal{A}$ und λ sind verallgemeinerte BPO-Terme. Seien α_1 und α_2 verallgemeinerte BPO-Terme und sei X eine Menge von Schnittstellen-BPOs. Dann sind $\alpha = \alpha_1 ;_X \alpha_2$ (partielle sequentielle Komposition), $\alpha = \alpha_1 + \alpha_2$ (Vereinigung), $\alpha = (\alpha_1)^{*_X}$ (partielle Iteration) und $\alpha = \alpha_1 \parallel \alpha_2$ (parallele Komposition) verallgemeinerte BPO-Terme.

Die grundsätzliche Idee hinter der partiellen sequentiellen Komposition von zwei BPOs A_1 und A_2 bzgl. einer Schnittstellen-BPO $x \in X$ ist es, die BPO x als Teil-BPO pre von A_1 wiederzufinden. Aber wir betrachten nur solche Teil-BPOs von A_1, welche ein Präfix einer elementaren BPO aus $\mathcal{A}$, die im Rahmen der Konstruktion von A_1 entsprechend des zugrundeliegenden Terms maximal ist, darstellen.

BEISPIEL: *Beispielsweise betrachten wir für $A_1 ;_X A_2$, $A, B, C \in \mathcal{A}$ und $A_1 = A ;_Y (B \parallel C)$ nur Präfixe von $B \parallel C$, um A_2 anzuhängen.* *Beispiel*

Mit dieser Einschränkung lassen sich Schnittstellen erst sinnvoll spezifizieren, da so „Seiteneffekte“ von vorhergehenden BPOs vermieden werden. Dies wird insbesondere im Kontext der partiellen Iteration deutlich. Für $A_1 = (A ;_Y A ;_Y \ldots ;_Y A)$ werden so beispielsweise nur Präfixe des letzten Vorkommens von A zum Anhängen von A_2 verwendet. Wenn wir hier auch andere Teil-BPOs zum Anhängen zulassen würden, beispielsweise Präfixe von früheren Vorkommen der BPO A, so wäre die resultierende partielle Sprache für einen Nutzer vollkommen unvorhersehbar und nicht mehr bestimmbar.
Für die formale Definition von $\mathcal{L}(\alpha)$ für einen verallgemeinerten BPO-Term α müssen wir entsprechend den Überlegungen des letzten Absatzes also auch für jede BPO $A \in \mathcal{L}(\alpha)$ eine entsprechende Teil-BPO A^{MAX} definieren, an die folgende BPOs angehängt werden dürfen. Hierfür benutzen wir die folgenden Regeln: $A^{MAX} = A$ für $A \in \mathcal{A}$, $(A_1 ;_{pre} A_2)^{MAX} = A_2^{MAX}$ und $(A_1 \parallel A_2)^{MAX} = A_1^{MAX} \parallel A_2^{MAX}$.
Eine Schnittstellen-BPO $x \in X$ kann nun im Rahmen der sequentielle Komposition von zwei BPOs A_1 und A_2 für jedes Präfix pre von A_1^{MAX}, welches x entspricht, verwendet werden. Wir sammeln alle solchen Präfixe in einer Menge $int(x, A_1)$ und schreiben $int(X, A_1) = \bigcup_{x \in X} int(x, A_1)$.

Definition 5.1.6 (Partielle Sprache eines verallgemeinerten BPO-Terms)

DEFINITION 5.1.6 (PARTIELLE SPRACHE EINES VERALLGEMEINERTEN BPO-TERMS)
Wir setzen $\mathcal{L}(\lambda) = \{\lambda\}$ *und* $\mathcal{L}(A) = \{A\}$*. Weiter definieren wir für verallgemeinerte BPO-Terme* α_1 *und* α_2 *und eine Menge von Schnittstellen-BPOs* X *induktiv:*
$\mathcal{L}(\alpha_1 + \alpha_2) = \mathcal{L}(\alpha_1) \cup \mathcal{L}(\alpha_2)$
$\mathcal{L}(\alpha_1 ;_X \alpha_2) = \{A_1 ;_{pre} A_2 \mid A_1 \in \mathcal{L}(\alpha_1), A_2 \in \mathcal{L}(\alpha_2), pre \in \mathrm{int}(X, A_1)\}$
$\mathcal{L}((\alpha_1)^{*_X}) = \{A_1 ;_{pre_2} \dots ;_{pre_n} A_n \mid A_1, \dots, A_n \in \mathcal{L}(\alpha_1), pre_i \in \mathrm{int}(X, A_{i-1})\} \cup \{\lambda\}$
$\mathcal{L}(\alpha_1 \parallel \alpha_2) = \{A_1 \parallel A_2 \mid A_1 \in \mathcal{L}(\alpha_1), A_2 \in \mathcal{L}(\alpha_2)\}$

Beispiel

BEISPIEL: *Abbildung 93 zeigt ein Beispiel für die partielle sequentielle Komposition. Die rechts dargestellte BPO ergibt sich durch die partielle sequentielle Komposition von* bpo_1 *mit sich selbst bzgl. der dargestellten Schnittstellen-BPO.*
Das Beispiel zeigt, dass sich die partielle Sprache aus Abbildung 91 durch einen verallgemeinerten BPO-Term repräsentieren lässt. Mit der dargestellten Schnittstelle ist es möglich, eine zweite Instanz von bpo_1 *nur an das zur Sequenz* $a \to b$ *isomorphe Präfix einer ersten Instanz von* bpo_1 *anzuhängen. Wird dieses Prinzip wiederholt angewandt, so ergibt sich gerade die partielle Sprache aus Abbildung 91. Eine derartige Iteration kann nun durch den verallgemeinerten BPO-Term* $(\mathrm{bpo}_1)^{*_{X_1}}$ *ausgedrückt werden. Genauer gesagt entspricht* $\mathcal{L}((\mathrm{bpo}_1)^{*_{X_1}})$ *der Menge der grauen BPOs aus Abbildung 91.*
Es ist hierbei aber zu beachten, dass die obige Definition von verallgemeinerten BPO-Termen es nicht ermöglicht, die partielle Sprache aus Abbildung 92 zu repräsentieren. Der Grund hierfür ist, dass es für keine Wahl einer zu iterierenden Teil-BPO ein Präfix der Teil-BPO gibt, so dass alle folgenden Ereignisse kausal von allen Ereignissen des Präfixes abhängen, wie es in obiger Definition gefordert wird. In diesem Beispiel hängen verschiedene folgende Ereignisse von verschiedenen Präfixen ab.

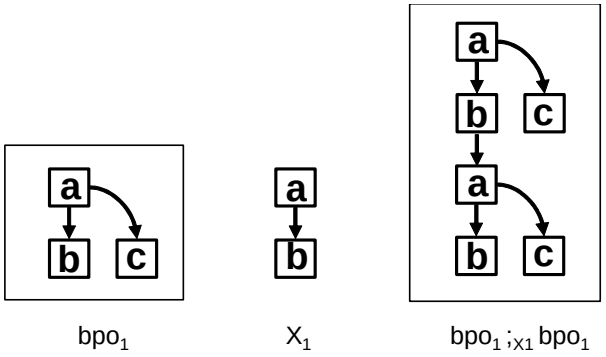

Abbildung 93: Komposition von BPOs bzgl. einer Schnittstelle.

Mit Definition 5.1.6 können wir nun ein entsprechendes Syntheseproblem formulieren.

Problemspezifikation 5.1.2 (Syntheseproblem für verallgemeinerte BPO-Terme)

PROBLEMSPEZIFIKATION 5.1.2 (SYNTHESEPROBLEM FÜR VERALLGEMEINERTE BPO-TERME)
Eingabe: *Ein verallgemeinerter BPO-Term* α*.*
Ausgabe: *Ein markiertes S/T-Netz* (N, m_0) *mit* $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L}(\alpha))$*, falls ein solches Lösungsnetz existiert, und eine „*`notexists`*“-Meldung sonst.*

Um dieses Problem zu lösen, verallgemeinern wir im Weiteren den Begriff der Markenfluss-Region eines BPO-Terms auf verallgemeiner-

te BPO-Terme. Ausgehend von einem verallgemeinerten BPO-Term α kann die Menge $\mathcal{L}(\alpha)$ mit ähnlichen Ideen wie im Falle von BPO-Termen durch eine Repräsentationsmenge $R(\alpha)$ und eine Iterationsmenge $I(\alpha)$ dargestellt werden.
Die Menge $R(\alpha)$ beschränkt wieder die Anzahl der Iterationen des $*_X$-Operators auf Null und Eins. Dann werden in $R(\alpha)$ alle Alternativen, welche durch die $+$-Operationen und die entsprechend beschränkten $*_X$-Operationen erzeugt werden, abgedeckt. Für die $;_X$-Operationen und die $*_X$-Operationen müssen die Schnittstellen-BPOs X geeignet berücksichtigt werden.
Die Menge $I(\alpha)$ repräsentiert wieder alle BPOs, welche iterierten Subtermen von α entsprechen. Da unterschiedliche Subterme bzgl. verschiedener Mengen von Schnittstellen-BPOs iteriert werden, ergänzen wir nun die Menge der BPOs, welche einem iterierten Subterm entspricht, mit einer entsprechenden Menge von Schnittstellen-BPOs. Somit hat $I(\alpha)$ die Form $I(\alpha) = \{(R(\alpha_1), X_1), \ldots, (R(\alpha_n), X_n)\}$, wobei $R(\alpha_i)$ die Repräsentationsmenge des iterierten Subterms α_i ist und X_i die entsprechende Menge von Schnittstellen-BPOs ist. Zur Definition von Markenfluss-Regionen für einen verallgemeinerten BPO-Term α fordern wir wiederum, dass die BPOs aus $I(\alpha)$ mindestens so viele Marken in der durch die Region definierten Stelle produzieren wie sie konsumieren. Natürlich müssen wir diese Forderung nun bzgl. der betrachteten Menge von Schnittstellen-BPOs formulieren.

Beispiel

BEISPIEL: *Für den verallgemeinerten BPO-Term* $\alpha_1 = (\mathrm{bpo}_1)^{*_{X_1}}$, *welcher die partielle Sprache aus Abbildung 91 repräsentiert, gilt beispielsweise* $R(\alpha_1) = \{\lambda, \mathrm{bpo}_1\}$ *und* $I(\alpha_1) = \{(\{\mathrm{bpo}_1\}, X_1)\}$.

Formal definieren wir nun folgendermaßen die Repräsentationsmenge und die Iterationsmenge für einen verallgemeinerten BPO-Term.

Definition 5.1.7 (Repräsentations- und Iterationsmenge)

DEFINITION 5.1.7 (REPRÄSENTATIONS- UND ITERATIONSMENGE)
Die Repräsentationsmenge $R(\alpha)$ *und die Iterationsmenge* $I(\alpha)$ *einer partiellen Sprache* $\mathcal{L}(\alpha)$ *sind für verallgemeinerte BPO-Terme* α_1 *und* α_2 *sowie eine Menge von Schnittstellen-BPOs* X *folgendermaßen induktiv definiert:*

- $R(\lambda) = \{\lambda\}$
- $R(A) = \{A\}$ *für* $A \in \mathcal{A}$
- $R(\alpha_1 + \alpha_2) = R(\alpha_1) \cup R(\alpha_2)$
- $R(\alpha_1 ;_X \alpha_2) = \{A_1 ;_{pre} A_2 \mid A_1 \in R(\alpha_1), A_2 \in R(\alpha_2), pre \in int(X, A_1)\}$
- $R((\alpha_1)^{*_X}) = R(\alpha_1) \cup \{\lambda\}$
- $R(\alpha_1 \parallel \alpha_2) = \{A_1 \parallel A_2 \mid A_1 \in R(\alpha_1),\ A_2 \in R(\alpha_2)\}$
- $I(\lambda) = \emptyset$
- $I(A) = \emptyset$ *für* $A \in \mathcal{A}$
- $I(\alpha_1 + \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$
- $I(\alpha_1 ; \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$
- $I((\alpha_1)^{*_X}) = I(\alpha_1) \cup \{(R(\alpha_1), X)\}$
- $I(\alpha_1 \parallel \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$

Wir bezeichnen die Menge der BPOs aus $I(\alpha)$ *durch* $J(\alpha) = \bigcup_{(R(\alpha'),X)\in I(\alpha)} R(\alpha')$.

Der Begriff der Markenfluss-Region eines BPO-Terms kann folgendermaßen auf verallgemeinerte BPO-Terme übertragen werden. Zuerst fordern wir zusätzlich zu den Ungleichungen (IT) auch, dass der globale Markenfluss auf $R(\alpha)$ konsistent auf $J(\alpha)$ erweitert wird, wobei die Bedingung (ANF) für die BPOs aus $J(\alpha)$ nicht gefordert wird. Damit ist eine Markenfluss-Region eines verallgemeinerten BPO-Terms α also eine globale Markenfluss-Funktion s von $R(\alpha) \cup J(\alpha)$, derart dass die zugehörige Familie von atomaren partiell geordneten Markenflüssen $(\mu_{bpo})_{bpo\in R(\alpha)\cup J(\alpha)}$, $\mu_{bpo} = (bpo^*, \{s|_{<^*}\})$, (ZU) und (AB) erfüllt und die Familie $(\mu_{bpo})_{bpo\in R(\alpha)}$ zusätzlich auch (ANF) erfüllt, also konsistent ist.
Weiter müssen die Ungleichungen (IT) strenger formuliert werden. Da entsprechend einer Menge von Schnittstellen-BPOs iterierte BPOs nur an bestimmte Teile von vorherigen BPOs angehängt werden, müssen wir sicherstellen, dass schon diese Teile und nicht die ganze BPO ausreichend Marken für eine Iteration produzieren. Nur die von diesen Teilen produzierten Marken können in weiteren Iterationen verwendet werden, da Marken ja nur entlang von Kanten der BPOs fließen können. Somit ergibt sich die Forderung, dass solche Teile einer BPO schon mindestens so viele Marken produzieren wie die gesamte BPO konsumiert.

Beispiel

BEISPIEL: *Für unseren Beispiel-Term* $\alpha_1 = (bpo_1)^{*X_1}$ *(siehe Abbildung 93) müssen das* a*- und das* b*-Ereignis mindestens so viele Marken produzieren wie das* a*-, das* b*- und das* c*-Ereignis konsumieren. Nur dann ist es entsprechend möglich, die gesamte BPO* bpo_1 *iterativ nur an die durch die Schnittstelle gegebene Teil-BPO von* bpo_1 *anzuhängen.*

Wir formulieren eine entsprechende Forderung für BPOs aus $I(\alpha)$ durch Ungleichungen, welche sicherstellen, dass die Anzahl der Marken $Final_{pre}(\cdot,\cdot)$, welche nach dem Schalten einer Schnittstellen-BPO verfügbar ist, abzüglich der Anzahl der Marken $Init(\cdot,\cdot)$, welche zum Aktivieren der gesamten BPO notwendig ist, nicht-negativ ist. Die Anzahl $Init(\cdot,\cdot)$ ist durch die Summe der Markenflüsse auf Kanten, welche vom Quellenknoten ausgehen, gegeben. Die Anzahl $Final_{pre}(\cdot,\cdot)$ ist durch die Summe der Markenflüsse auf Kanten, welche von einem Knoten der Schnittstellen-BPO zum Senkenknoten verlaufen, gegeben. Unglücklicherweise lässt sich die Differenz dieser zwei Anzahlen nicht unabhängig von einer konkreten Markenfluss-Verteilung formulieren, wie es im Falle von $Prod(\cdot,\cdot)$ bei (IT) möglich war. Formal ergibt sich die folgende Modifikation von (IT):
Sei $(R(\alpha'),X) \in I(\alpha)$, $A = (V_A, <_A, l_A) \in R(\alpha')$ und $pre = (V, <_A |_{V\times V}, l_A|_V) \in int(X,A)$. Wir bezeichnen durch $V_{pre} = \{v \in V_A \mid \exists u \in V : v = u \vee v <_A u\}$ die Menge der Knoten von A, welche auch Knoten von pre sind oder vor einem Knoten aus pre geordnet sind. Wir betrachten eine Markenfluss-Region r von $\{A\}$. Dann definieren wir $Final_{pre}(A,r) = \sum_{e\in(q_A\cup V_{pre})\times s_A} r(e)$ als die Summe der Markenflüsse auf Kanten vom Quellenknoten und von Knoten aus V_{pre} zum Senkenknoten von $A^\star$ und definieren:

$$Prod_{pre}(A,r) = Final_{pre}(A,r) - Init(A,r).$$

DEFINITION 5.1.8 (MARKENFLUSS-REGION EINES VERALLGEMEINERTEN BPO-TERMS)
Eine Markenfluss-Region s *eines verallgemeinerten BPO-Terms* α *ist eine globale Markenfluss-Funktion von* $R(\alpha) \cup J(\alpha)$, *so dass* $(\mu_{bpo})_{bpo \in R(\alpha) \cup J(\alpha)}$, $\mu_{bpo} = (bpo^*, \{s|_{<^*}\})$, (ZU) *und* (AB) *erfüllt und* $(\mu_{bpo})_{bpo \in R(\alpha)}$ *zusätzlich auch* (ANF) *erfüllt und dass für jede BPO* $bpo \in R(\alpha')$, $(R(\alpha'), X) \in I(\alpha)$, *und* $pre \in int(X, bpo)$ *die folgende Eigenschaft erfüllt ist:*

Definition 5.1.8 (Markenfluss-Region eines verallgemeinerten BPO-Terms)

$$(IT)' \qquad Prod_{pre}(bpo, s) \geqslant 0.$$

Die Definition von $\overrightarrow{s}$ *ergibt sich hierbei dadurch, dass* s *eine Markenfluss-Region von* $R(\alpha)$ *ist. Wir bezeichnen* $\overrightarrow{s}$ *als das zu* s *korrespondierende Stellentripel über* T.

Wir beweisen im Folgenden ähnlich wie für BPO-Terme, dass die Menge der Stellentripel, welche von Markenfluss-Regionen eines verallgemeinerten BPO-Terms α definiert werden, mit der Menge der Stellentripel, welche von Markenfluss-Regionen von $\mathcal{L}(\alpha)$ definiert werden, übereinstimmt. Damit wird durch den hier neu eingeführten Regionenbegriff genau die Menge der zulässigen Stellentripel von $\mathcal{L}(\alpha)$ definiert. Vor der Herleitung dieses Satzes beweisen wir das folgende wichtige technische Lemma. Es zeigt insbesondere, dass $(IT)'$ in einem gewissen Sinne eine Invariante bzgl. verschiedener Markenfluss-Verteilungen für eine Stelle darstellt. In dem Lemma wird gezeigt, wie sich eine für die partielle Iteration einer BPO ungünstige Markenfluss-Verteilung unter bestimmten Bedingungen zu einer günstigeren umverteilen lässt. Die günstigere Verteilung konsumiert noch genauso viele Marken, produziert aber mehr Marken, welche für die partielle Iteration verwendet werden können. Diese Umverteilung ist die zentrale Idee für die Verallgemeinerung der Regionendefinition von BPO-Termen auf verallgemeinerte BPO-Terme. Technisch lässt sich dieses Lemma im Kontext von $(IT)'$ derart anwenden, dass die Eigenschaft $(IT)'$, welche eine entsprechende partielle Iteration erst ermöglicht, durch eine Markenfluss-Umverteilung von der Iterationsmenge auf die relevanten BPOs der Repräsentationsmenge übertragen werden kann.

LEMMA 5.1.7
Sei $(R(\alpha'), X) \in I(\alpha)$, $A = (V_A, <_A, l_A) \in R(\alpha')$ *und* $pre = (V, <_A |_{V \times V}, l_A|_V) \in int(X, A)$. *Seien* r, r' *Markenfluss-Regionen von* $\{A\}$, *welche* $(IT)'$ *erfüllen, mit* ${}^\bullet\overrightarrow{r} = {}^\bullet\overrightarrow{r'}$, $\overrightarrow{r}^\bullet = \overrightarrow{r'}^\bullet$ *und* $r(q_A, s_A) = r'(q_A, s_A) = 0$. *Dann gibt es eine Markenfluss-Region* s *von* $\{A\}$ *mit* ${}^\bullet\overrightarrow{s} = {}^\bullet\overrightarrow{r}$, $\overrightarrow{s}^\bullet = \overrightarrow{r}^\bullet$ *und* $s(q_A, s_A) = 0$, *so dass* $Init(A, s) \leqslant Init(A, r)$ *und* $Prod_{pre}(A, s) \geqslant Prod_{pre}(A, r')$.

Lemma 5.1.7

BEWEIS: Falls $Init(A, r) \geqslant Init(A, r')$, ergibt sich die Behauptung durch Setzen von $s = r'$. Falls $Prod_{pre}(A, r) \geqslant Prod_{pre}(A, r')$, ergibt sich die Behauptung durch Setzen von $s = r$.
Sei nun also $Prod_{pre}(A, r) < Prod_{pre}(A, r')$, d.h. $Final_{pre}(A, r) - Init(A, r) < Final_{pre}(A, r') - Init(A, r')$. Außerdem sei $Init(A, r) < Init(A, r')$. Wir betrachten die Funktion $d = r - r'$ auf $<_A^\star$. Aus $Init(A, r) < Init(A, r')$ folgt $Init(A, d) < 0$. Da ${}^\bullet\overrightarrow{r} = {}^\bullet\overrightarrow{r'}$, $\overrightarrow{r}^\bullet = \overrightarrow{r'}^\bullet$ und das gewöhnliche Prod unabhängig von der konkreten Markenfluss-Verteilung ist, ergibt sich $Final(A, d) - Init(A, d) = Prod(A, d) = 0$, d.h. $Final(A, d) = Init(A, d) < 0$. Aus $Final_{pre}(A, r) - Init(A, r) < Final_{pre}(A, r') - Init(A, r')$ folgt $Final_{pre}(A, d) = Final_{pre}(A, r) -$

$Final_{pre}(A, r') < Init(A, d) = Final(A, d) (< 0)$. Es gilt also $Final(A, d) - Final_{pre}(A, d) > 0$.
Wir behaupten, dass wir s konstruieren können, indem wir den Markenfluss von r auf nicht mit V_{pre} verbundenen Kanten derart umverteilen, dass $Final(A, s - r') - Final_{pre}(A, s - r') \leqslant 0$ gilt. Dann folgt die Behauptung durch folgende Berechnung: $Prod_{pre}(A, s) = Final_{pre}(A, s) - Init(A, s) \geqslant Final(A, s) - Final(A, r') + Final_{pre}(A, r') - Init(A, s) = Prod(A, s) - Final(A, r') + Final_{pre}(A, r') = Prod(A, r') - Final(A, r') + Final_{pre}(A, r') = Prod_{pre}(A, r')$.
Es bleibt zu zeigen, dass es eine derartige Umverteilung gibt. Da $Final(A, d) - Final_{pre}(A, d) > 0$, gibt es eine Kante (v_1, s_A) mit $v_1 \notin V_{pre}$ und $r(v_1, s_A) > r'(v_1, s_A)$. Da ${}^\bullet\overrightarrow{r} = {}^\bullet\overrightarrow{r'}$, $\overrightarrow{r}^\bullet = \overrightarrow{r'}^\bullet$, ist der Marken-Abfluss von v_1 bzgl. r und r' identisch, d.h. es muss eine Kante (v_1, w_1) mit $w_1 \notin V_{pre}$ und $r(v_1, w_1) < r'(v_1, w_1)$ geben. Auch der Marken-Zufluss von w_1 bzgl. r und r' ist identisch, d.h. es muss weiter eine Kante (v_2, w_1) mit $r(v_2, w_1) > r'(v_2, w_1)$ geben. Durch wiederholtes Anwenden dieser Argumentation folgt, dass es eine Folge von Knoten $v_1, w_1, v_2, w_2, \ldots$ gibt, so dass

- $r(v_1, s_A) - 1 \geqslant r'(v_1, s_A)$,
- $\forall i : r(v_i, w_i) \leqslant r'(v_i, w_i) - 1$,
- $\forall i : r(v_{i+1}, w_i) - 1 \geqslant r'(v_{i+1}, w_i)$.

Die Idee ist nun, den Markenfluss von r entlang einer solchen Folge von Knoten umzuverteilen, indem im Prinzip der Fluss auf Kanten (v, v') mit $r(v, v') > r'(v, v')$ um Eins erniedrigt wird und der Fluss auf Kanten (v, v') mit $r(v, v') < r'(v, v')$ um Eins erhöht wird. Es ist dabei zu beachten, dass ein Kante (v, v') mehrfach auftreten kann. Für jedes weitere Auftreten muss allerdings noch gefordert werden, dass die Differenz zwischen $r(v, v')$ und $r'(v, v')$ jeweils um Eins größer ist. Entsprechend wird dann auch für jedes Auftreten der Fluss um Eins erhöht bzw. erniedrigt. Insgesamt ist daher die Häufigkeit des Auftretens einer Kante (v, v') durch $|d(v, v')|$ beschränkt. Also ist jede solche Folge von Knoten endlich.
Sobald in einer solchen Folge ein Knoten aus V_{pre} vorkommt, brechen wir die Folge ab. Dadurch garantieren wir, dass wir den Markenfluss innerhalb des durch V_{pre} gegebenen Präfixes nicht umverteilen. Alle Folgen beinhalten dann bis auf den letzten Knoten nur Knoten, welche nicht aus V_{pre} sind. Außerdem lässt sich obige Argumentation der Konstruktion einer Folge von Knoten auch nicht mehr weiter führen, wenn wir den Quellen- oder Senkenknoten erreichen. In diesem Fall brechen wir die Folge daher auch ab und der Quellen- bzw. Senkenknoten ist der letzte Knoten der Folge. Somit gibt es die drei folgenden Abbruchkriterien für die Konstruktion einer solchen Folge von Knoten. Egal wie die Konstruktion durchgeführt wird, wird immer eines der Abbruchkriterien erreicht.

- Es wird der Knoten $w_n = s_A$ erreicht: In diesem Fall bleiben $Init(A, r)$, $Final(A, r)$, $Final_{pre}(A, r)$ und $Prod_{pre}(A, r)$ durch die Umverteilung unverändert.
- Es wird der Knoten $v_{n+1} = q_A$ erreicht: In diesem Fall werden $Init(A, r)$ und $Final(A, r)$ durch die Umverteilung reduziert. Da sich $Final_{pre}(A, r)$ nicht verändert, wird dadurch $Prod_{pre}(A, r)$ erhöht.

- Es wird ein Knoten $v_{n+1} \in V_{pre}$ erreicht: In diesem Fall, wird zusätzlich noch der Wert von r für die Kante (v_{n+1}, v_{final}) um Eins erhöht. Dies stellt sicher, dass der Marken-Abfluss von v_{n+1} nicht verändert wird, so dass $\bullet\overrightarrow{r}$ und $\overrightarrow{r}\bullet$ auch in diesem Fall erhalten bleiben. Es folgt dann, dass $\mathtt{Init}(A, r)$ und $\mathtt{Final}(A, r)$ durch die Umverteilung nicht verändert werden. Da aber $\mathtt{Final}_{pre}(A, r)$ erhöht wird, wird dadurch $\mathtt{Prod}_{pre}(A, r)$ erhöht.

Solange es also derartige Folgen von Knoten gibt, modifizieren wir r, wie beschrieben, iterativ entlang einer beliebigen solchen Folge. Die nächste Folge wird immer bzgl. des im vorherigen Schritt veränderten Markenflusses berechnet. Offensichtlich terminiert dieses Vorgehen, da in jedem Schritt für eine Kante (v_1, s_A) mit $v_1 \notin V_{pre}$ der Unterschied $r(v_1, s_A) > r'(v_1, s_A)$ reduziert wird und eine solche Kante für die Konstruktion einer entsprechenden Folge von Knoten vorausgesetzt wird. Sei s die durch die Markenfluss-Umverteilung resultierende Markenfluss-Region von $\{A\}$. Wie gefordert, erhalten wir $\mathtt{Final}(A, s - r') - \mathtt{Final}_{pre}(A, s - r') \leqslant 0$, da entsprechend der Terminierungsüberlegungen für das Verfahren $s(v, s_A) \leqslant r'(v, s_A)$ für jedes $v \notin V_{pre}$ gilt. Außerdem gilt per Konstruktion $\mathtt{Init}(A, s) \leqslant \mathtt{Init}(A, r)$, da $\mathtt{Init}(A, r)$ in jedem Umverteilungsschritt gleich bleibt oder kleiner wird. □

SATZ 5.1.2 *Satz 5.1.2*

Sei α ein verallgemeinerter BPO-Term. Es gilt:
(i) Sei s eine Markenfluss-Region von α. Dann gibt es eine Markenfluss-Region r von $\mathcal{L}(\alpha)$ mit $\overrightarrow{r} = \overrightarrow{s}$.
(ii) Sei r eine Markenfluss-Region von $\mathcal{L}(\alpha)$. Dann gibt es eine Markenfluss-Region s von α mit $\overrightarrow{s} = \overrightarrow{r}$.

BEWEIS: **(i)**: Es lassen sich die Aussagen der Lemmata 5.1.2, 5.1.3 und 5.1.4 für verallgemeinerte BPO-Terme anpassen, indem die neuen Definitionen von $\mathtt{Prod}_{pre}(\cdot,\cdot)$ und $\mathtt{Final}_{pre}(\cdot,\cdot)$ verwendet werden. Mit diesen Vorüberlegungen lässt sich der Beweis nach den selben Prinzipien wie in Satz 5.1.1 führen.
Sei s eine Region von α. Dann ist s eine globale Markenfluss-Funktion von $R(\alpha) \cup J(\alpha)$. Jede BPO $bpo \in R(\alpha')$ für ein $(R(\alpha'), X) \in I(\alpha)$ kommt auch als Teil-BPO einer BPO aus $R(\alpha)$ vor. Im Allgemeinen definiert s unterschiedliche Markenfluss-Verteilungen für die zwei Vorkommen der BPO. Eigentlich gilt nur für die Markenfluss-Funktion auf $R(\alpha')$ die Eigenschaft $(IT)'$. Allerdings lässt sich die Markenfluss-Funktion auf $R(\alpha)$ derart umformen, dass der Markenfluss auf einer zur BPO bpo entsprechenden Teil-BPO auch $(IT)'$ erfüllt. Dies lässt sich wie folgt einsehen. Falls die ursprüngliche Markenfluss-Verteilung auf der Teil-BPO mindestens so viele Marken von vorhergehenden Knoten benötigt wie die Markenfluss-Verteilung auf $bpo \in R(\alpha')$, welche $(IT)'$ erfüllt, so kann letztere einfach für die Teil-BPO übernommen werden. Andernfalls gibt es entsprechend Lemma 5.1.7 eine Markenfluss-Verteilung für bpo, welche höchstens so viele Marken benötigt wie die Markenfluss-Verteilung auf der Teil-BPO und $(IT)'$ erfüllt. Somit kann diese Markenfluss-Verteilung für die Teil-BPO übernommen werden. Diese Überlegung ist der wesentliche für verallgemeinerte BPO-Terme gegenüber BPO-Termen notwendige zusätzliche Schritt im Rahmen der Definition einer Markenfluss-Region r von $L(\alpha)$. Mit dieser Voraussetzung lässt sich die Markenfluss-Funktion s nämlich nun konsistent auf

die Iterations-Teile übertragen. Somit lässt sie sich wie in Satz 5.1.1 konsistent zu einer Region r von $\mathcal{L}(\alpha)$ erweitern.
(ii): Aus einer Markenfluss-Region r von $\mathcal{L}(\alpha)$ konstruieren wir eine Markenfluss-Region s von $R(\alpha) \cup J(\alpha)$. Da $R(\alpha) \subseteq \mathcal{L}(\alpha)$, können wir wie in Satz 5.1.1 zur Definition von s auf $R(\alpha)$ einfach r übernehmen. Weiter müssen wir s zusätzlich geeignet auf $J(\alpha)$ definieren. Im Prinzip lassen sich auch hier konsistente Markenflüsse von r übernehmen. Dabei muss aber noch sichergestellt werden, dass $(IT)'$ erfüllt ist. Hier können wir wie im Beweis von Satz 5.1.1 wieder annehmen, dass dies für eine BPO aus $J(\alpha)$ nicht möglich ist. Ähnlich wie im Beweis von Satz 5.1.1 führt dies zu einem Widerspruch, da dann eine BPO aus $\mathcal{L}(\alpha)$ mit negativen Markenflüssen r gefunden werden kann. □

Somit lässt sich also über Markenfluss-Regionen eines verallgemeinerten BPO-Terms α die Menge der zulässigen Stellentripel von $\mathcal{L}(\alpha)$ definieren. Eine endliche linear algebraische Charakterisierung für die Menge der Markenfluss-Regionen von α ergibt sich auf natürliche Weise ähnlich wie im Falle von BPO-Termen. Damit lassen sich entsprechend wie bei BPO-Termen auch Algorithmen zur Berechnung einer geeigneten endlichen Repräsentation des gesättigt zulässigen Netzes von $\mathcal{L}(\alpha)$ aus α formulieren. Insbesondere zur Berechnung einer Erzeugendensystem-Repräsentation lässt sich wiederum direkt Algorithmus 4.4.3 unter Verwendung einer entsprechenden linear algebraischen Charakterisierung nutzen. Ein entsprechender Algorithmus berechnet also aus einem verallgemeinerten BPO-Term α ein S/T-Netz, welches das durch $\mathcal{L}(\alpha)$ gegebene Ablaufverhalten aufweist, falls es solch ein Netz gibt. Auf diese Weise ergibt sich eine Lösung für den konstruktiven Teil des Syntheseproblems aus Definition 5.1.2. Auf die Entscheidbarkeit, ob es überhaupt ein Netz mit dem spezifizierten Verhalten gibt, gehen wir hier wie schon im letzten Unterabschnitt nicht näher ein.
Es ist möglich, den Begriff des BPO-Terms noch weiter zu verallgemeinern, so dass weitere partielle Sprachen wie die Sprache aus Abbildung 92 abgedeckt werden. Für solche Sprachen genügt es nicht, die Teile einer BPO, an die folgende BPOs angehängt werden können, zu spezifizieren, sondern es ist zusätzlich notwendig, genau die Kanten anzugeben, durch welche eine folgende BPO mit dem spezifizierten Teil verbunden werden kann. Allerdings erscheint die Definition geeigneter Regionen für solche Spezifikationen sehr schwierig.
Schließlich gibt es auch noch gänzlich andere Möglichkeiten, um allgemeinere unendliche partielle Sprachen endlich zu repräsentieren. Beispielsweise kann die Iteration von Teilen von BPOs auch durch das Betrachten von rekursiven Gleichungen der Form $X = \alpha(X)$ ausgedrückt werden, wobei $\alpha(X)$ ein BPO-Term ist und X eine Variable, welche in $\alpha(X)$ vorkommt, ist. An den Stellen des Auftretens der Variable X kann dann $\alpha(X)$ iteriert werden. Eine andere Möglichkeit ist die Betrachtung von endlichen Ereignisstrukturen ähnlich wie die Entfaltungen in [34]. Analog zu Cut-Off-Kriterien von Entfaltungen können hier verschiedene Konfigurationen identifiziert werden. Die Intention ist hier, dass das Schalten aller identifizierter Konfigurationen zu den selben Zuständen führt. Durch die Identifikation einer Konfiguration mit einem ihrer Präfixe kann so iteratives Verhalten ausgedrückt werden. Diese beiden Repräsentationsmöglichkeiten für unendliche partielle Sprachen wurden in [154] kurz diskutiert, allerdings ist nicht klar, ob sich in diesen Fällen sinnvolle Regionendefinitionen finden lassen.

Eine weitere vielversprechende Möglichkeit zur Spezifikation unendlicher partieller Sprachen ist die Betrachtung des Ablaufverhaltens eines einssicheren Netzes mit Beschriftungen der Transitionen. Auch eine Untersuchung der Beziehungen der verschiedenen Repräsentationen zu regulären partiellen Sprachen, wie sie beispielsweise in [94] betrachtet werden, ist interessant.

5.2 SPRACHTYPEN

Wir betrachten in dieser Arbeit die Synthese von Petrinetzen aus Szenarien in der Form von endlichen halbgeordneten Abläufen. Eine natürliche und sinnvolle Darstellung für halbgeordnete Abläufe sind, wie dargestellt, BPOs bzw. partielle Sprachen. Diese haben wir bisher unter der Standard-Ablaufsemantik betrachtet. BPOs erlauben es, halbgeordnetes Verhalten in einer sehr allgemeinen Form darzustellen, da sich beliebige Abhängigkeitsbeziehungen und damit auch Unabhängigkeiten zwischen den Ereignissen eines Ablaufes modellieren lassen. Daher eignen sich partielle Sprachen in vielen praktischen Anwendungen zur Spezifikation von Ablaufverhalten.
Auch wenn wir dieses allgemeine Prinzip der Darstellung von beliebigen Abhängigkeitsbeziehungen durch entsprechende Relationen zwischen Ereignissen beibehalten möchten, lässt sich über alternative Sprachtypen zur Modellierung einer Menge von halbgeordneten Abläufen im Rahmen der Synthese von Petrinetzen nachdenken. Wir betrachten in diesem Abschnitt Sprachtypen, welche in verschiedener Hinsicht andere Ausdrucksmöglichkeiten zulassen als partielle Sprachen unter der Ablaufsemantik.
Genauer gesagt betrachten wir zuerst geschichtete Sprachen, welche partielle Sprachen derart erweitern, dass echte Synchronität in den Abläufen berücksichtigt werden kann. In BPOs lässt sich nicht zwischen Nebenläufigkeit und Synchronität unterscheiden. Für Systeme, bei denen echte Synchronität von Ereignissen eine Rolle spielt, sind diese daher zur Modellierung von Abläufen nicht ausreichend. Zur Spezifikation des Verhaltens solcher, in vielen Kontexten sinnvoller Systeme eignen sich dann geschichtete Sprachen. Weiter betrachten wir partielle Sprachen unter den zwei anderen Ablaufsemantiken aus Definition 3.3.12. Wir diskutieren also zum einen partielle Sprachen, welche die Minimalablaufsemantik eines S/T-Netzes darstellen. Hier wird vorausgesetzt, dass nur minimale Kausalitäten, d.h. solche die zwingend notwendig sind, in den Abläufen modelliert werden. Dies entspricht vielfach dem intuitiven Vorgehen bei der Erstellung einer Verhaltens-Spezifikation. Die Minimalität von Abläufen zu fordern, ist daher eine Einschränkung, welche die Entwickler einer Ablauf-Spezifikation zu einer achtsamen und strukturierten Vorgehensweise zwingt. Zum anderen untersuchen wir partielle Sprachen, welche die Prozessablaufsemantik eines S/T-Netzes repräsentieren. Hier ergibt sich die Möglichkeit, nicht nur die Abhängigkeiten zwischen Ereignissen eines Ablaufs, sondern sogar den genauen Fluss von Ressourcen bzw. Marken vorzugeben. Dadurch lassen sich die wahren kausalen Abhängigkeiten von Ereignissen exakt spezifizieren. Eine solche Berücksichtigung des genauen Flusses von Ressourcen ist für viele Anwendungen sinnvoll. Darüber hinaus ist es auch ein sehr intuitives und natürliches Vorgehen, jede in einer BPO spezifizierte Abhängigkeitsbeziehung als eine wahre im System tatsächlich mögliche Abhängigkeit

zu interpretieren und umgekehrt auch zu verlangen, dass all diese möglichen wahren Abhängigkeiten in den BPOs modelliert werden. Nicht zuletzt der Erfolg der Prozessablaufsemantik zur Beschreibung des Verhaltens von Petrinetzen ist ein deutlicher Beweis hierfür.
In Anwendungen kommen verschiedenste Arten von Abläufen vor, welche sich nicht immer auf BPOs unter der Ablaufsemantik zurückführen lassen. In solchen Fällen können die neuen Sprachtypen dieses Abschnittes Abhilfe bringen. Die hinzugewonnenen Ausdrucksmöglichkeiten decken wesentliche Einschränkungen bei der Modellierung von Szenarien mit einfachen BPOs unter der Ablaufsemantik ab und stellen für entsprechende Situationen intuitivere Modellierungsansätze dar.
Wir behandeln in diesem Abschnitt nun das Syntheseproblem des letzte Kapitels, wobei wir als Eingabe einen anderen Sprachtyp verwenden. In einem ersten Unterabschnitt diskutieren wir das Problem der Synthese eines STI-Netzes aus einer endlichen geschichteten Sprache. Im zweiten Unterabschnitt betrachten wir dann die Synthese eines S/T-Netzes aus einer endlichen partiellen Sprache bzgl. der Minimalablaufsemantik und der Prozessablaufsemantik.

5.2.1 *Geschichtete Sprachen*

Wie in Unterabschnitt 3.3.3 erläutert, lässt sich mit BGOs nicht nur die übliche „früher als“-Beziehung zwischen Ereignissen, sondern zusätzlich auch eine „nicht später als“-Beziehung darstellen. Damit kann neben der Nebenläufigkeit auch die Synchronität von Ereignissen in ihrer allgemeinsten Form abgebildet werden. Derartige Beziehungen können, wie in Unterabschnitt 3.2.2 gezeigt, auch von STI-Netzen modelliert werden. Entsprechend stellen BGOs ein geeignetes Ablaufmodell zur Repräsentation von Ablaufverhalten von STI-Netzen dar. In diesem Zusammenhang ergibt sich das folgende Syntheseproblem.

Problemspezifikation 5.2.1 (Syntheseproblem für geschichtete Sprachen)

PROBLEMSPEZIFIKATION 5.2.1 (SYNTHESEPROBLEM FÜR GESCHICHTETE SPRACHEN)
Eingabe: *Eine endliche geschichtete Sprache* $\mathcal{L}$.
Ausgabe: *Ein markiertes STI-Netz* (N, m_0) *mit* $\mathfrak{ABgo}(N, m_0) = PS(\mathcal{L})$, *falls ein solches Lösungsnetz existiert, und eine „*$\mathtt{notexists}$*“-Meldung sonst.*

Beispiel

BEISPIEL: *Als Beispiel betrachten wir in diesem Unterabschnitt die geschichtete Sprache aus Abbildung 49. Wir haben in Unterabschnitt 3.3.4 dargestellt, dass das STI-Netz aus Abbildung 43 das durch die geschichtete Sprache aus Abbildung 49 gegebene Ablaufverhalten hat. Das Netz löst somit das Syntheseproblem für diese Sprache.*

Zur Lösung des Problems werden wir im Weiteren die Definition einer Markenfluss-Region einer partiellen Sprache aus Kapitel 4 auf geschichtete Sprachen übertragen. Bei partiellen Sprachen und S/T-Netzen definiert eine Region eine Stelle, indem die Anfangsmarkierung und die Gewichte der Verbindungskanten der Stelle mit den Transitionen festgelegt werden. Für geschichtete Sprachen und STI-Netze muss eine Region darüber hinaus noch die Gewichte der von der Stelle zu den Transitionen führenden Inhibitorkanten bestimmen. Es zeigt sich, dass der Begriff einer Markenfluss-Region für geschichtete Sprachen aufbauend auf dem Begriff der Markenfluss-Region für partielle Sprachen entwickelt werden kann. Genauer gesagt führt das Weglassen aller „nicht später als“-Beziehungen einer geschichteten Sprache zu

einer Menge von BPOs, welche wir als die der geschichteten Sprache zugrundeliegende partielle Sprache bezeichnen. Wir beginnen die Definition von Markenfluss-Regionen für geschichtete Sprachen, indem wir Markenfluss-Regionen der jeweils zugrundeliegenden partiellen Sprache betrachten. Diese erweitern wir durch sog. mögliche Inhibitorkanten, wie sie in [47] genannt werden. In diesem Sinne ist der Ansatz ähnlich zu [47, 185, 186], wobei dort Regionen von Schritt-Transitionssystemen elementarer Inhibitornetze betrachtet wurden. In unserem Kontext ist die Identifikation möglicher Inhibitorkanten komplizierter als in diesen Arbeiten. Grob gesagt betrachten wir eine mögliche Inhibitorkante, falls für jede mögliche Zwischenmarkierung einer spezifizierten BGO, die in der BGO folgenden Ereignisse durch die Inhibitorkante nicht verhindert werden.
Diese Überlegungen sollen nun formalisiert werden. Hierzu beginnen wir mit einer geeigneten Definition zulässiger Stellen für STI-Netze.

Definition 5.2.1 (Stellenquadrupel)

Definition 5.2.1 (Stellenquadrupel)
Ein Stellenquadrupel über einer endlichen Menge von Transitionen T *ist ein Quadrupel* $st = (st_0, {}^\circ st, st^\circ, st^-)$, *wobei* $st_0 \in \mathbb{N}$, ${}^\circ st, st^\circ \in \mathbb{N}^T$ *und* $st^- : T \to \mathbb{N}_\infty$ *eine Funktion ist.*
Ein markiertes STI-Netz (N, m_0), $N = (P, T, W, I)$, *heißt atomar, falls* $|P| = 1$. *Ein Stellenquadrupel* st *über* T *definiert ein atomares Netz* (N_{st}, m_{st}), $N_{st} = (\{p_{st}\}, T, W_{st}, I_{st})$, *wobei* $m_{st}(p_{st}) = st_0$, $W_{st}(t, p_{st}) = {}^\circ st(t)$ *für alle* $t \in T$, $W_{st}(p_{st}, t) = st^\circ(t)$ *für alle* $t \in T$ *und* $I_{st}(p_{st}, t) = st^-(t)$ *für alle* $t \in T$.

Definition 5.2.2 (Zulässiges Stellenquadrupel)

Definition 5.2.2 (Zulässiges Stellenquadrupel)
Sei $\mathcal{L}$ *eine geschichtete Sprache mit endlicher Beschriftungsmenge* T. *Ein Stellenquadrupel* st *über* T *heißt zulässig bzgl.* $\mathcal{L}$, *falls* $\mathcal{L} \subseteq \mathfrak{AB}\mathfrak{go}(N_{st}, m_{st})$.

Mit diesen Definitionen lassen sich alle weiteren in Abschnitt 4.3 für zulässige Stellentripel und S/T-Netze durchgeführten Überlegungen analog auch auf Stellenquadrupel und STI-Netze übertragen. Dies wird hier nicht im Einzelnen vorgeführt.

Beispiel

Beispiel: *Abbildung 94 illustriert das Konzept des Stellenquadrupels für STI-Netze im Kontext der geschichteten Sprache aus Abbildung 49. Entsprechend den Überlegungen in Unterabschnitt 3.3.4 ist die Stelle in Abbildung 95 links bzgl. der betrachteten geschichteten Sprache zulässig. Die Stelle rechts ist dahingegen nicht zulässig. Die zur Transition* c *gerichtete Inhibitorkante besitzt hier ein Gewicht von* 1 *anstelle von* 2. *Dies ist zu restriktiv. So ist dann beispielsweise ein Schalten von* c *nach einem zweifachen Schalten von* a *nicht mehr möglich. Somit ist* bgo_1 *nicht aktiviert.*

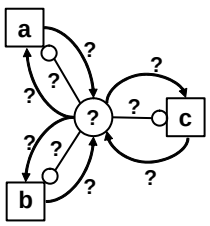

Abbildung 94: Hinzufügen einer Stelle.

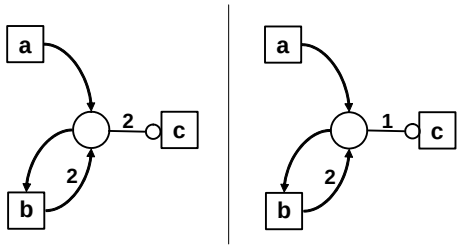

Abbildung 95: Links: Eine zulässige Stelle. Rechts: Eine nicht-zulässige Stelle.

Zur formalen Herleitung eines Regionenbegriffs für geschichtete Sprachen ist die folgende Überlegung für ein markiertes STI-Netz (N, m_0),

$N = (P, T, W, I)$, und eine zugehörige BGO $bgo = (V, \prec, \sqsubset, l)$ wichtig. Wir nehmen an, dass bgo bzgl. (N, m_0) aktiviert ist. Da die Inhibitorkanten von (N, m_0) das Verhalten des (N, m_0) zugrundeliegenden markierten S/T-Netzes (N', m_0), $N' = (P, T, W)$, einschränken, ist bgo offensichtlich auch bzgl. (N', m_0) aktiviert, wenn wir dieses Netz als STI-Netz interpretieren. In einem S/T-Netz können Transitionen, welche synchron aktiviert sind, auch nebenläufig schalten, da dies nicht unterschieden wird. Daher ist auch die BPO $bpo_{bgo} = (V, \prec, l)$, welche durch Weglassen der „nicht später als"-Beziehung aus bgo entsteht, bzgl. (N', m_0) aktiviert. Die BPO bpo_{bgo} nennen wir die der BGO bgo zugrundeliegende BPO. Insgesamt gilt also, dass für eine Menge aktivierter BGOs eines STI-Netzes (N, m_0) die Menge der den BGOs zugrundeliegenden BPOs bzgl. des (N, m_0) zugrundeliegenden S/T-Netzes aktiviert ist. Aus dieser Überlegung folgt als notwendige Bedingung für die Zulässigkeit eines Stellenquadrupels st bzgl. einer geschichteten Sprache $\mathcal{L}$, dass das Stellentripel st', welches durch $st'_0 = st_0$, $^\circ st' = {}^\circ st$ und $st'^\circ = st^\circ$ gegeben ist, zulässig bzgl. der partiellen Sprache aller BPOs, welche BGOs aus $\mathcal{L}$ zugrundeliegen, ist.

Definition 5.2.3 (Zugrundeliegende partielle Sprache)

Definition 5.2.3 (Zugrundeliegende partielle Sprache)
Sei $\mathcal{L}$ eine geschichtete Sprache, dann heißt $\mathcal{L}' = \{bpo_{bgo} \mid bgo \in \mathcal{L}\}$ die $\mathcal{L}$ zugrundeliegende partielle Sprache.

Lemma 5.2.1

Lemma 5.2.1
Sei $\mathcal{L}$ eine geschichtete Sprache mit endlicher Beschriftungsmenge T und sei $\mathcal{L}'$ die $\mathcal{L}$ zugrundeliegende partielle Sprache. Dann gilt für jedes zulässige Stellenquadrupel st von $\mathcal{L}$, dass das st zugrundeliegende Stellentripel st' mit $st'_0 = st_0$, $^\circ st' = {}^\circ st$ und $st'^\circ = st^\circ$ zulässig bzgl. $\mathcal{L}'$ ist.

Beweis: Für einen formalen Beweis müssen einfach die formalen Aktiviertheits-Definitionen für BPOs und BGOs entsprechend den Überlegungen des dem Lemma vorausgehenden Absatzes verwendet werden. □

Somit ergibt sich jedes bzgl. $\mathcal{L}$ zulässige Stellenquadrupel st aus einem bzgl. der zugrundeliegenden partiellen Sprache $\mathcal{L}'$ zulässigen Stellentripel st', indem noch geeignete Inhibitorgewichte durch die Festlegung von st^- gewählt werden. Es lässt sich weiter zeigen, dass durch die Wahl von $st^- \equiv \infty$ immer ein zulässiges Stellenquadrupel entsteht, d.h. für jedes bzgl. $\mathcal{L}'$ zulässige Stellentripel st' gilt, dass das Stellenquadrupel st, welches durch $st_0 = st'_0$, $^\circ st = {}^\circ st'$, $st^\circ = st'^\circ$ und $st^- \equiv \infty$ gegeben ist, bzgl. $\mathcal{L}$ zulässig ist. Letzteres ist eine direkt Folgerung daraus, dass (N_{st}, m_{st}) keine Inhibitorkanten enthält, also identisch zu $(N_{st'}, m_{st'})$ ist, und eine einer BGO zugrundeliegende BPO mehr Schrittfolgen erzeugt als die BGO, d.h. die BGO ist stärker geordnet als die BPO. Wir können also folgern, dass die Menge der bzgl. $\mathcal{L}$ zulässigen Stellenquadrupel st mit $st^- \equiv \infty$ zur Menge der bzgl. der partiellen Sprache $\mathcal{L}'$ zulässigen Stellentripel korrespondiert. Diese Menge lässt sich mithilfe der Markenfluss-Regionen für partielle Sprachen aus dem letzten Kapitel charakterisieren. Alle weiteren bzgl. $\mathcal{L}$ zulässigen Stellenquadrupel ergeben sich durch Anpassung von st^-. Entsprechend führen wir eine Definition von Markenfluss-Regionen für geschichtete Sprachen und STI-Netze aufbauend auf der Definition von Markenfluss-Regionen für partielle Sprachen und S/T-Netze des letzten Kapitels ein. Genauer gesagt beginnen wir mit Markenfluss-Regionen

der einer geschichteten Sprache $\mathcal{L}$ zugrundeliegenden partiellen Sprache $\mathcal{L}'$. Hierdurch lassen sich, wie erklärt, alle bzgl. $\mathcal{L}$ zulässigen Stellenquadrupel st mit $st^- \equiv \infty$ gewinnen. Dann prüfen wir für jedes solche Stellenquadrupel st, welche andere Wahl von st^- auch die Zulässigkeit von st gewährleistet. Hierbei ist entscheidend, dass die Erhöhung eines Inhibitorgewichtes in einem STI-Netz dessen Verhaltenseinschränkung vermindert, d.h. die Menge der aktivierten BGOs des Netzes kann hierdurch nur größer werden. Daher führt eine Erhöhung der Funktion st^- eines zulässigen Stellenquadrupels st wieder zu einem zulässigen Stellenquadrupel. Ausgehend von einem zulässigen Stellenquadrupel st mit $st^- \equiv \infty$ gibt es also für jede Transition $t \in T$ einen minimalen Wert $st^-_{min}(t) \in \mathbb{N}_\infty$ derart, dass das Stellenquadrupel st′ mit $st'_0 = st_0$, $^\circ st' = {}^\circ st$, $st'^\circ = st^\circ$, $st^-(t') = \infty$ für alle $t' \in T \setminus \{t\}$ und $st'^-(t) = st^-_{min}(t)$ zulässig ist. Hier ist nun entscheidend, dass sich diese verschiedenen minimalen Werte $st^-_{min}(t)$ für verschiedene $t \in T$ zu einer globalen minimalen Schranke zusammenfassen lassen. Es gilt für ein zulässiges Stellenquadrupel st mit $st^- \equiv \infty$, dass ein Stellenquadrupel st′ mit $st'_0 = st_0$, $^\circ st' = {}^\circ st$ und $st'^\circ = st^\circ$ genau dann zulässig ist, wenn $st'^-(t) \geqslant st^-_{min}(t)$ für alle $t \in T$. Diese Zusammenfassung der einzelnen unteren Grenzen $st^-_{min}(t)$ ist möglich, da verschiedene Inhibitorkanten keinen kausalen Zusammenhang zwischen einander aufweisen bzw. erzeugen. In anderen Worten bedeutet dies, dass die Aktiviertheit einer BGO in einem STI-Netz bzgl. der Inhibitorkanten geprüft werden kann, indem jede Inhibitorkante einzeln betrachtet wird. Insgesamt lässt sich also die Menge der bzgl. einer geschichteten Sprache $\mathcal{L}$ zulässigen Stellenquadrupel st durch die Menge der bzgl. der zugrundeliegenden partiellen Sprache $\mathcal{L}'$ zulässigen Stellentripel jeweils zusammen mit einer globalen unteren Schranke $st^-_{min} : T \to \mathbb{N} \cup \{\infty\}$ für st^- charakterisieren.
Formal definieren wir nun also zunächst eine sog. konsistente globale Markenfluss-Funktion einer geschichteten Sprache als eine Markenfluss-Region der zugrundeliegenden partiellen Sprache.

DEFINITION 5.2.4 (KONSISTENTE GLOBALE MARKENFLUSS-FUNKTION)
Sei $\mathcal{L}$ eine geschichtete Sprache. Wir bezeichnen eine Markenfluss-Region der $\mathcal{L}$ zugrundeliegenden partiellen Sprache $\mathcal{L}'$ als konsistente globale Markenfluss-Funktion von $\mathcal{L}$.

Definition 5.2.4 (Konsistente globale Markenfluss-Funktion)

BEISPIEL: *Abbildung 96 illustriert oben eine konsistente globale Markenfluss-Funktion der geschichteten Sprache aus Abbildung 49. Die Abbildung zeigt nur die echt positiven Markenflüsse zugeordnet zu den entsprechenden $\prec$-Kanten (insbesondere werden Nicht-Gerüstkanten mit einem Markenfluss von Null weggelassen). Unten in der Abbildung ist die zu der konsistenten globalen Markenfluss-Funktion korrespondierende Stelle abgebildet. Die Gewichte von Inhibitorkanten werden zunächst alle als ∞ angenommen. Entsprechend den bisherigen Überlegungen ist diese Stelle dann bzgl. der betrachteten geschichteten Sprache zulässig.*

Beispiel

Wir wissen schon allgemein, dass die Menge der zulässigen Stellenquadrupel st von $\mathcal{L}$ mit $st^- \equiv \infty$ durch die Menge der Stellentripel $\overrightarrow{r}$ für Markenfluss-Regionen r von $\mathcal{L}'$ gegeben ist. Sei im Weiteren r eine konsistente globale Markenfluss-Funktion von $\mathcal{L}$ und st das entsprechende durch $st_0 = \overrightarrow{r}_0$, $^\circ st = {}^\circ \overrightarrow{r}$, $st^\circ = \overrightarrow{r}^\circ$ und $st^- \equiv \infty$ gegebene zulässige Stellenquadrupel.
Es bleibt noch, die untere Schranke st^-_{min} für die Inhibitorgewichte von st aus der Markenfluss-Funktion r abzuleiten. Hierzu verwenden wir

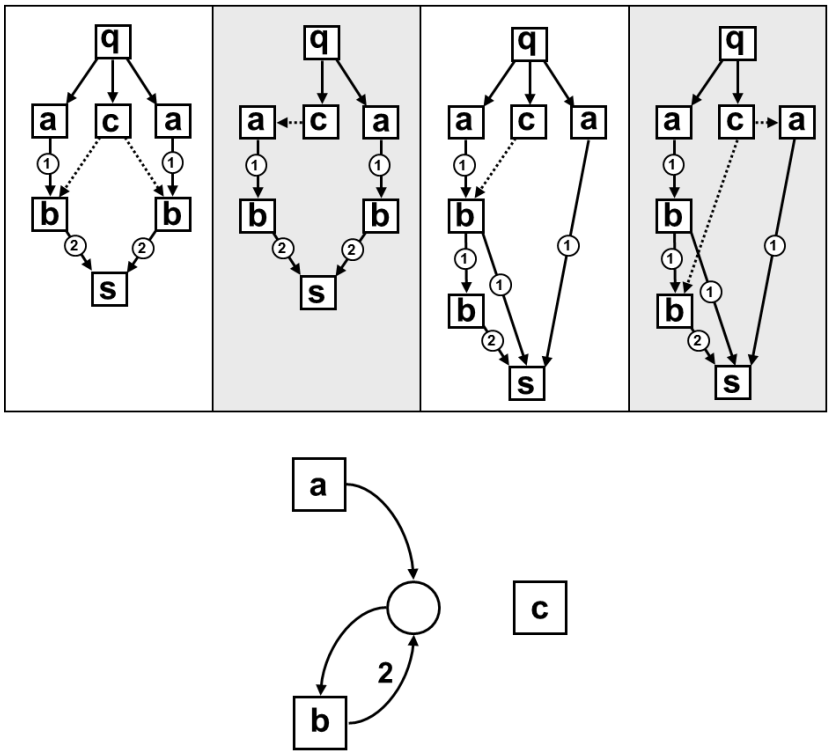

Abbildung 96: Konsistente globale Markenfluss-Funktion und korrespondierende Stelle.

die folgende Strategie. Für einen Knoten v einer BGO $bgo = (V, \prec, \sqsubset, l) \in L$ berechnen wir das minimale Inhibitorgewicht $Inh(r, v)$ einer Inhibitorkante von p_{st} zu $l(v)$ derart, dass die Transition $l(v)$ entsprechend der durch bgo gegebenen kausalen Abhängigkeiten in (N_{st}, m_{st}) schalten kann. In anderen Worten wird das Ereignis v des Ablaufs bgo nicht durch eine Inhibitorkante von p_{st} zu $l(v)$ mit $I(p_{st}, l(v)) \geqslant Inh(r, v)$ verhindert, aber durch eine Kante mit $I(p_{st}, l(v)) < Inh(r, v)$. In letzterem Fall verhindert eine Zwischenmarkierung des Ablaufs bgo mit einer zu großen Anzahl von Marken in p_{st} das Ereignis v. Um demzufolge $Inh(r, v)$ zu berechnen, ist es nötig, die Anzahl der Marken in p_{st} für jede Zwischenmarkierung, welche v entsprechend bgo aktiviert, zu berechnen. Diese Markierungen sind gerade durch die Präfixe von v bestimmt. Das Maximum all dieser Anzahlen von Marken in p_{st} für Markierungen entsprechender Präfixe definiert dann $Inh(r, v)$ für v, da entsprechend dem Ablauf bgo die Transition $l(v)$ in jeder dieser Markenallokationen von p_{st} aktiviert sein soll. Für ein Präfix kann die zugehörige Anzahl der Marken in p_{st} mithilfe der Markenfluss-Funktion r berechnet werden. Sie ist durch die Folgemarkierung von (N_{st}, m_{st}) nach dem Schalten des Präfixes gegeben. Diese Anzahl nennen wir Folgemarkierung des Präfixes bzgl. r. Per Konstruktion korrespondieren die Werte von r auf Kanten, welche von einem Knoten des Präfixes zu einem anderen Knoten des Präfixes verlaufen, zu Marken, welche von Ereignissen des Präfixes in p_{st} produziert und konsumiert werden. Andererseits entsprechen die Werte von r auf Kanten, welche von einem Knoten des Präfixes zu einem Knoten außerhalb des Präfixes verlaufen, Marken, welche von Ereignissen des Präfixes in p_{st} produziert werden

und auch nach der Ausführung des Präfixes in p_{st} verbleiben. Folglich wird die Folgemarkierung eines Präfixes bzgl. r mithilfe der Werte von r auf Kanten, welche das Präfix verlassen, bestimmt.

DEFINITION 5.2.5 (FOLGEMARKIERUNG EINES PRÄFIXES)
Sei $\mathcal{L}$ eine geschichtete Sprache und r eine konsistente globale Markenfluss-Funktion von $\mathcal{L}$. Sei $bgo' = (V', \prec', \sqsubset', l') $ ein Präfix von $bgo = (V, \prec, \sqsubset, l) \in \mathcal{L}$. Die Folgemarkierung von bgo' bzgl. r wird durch $m_{bgo'}(r) = \sum_{u \in V' \cup \{q_{bpo_{bgo}}\}, v \in (V \setminus V') \cup \{s_{bpo_{bgo}}\}, u \prec^* v} r(u,v)$ definiert und bezeichnet.

Definition 5.2.5 (Folgemarkierung eines Präfixes)

Die Folgemarkierung eines Präfixes bzgl. r kann wie folgt äquivalent berechnet werden, indem alle zum Präfix korrespondierenden Transitionen in dem Netz $(N_{\overrightarrow{r}}, m_{\overrightarrow{r}})$ geschaltet werden. $m_{bgo'}(x) = \sum_{u \in V' \cup \{q_{bpo_{bgo}}\}, v \in (V \setminus V') \cup \{s_{bpo_{bgo}}\}, u \prec^* v} r(u,v) = \sum_{u \in V' \cup \{q_{bpo_{bgo}}\}} (\sum_{u \prec^* v} r(u,v) - \sum_{v \prec^* u} r(v,u)) = Ab_r(q_{bpo_{bgo}}) + \sum_{v \in V'} (Ab_r(v) - Zu_r(v)) = m_{\overrightarrow{r}}(p_{\overrightarrow{r}}) + \sum_{v \in V'} (W_{\overrightarrow{r}}(l(v), p_{\overrightarrow{r}}) - W_{\overrightarrow{r}}(p_{\overrightarrow{r}}, l(v)))$. Damit ist die Folgemarkierung eines Präfixes bzgl. r unabhängig von der konkreten Markenfluss-Verteilung von r und nur von $\overrightarrow{r}$ abhängig.
Nun wird die Berechnung von $Inh(r,v)$ derart durchgeführt, dass alle Präfixe von v identifiziert werden und die Folgemarkierung all dieser Präfixe bzgl. r berechnet wird. Das Maximum all dieser Zahlen ergibt $Inh(r,v)$. Dieser Wert gibt an wie klein das Inhibitorgewicht $I(p_{st}, l(v))$ gewählt werden kann, so dass das Ereignis v dadurch nicht verhindert wird. Der Wert $Inh(r,v)$ wird im Weiteren als Inhibitorwert von v bzgl. r bezeichnet.

DEFINITION 5.2.6 (INHIBITORWERT)
Sei $\mathcal{L}$ eine geschichtete Sprache und r eine konsistente globale Markenfluss-Funktion von $\mathcal{L}$. Sei weiter v ein Knoten einer BGO $bgo \in \mathcal{L}$. Der Inhibitorwert von v bzgl. r ist definiert durch $Inh(r,v) = \max\{m_{bgo'}(r) \mid bgo'$ Präfix von v bzgl. $bgo\}$.

Definition 5.2.6 (Inhibitorwert)

Nachdem wir $Inh(r,v)$ für alle Knoten v aller BGOs aus $\mathcal{L}$ bestimmt haben, lässt sich das minimale Inhibitorgewicht $I(p_{st}, t)$, welches sicherstellt, dass kein mit t beschrifteter Knoten durch dieses Gewicht verhindert wird, bestimmen, indem das Supremum aller Inhibitorwerte $Inh(r,v)$ für mit t beschriftete Knoten v berechnet wird. Dies ergibt gerade den gesuchten Wert $st^-_{min}(t)$, denn die Tatsache, dass kein mit t beschriftetes Ereignis durch $I(p_{st}, t)$ verhindert wird, entspricht gerade der Zulässigkeitsforderung. Somit gilt $st^-_{min}(t) = sup(\{Inh(r,v) \mid v \in \bigcup_{(V,\prec,\sqsubset,l) \in \mathcal{L}} V, l(v) = t\} \cup \{0\})$. Zur Definition von Regionen, betrachten wir also alle Inhibitorwerte aller Knoten von $\mathcal{L}$ bzgl. einer Markenfluss-Funktion r. Die Suprema von gleich beschrifteten Knoten führen zu den minimalen Inhibitorgewichten st^-_{min}, welche die Zulässigkeit des von r definierten Stellenquadrupels st noch gewährleisten. Somit ist eine Markenfluss-Region einer geschichteten Sprache bzgl. STI-Netzen durch eine konsistente globale Markenfluss-Funktion r ergänzt um eine Abbildung $I : T \to \mathbb{N} \cup \{\infty\}$, welche ein Inhibitorgewicht größer als $st^-_{min}(t)$ für jede Transition $t \in T$ festlegt, definiert.

Definition 5.2.7 (Markenfluss-Region)

DEFINITION 5.2.7 (MARKENFLUSS-REGION)
Sei $\mathcal{L}$ eine geschichtete Sprache mit endlicher Beschriftungsmenge T. Eine Markenfluss-Region von $\mathcal{L}$ bzgl. STI-Netzen ist ein Paar $ri = (r, I)$, wobei r eine konsistente globale Markenfluss-Funktion von $\mathcal{L}$ ist und $I : T \to \mathbb{N}_\infty$ eine Abbildung mit

$$(INH) \qquad I(t) \geqslant \sup(\{Inh(r,v) \mid v \in \bigcup_{(V,\prec,\sqsubset,l)\in\mathcal{L}} V, l(v) = t\} \cup \{0\})$$

für alle $t \in T$ ist, welche jeder Transition ein Inhibitorgewicht zuordnet. Das zu ri korrespondierende Stellenquadrupel $\overrightarrow{ri}$ ist durch $\overrightarrow{ri}_0 = \overrightarrow{r}_0$, $\circ\overrightarrow{ri} = \circ\overrightarrow{r}$, $\overrightarrow{ri}^\circ = \overrightarrow{r}^\circ$ und $\overrightarrow{ri}^- = I$ gegeben.

Beispiel

BEISPIEL: *Abbildung 97 zeigt oben die konsistente globale Markenfluss-Funktion aus Abbildung 96 ergänzt um Inhibitorwerte für alle Knoten. Die Inhibitorwerte sind in kleinen an die Knoten angehängten Quadraten dargestellt. Beispielsweise hat das c-Ereignis der ersten BGO vier Präfixe. Das leere Präfix mit Folgemarkierung 0, zwei Präfixe, welche nur aus einem a-Ereignis bestehen jeweils mit Folgemarkierung 1 und ein aus beiden a-Ereignissen bestehendes Präfix mit Folgemarkierung 2. Somit ergibt sich der Inhibitorwert als 2.*
Die abgebildete konsistente globale Markenfluss-Funktion r zusammen mit der Abbildung I, welche durch $I(a) = 3, I(b) = 3, I(c) = 2$ gegeben ist, definiert eine Markenfluss-Region $ri = (r, I)$ der geschichteten Sprache. Die I-Werte sind größer als die entsprechenden Inhibitorwerte. Tatsächlich ist I hier sogar minimal gewählt. Die Region ri weist also die minimal möglichen Inhibitorgewichte auf, d.h. $ri' = (r, I')$ ist eine Markenfluss-Region, falls $I' \geqslant I$, und keine Markenfluss-Region, falls $I' \ngeqslant I$. Unten in der Abbildung ist die zu ri korrespondierende Stelle abgebildet.

Entsprechend den bisherigen Überlegungen ist die Menge der durch Markenfluss-Regionen von $\mathcal{L}$ definierten Stellenquadrupel genau die Menge der bzgl. $\mathcal{L}$ zulässigen Stellenquadrupel. Dieses Hauptresultat des Unterabschnittes zeigen wir im Folgenden formal. Der Beweis verwendet essentiell die Definition der Aktiviertheit einer BGO bzgl. eines STI-Netzes aus Unterabschnitt 3.3.4, welche auf Schrittlinearisierungen basiert. Folgendes Lemma zeigt hierfür, dass die Frage nach der Aktiviertheit eines Ereignisses einer BGO nach einem Präfix des Ereignisses auf die Menge der Schrittlinearisierungen zurückgeführt werden kann.

Lemma 5.2.2

LEMMA 5.2.2
Sei $go = (V, \prec, \sqsubset)$ eine geschichtete Ordnung, $V' \subseteq V$ und $v \in V$. Dann definiert V' genau dann ein Präfix von v bzgl. go, wenn es eine Schrittlinearisierung $go' = (V, \prec', \sqsubset') \in Slin(go)$ gibt, so dass V' ein Präfix von v bzgl. go' definiert.

BEWEIS: Die **Wenn**-Aussage folgt offensichtlich aus $go' \in Seq(go)$.
Für die **Genau dann**-Aussage konstruieren wir eine Folge von Knotenmengen $V_1 \dots V_n$ mit $V = V_1 \cup \dots \cup V_n$, welche go' durch $\prec' = \bigcup_{i<j} V_i \times V_j$ und $\sqsubset' = ((\bigcup_{i=1}^n V_i \times V_i) \cup \prec') \setminus id_V$ festlegt. Wir gehen folgendermaßen vor: $V_1 = \{v \in V' \mid \forall v' \in V' : v' \not\prec v\}$, $V_2 = \{v \in V' \setminus V_1 \mid \forall v' \in V' \setminus V_1 : v' \not\prec v\}$ u.s.w., d.h. wir definieren $V_i \subseteq V'$ als die Menge der Knoten $\{v \in V' \setminus (\bigcup_{j=1}^{i-1} V_j) \mid \forall v' \in V' \setminus (\bigcup_{j=1}^{i-1} V_j) : v' \not\prec v\}$, welche minimal bzgl. der Einschränkung der Ordnung $\prec$ auf die Knotenmenge $V' \setminus (\bigcup_{j=1}^{i-1} V_j)$ ist. Dieses Vorgehen führen wir durch bis $V' \setminus (\bigcup_{j=1}^{i} V_j) = \emptyset$. Dann fahren wir mit

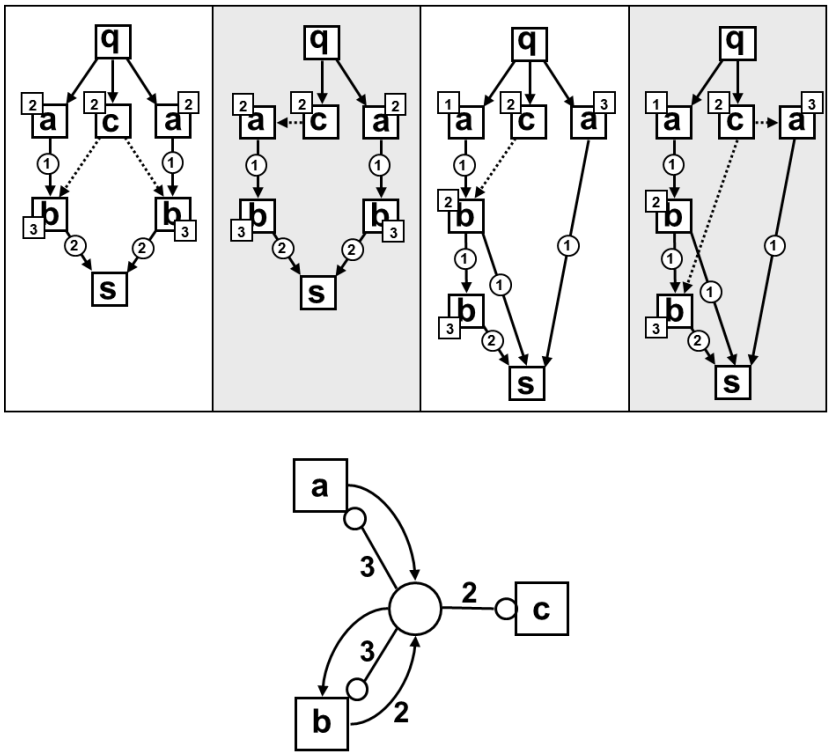

Abbildung 97: Markenfluss-Region und korrespondierende Stelle.

demselben Vorgehen auf der Menge $V \setminus V' = V \setminus (\bigcup_{j=1}^{i} V_j)$ fort, d.h. $V_{i+1} = \{v \in V \setminus (\bigcup_{j=1}^{i} V_j) \mid \forall v' \in V \setminus (\bigcup_{j=1}^{i} V_j) : v' \not\prec v\}$ usw. Durch diese Konstruktion definiert V' ein Präfix von v bzgl. go'. Es lässt sich auch leicht einsehen, dass $go' \in \mathrm{Slin}(go)$. □

SATZ 5.2.1 *Satz 5.2.1*

Sei $\mathcal{L}$ eine geschichtete Sprache mit endlicher Beschriftungsmenge T. Ein Stellenquadrupel über T ist genau dann zulässig bzgl. $\mathcal{L}$, wenn es zu einer Markenfluss-Region von $\mathcal{L}$ korrespondiert.

BEWEIS: **Wenn**-Aussage: Sei st zu einer Markenfluss-Region $ri = (r, I)$ korrespondierend. Wir müssen zeigen, dass $bgo \in \mathcal{L}$ in dem Netz (N_{st}, m_{st}) aktiviert ist. Da r eine Region der $\mathcal{L}$ zugrundeliegenden partiellen Sprache $\mathcal{L}'$ ist, gilt entsprechend Satz 4.3.5, dass bpo_{bgo} bzgl. $(N_{\overrightarrow{r}}, m_{\overrightarrow{r}})$ aktiviert ist. Es gilt $\{\sigma(bgo') \mid bgo' \in \mathrm{Slin}(bgo)\} \subseteq \{\sigma(bpo') \mid bpo' \in \mathrm{Slin}(bpo_{bgo})\}$. Da die Schaltregel für STI-Netze und S/T-Netze bis auf die Berücksichtigung der Inhibitorkanten übereinstimmt, lässt sich hieraus folgern, dass bgo bzgl. $(N_{st'}, m_{st'})$ aktiviert ist, wobei $st'_0 = \overrightarrow{r}_0$, $^\circ st' = {}^\circ \overrightarrow{r}$, $st'^\circ = \overrightarrow{r}^\circ$ und $st^- \equiv \infty$. Die Stellenquadrupel st und st′ unterscheiden sich nur bzgl. der Inhibitorgewichte. Um zu zeigen, dass bgo auch bzgl. (N_{st}, m_{st}) aktiviert ist, betrachten wir eine Schrittfolge $\sigma(bgo') = \tau_1 \dots \tau_n$ mit $bgo' \in \mathrm{Slin}(bgo)$. Wir müssen zeigen, dass $\sigma(bgo')$ bzgl. (N_{st}, m_{st}) aktiviert ist. Hierfür zeigen wir induktiv für $0 \leqslant k \leqslant n-1$, dass, falls $\sigma_k = \tau_1 \dots \tau_k$ in (N_{st}, m_{st}) aktiviert ist, dann τ_{k+1} in der Folgemarkierung m von σ_k aktiviert ist. Aus der Aktiviertheit von bgo und damit $\sigma(bgo')$ bzgl. $(N_{st'}, m_{st'})$ folgt direkt die erste Forderung $m \geqslant {}^\circ\tau_{k+1}$ für die Aktiviertheit von τ_{k+1}. Es bleibt noch, die Forderung $m(p_{st}) \leqslant^- \tau_{k+1}(p_{st}) = \min\{^-t(p_{st}) \mid$

$t \in \tau_{k+1}\}$ nachzuweisen. Wir müssen also $m(p_{st}) \leqslant I_{st}(p_{st}, t)$ für alle $t \in \tau_{k+1}$ zeigen. Wir betrachten die BGO $bgo'_k = (V_k, \prec_k, \sqsubset_k, l_k)$ mit $\sigma(bgo'_k) = \sigma_k$. Dann ist bgo'_k ein Präfix eines Knotens v mit $l(v) = t$ bzgl. bgo'. Nach Lemma 5.2.2 definiert V_k dann auch ein Präfix bgo_k von v bzgl. bgo. Es genügt zu zeigen, dass $m(p_{st}) = m_{bgo_k}(r)$, da $m_{bgo_k}(r) \leqslant Inh(r, v) \leqslant I(l(v)) = st^-(t) = I_{st}(p_{st}, t)$. Hierzu berechnen wir: $m(p_{st}) = m_{st}(p_{st}) + \sum_{i=1}^{k} \sum_{t \in \tau_i} \tau_i(t)(W_{st}(t, p_{st}) - W_{st}(p_{st}, t)) = m_{st}(p_{st}) + \sum_{v \in V_k}(W_{st}(l(v), p_{st}) - W_{st}(p_{st}, l(v))) = m_{bgo_k}(r)$. Die letzte Gleichung folgt entsprechend der Anmerkung im Anschluss an Definition 5.2.5.

Genau dann-Aussage: Sei st ein bzgl. $\mathcal{L}$ zulässiges Stellenquadrupel. Dann ist entsprechend Lemma 5.2.1 das st zugrundeliegende Stellentripel st′ bzgl. der $\mathcal{L}$ zugrundeliegenden partiellen Sprache $\mathcal{L}'$ zulässig. Nach Satz 4.3.5 gibt es also eine Markenfluss-Region r von $\mathcal{L}'$ mit $\overrightarrow{r} = st'$. Wir zeigen nun, dass $ri = (r, st^-)$ eine Markenfluss-Region von $\mathcal{L}$ ist. Diese definiert st. Hierzu müssen wir nur zeigen, dass $st^-(t) \geqslant \sup(\{Inh(r, v) \mid v \in \bigcup_{(V,\prec,\sqsubset,l)\in\mathcal{L}} V, l(v) = t\} \cup \{0\})$. Sei hierfür $bgo = (V, \prec, \sqsubset, l) \in \mathcal{L}$, $v \in V$ und $l(v) = t$. Sei weiter $bgo = (V', \prec', \sqsubset', l')$ ein Präfix von v bzgl. bgo. Wir müssen zeigen, dass $m_{bgo'}(r) \leqslant st^-(t)$. Nach Lemma 5.2.2 gibt es eine BGO $bgo_s \in Slin(bgo)$, derart, dass V' ein Präfix bgo'_s von v bzgl. bgo_s definiert. Da bgo in (N_{st}, m_{st}) aktiviert ist, folgt, dass $\sigma(bgo_s) = \tau_1 \dots \tau_n$ in (N_{st}, m_{st}) aktiviert ist. Aus der Abgeschlossenheit von V' bzgl. bgo_s folgt, dass $\sigma(bgo'_s) = \tau_1 \dots \tau_m$, $m < n$, und $l(v) = t \in \tau_{m+1}$. Es genügt nun zu zeigen, dass $m(p_{st}) = m_{bgo'}(r)$ für die Folgemarkierung m von $\sigma(bgo'_s)$, da $m(p_{st}) \leqslant I_{st}(p_{st}, t) = st^-(t)$ für jede Transition $t \in \tau_{m+1}$. Hierzu berechnen wir wie vorher: $m(p_{st}) = m_{st}(p_{st}) + \sum_{i=1}^{m} \sum_{t \in \tau_i} \tau_i(t)(W_{st}(t, p_{st}) - W_{st}(p_{st}, t)) = m_{st}(p_{st}) + \sum_{v \in V'}(W_{st}(l(v), p_{st}) - W_{st}(p_{st}, l(v))) = m_{bgo'}(r)$. □

Somit lässt sich die Menge aller bzgl. einer geschichteten Sprache $\mathcal{L}$ zulässigen Stellenquadrupel aus der Menge aller Markenfluss-Regionen ableiten. Für die Menge der Markenfluss-Regionen einer endlichen geschichteten Sprache $\mathcal{L}$ lässt sich auch hier eine endliche linear algebraische Charakterisierung gewinnen. Diese basiert auf der linear algebraischen Charakterisierung für Markenfluss-Regionen r der $\mathcal{L}$ zugrundeliegenden partiellen Sprache. Zusätzlich muss nur die Restriktion an die Inhibitorgewicht-Funktion I berücksichtigt werden. Diese lässt sich durch endlich viele homogene lineare Ungleichungen mit ganzzahligen Koeffizienten repräsentieren, indem für jede Transition t, jeden Knoten v mit $l(v) = t$ und jedes Präfix bgo' von v die Beziehung $I(t) \geqslant m_{bgo'}(r)$ kodiert wird.

Mit einer solchen Charakterisierung lassen sich dann Verfahren zur Synthese eines STI-Netzes aus einer endlichen geschichteten Sprache ähnlich wie im letzten Kapitel entwickeln. Auch hier ist es möglich, verschiedene Repräsentationen eines gesättigt zulässigen STI-Netzes zu berechnen. Anschließend ist dann wiederum ein Übereinstimmungstest notwendig, um zu prüfen, ob das synthetisierte Netz das spezifizierte Verhalten aufweist. Auf diesem Wege lässt sich dann wie in Kapitel 4 das Syntheseproblem dieses Unterabschnittes lösen.

Es gibt also auch für dieses Problem wiederum entsprechend zu unserem in der Einleitung diskutierten Synthesebaukasten verschiedene Möglichkeiten für die Berechnung von Regionen und für einen Übereinstimmungstest. Außerdem lassen sich neben der Definition von Markenfluss-Regionen natürlich auch für geschichtete Sprachen wie in

Kapitel 4 weitere Regionendefinitionen herleiten. Auf diese Aspekte gehen wir aber hier nicht im Detail ein. Weiter sei noch angemerkt, dass es neben STI-Netzen auch andere Petrinetzklassen gibt, deren Ablaufverhalten sich durch BGOs beschreiben lässt. Die Synthese von Petrinetzen verschiedener Netzklassen wird in Abschnitt 5.3 behandelt. Hierbei werden solche Netzklassen allerdings nicht berücksichtigt. Für eine entsprechende Erweiterung müssten Konzepte dieses Unterabschnittes aufgegriffen werden. Schlussendlich weisen wir noch darauf hin, dass es neben geschichteten Ordnungen auch andere Erweiterungen von partiellen Ordnungen gibt. Ein Beispiel, welches auch im Rahmen von Petrinetzsynthese interessant ist, sind allgemeine gerichtete azyklische Graphen [134].

5.2.2 *Alternative Ablaufsemantiken für partielle Sprachen*

In diesem Unterabschnitt betrachten wir zwar wieder eine partielle Sprache als Eingabe für das Syntheseproblem, diesmal aber unter Verwendung der Minimalablaufsemantik und der Prozessablaufsemantik von S/T-Netzen. Die BPOs modellieren in diesen Fällen Abläufe auf eine andere Art als im Falle der Standard-Ablaufsemantik. Bei der Standard-Ablaufsemantik ist die Interpretation einer BPO derart, dass zwischen geordneten Ereignissen nicht notwendigerweise eine kausale Abhängigkeit besteht. Die Ereignisse müssen nur bei einer Durchführung des Ablaufes in der gegeben Reihenfolge stattfinden. Dies ist auch dann möglich, wenn sie unabhängig, also nebenläufig, zueinander schalten. Ein Paar von ungeordneten Ereignissen des Ablaufs muss dabei natürlich innerhalb einer Durchführung des Ablaufs nebenläufig schalten. Bei der Minimalablaufsemantik gibt es nun einen kleinen aber wesentlichen Unterschied. Zwischen geordneten Ereignissen muss in diesem Fall notwendigerweise eine kausale Abhängigkeit bestehen, d.h. eine Durchführung des Ablaufes darf nur derart möglich sein, dass zwischen allen geordneten Ereignissen auch tatsächlich ein kausaler Zusammenhang besteht, der nur ein Schalten in der gegebenen Reihenfolge zulässt. Ist eine Durchführung des Ablaufs möglich, bei der zwei geordnete Ereignisse unabhängig voneinander schalten, so ist die Ordnung nicht minimal. Auch der Unterschied zur Prozessablaufsemantik ist sehr fein. Hier gilt nun, dass es eine Durchführung des Ablaufes geben muss, bei der zwischen allen geordneten Ereignissen auch tatsächlich ein kausaler Zusammenhang besteht, der nur ein Schalten in der gegebenen Reihenfolge zulässt. Diese Durchführung entspricht dann einem zum Ablauf gehörigen Prozessnetz. Es darf aber auch andere Durchführungen des Ablaufes geben, die diese Forderung nicht erfüllen. Bei der Minimalablaufsemantik muss also die Forderung der Entsprechung von Ordnungsbeziehungen der BPO zu wahren Kausalitäten einer Durchführung des Ablaufes für alle Durchführungen gelten, während für die Prozessablaufsemantik nur die Existenz einer solchen Durchführung gefordert wird.

Beispiel

BEISPIEL: *Wir betrachten hier als Beispiel wieder die partielle Sprache aus Abbildung 4. Zusammen mit allen Präfixen stellt sie die Minimalablaufsemantik des Netzes aus Abbildung 3 dar. Dieses Netz wäre also eine mögliche Lösung im Rahmen eines entsprechenden Syntheseverfahrens. Bzgl. der Prozessablaufsemantik ist dieses Netz allerdings keine Lösung, da es, wie in Abbildung 48 dargestellt, auch eine Prozess-BPO besitzt, welche eine Sequentialisierung von* bpo_2 *ist.*

Die Prozessablaufsemantik des Netzes aus Abbildung 3 ist insgesamt durch eben die drei BPOs aus Abbildung 48 zusammen mit deren Präfixen gegeben. Das Netz stellt also für diese BPOs eine mögliche Lösung für ein Syntheseverfahren bzgl. der Prozessablaufsemantik dar. Da eine der drei BPOs keine minimale Kausalität aufweist, gäbe es dahingegen für diese BPOs bzgl. der Minimalablaufsemantik kein Lösungsnetz.
Betrachten wir den Präfix- und Sequentialisierungsabschluss der zwei in den letzten beiden Absätzen diskutierten partiellen Sprachen, so ist dieser identisch. Daher ist das Netz aus Abbildung 3 bzgl. unseres Standard-Syntheseproblems, welches die Standard-Ablaufsemantik verwendet, für beide Sprachen eine mögliche Lösung.

Wir beginnen nun mit dem Problem der Synthese eines S/T-Netzes, welches die durch eine partielle Sprache gegebene Minimalablaufsemantik aufweist. Das zu synthetisierende Netz soll also genau die spezifizierten Abläufe in dem gerade beschriebenen Sinne der Minimalablaufsemantik erlauben. Es ist zu beachten, dass wir hier eine exakte Spezifikation der Minimalablaufsemantik des gesuchten Netzes als Ausgangspunkt für die Synthese voraussetzen. Eine Vereinfachung der Spezifikation, ähnlich wie wir sie in Kapitel 4 durch die Betrachtung von $\mathrm{PS}(\mathcal{L})$ für eine gegeben Sprache $\mathcal{L}$ zugelassen haben, würde die technischen Ausführungen hier deutlich komplizieren. Dies liegt daran, dass die Minimalablaufsemantik eines S/T-Netzes im Allgemeinen weder präfix- noch sequentialisierungsabgeschlossen ist.

Problemspezifikation 5.2.2 (Syntheseproblem für Minimalablaufsemantik)

PROBLEMSPEZIFIKATION 5.2.2 (SYNTHESEPROBLEM FÜR MINIMALABLAUFSEMANTIK)
Eingabe: *Eine endliche partielle Sprache $\mathcal{L}$.*
Ausgabe: *Ein markiertes S/T-Netz (N, m_0) mit $\mathfrak{MBpo}(N, m_0) = \mathcal{L}$, falls ein solches Lösungsnetz existiert, und eine „*`notexists`*"-Meldung sonst.*

Dieses Problem lässt sich entsprechend folgendem Lemma auf unser übliches Syntheseproblem aus dem letzten Kapitel zurückführen.

Lemma 5.2.3

LEMMA 5.2.3
Sei $\mathcal{L}$ eine partielle Sprache und (N, m_0) ein S/T-Netz. Es gilt genau dann $\mathfrak{MBpo}(N, m_0) = \mathcal{L}$, wenn $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$ und $\mathcal{L} = \mathrm{MinO}(\mathrm{PS}(\mathcal{L}))$ erfüllt ist.

BEWEIS: Falls $\mathfrak{MBpo}(N, m_0) = \mathcal{L}$, dann folgt $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathfrak{MBpo}(N, m_0)) = \mathrm{PS}(\mathcal{L})$. Außerdem gilt dann $\mathcal{L} = \mathfrak{MBpo}(N, m_0) = \mathrm{MinO}(\mathfrak{Bpo}(N, m_0)) = \mathrm{MinO}(\mathrm{PS}(\mathcal{L}))$.
Gelte umgekehrt $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$ und $\mathcal{L} = \mathrm{MinO}(\mathrm{PS}(\mathcal{L}))$. Dann folgt $\mathfrak{MBpo}(N, m_0) = \mathrm{MinO}(\mathfrak{Bpo}(N, m_0)) = \mathrm{MinO}(\mathrm{PS}(\mathcal{L})) = \mathcal{L}$. □

Die Forderung $\mathcal{L} = \mathrm{MinO}(\mathrm{PS}(\mathcal{L}))$ lässt sich für eine gegebene endliche partielle Sprache $\mathcal{L}$ ohne Probleme prüfen. Hierzu müssen ähnlich wie im Rahmen des optimistischen Übereinstimmungstests aus Unterabschnitt 4.5.1 spezielle Graphisomorphie-Probleme für BPOs gelöst werden. Auf ein genaues Vorgehen wird hier nicht näher eingegangen. Falls die Überprüfung ergibt, dass $\mathcal{L} \neq \mathrm{MinO}(\mathrm{PS}(\mathcal{L}))$, so kann entsprechend Lemma 5.2.3 eine negative Antwort auf das betrachtete Syntheseproblem gegeben werden. Falls $\mathcal{L} = \mathrm{MinO}(\mathrm{PS}(\mathcal{L}))$ gilt, so ist entsprechend Lemma 5.2.3 die Forderung $\mathfrak{MBpo}(N, m_0) = \mathcal{L}$ für ein S/T-Netz (N, m_0) äquivalent zur Eigenschaft $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$. Dies ist aber gerade die in unserem Standard-Syntheseproblem aus

Kapitel 4 geforderte Eigenschaft. Damit lässt sich das neue Syntheseproblem in diesem Fall dadurch lösen, dass wir einen Algorithmus zur Lösung des Standard-Syntheseproblems aus Kapitel 4 auf die endliche partielle Sprache $\mathcal{L}$ anwenden. Mit diesem Vorgehen lässt sich also insgesamt das betrachtete Syntheseproblem für die Minimalablaufsemantik lösen.
Im Weiteren diskutieren wir das Problem der Synthese eines S/T-Netzes, welches die durch eine partielle Sprache gegebene Prozessablaufsemantik hat. Die Frage der Entscheidbarkeit eines ähnlichen Problems wurde auch in [180] betrachtet. Die Prozessablaufsemantik eines S/T-Netzes ist immer präfixabgeschlossen, im Allgemeinen aber nicht sequentialisierungsabgeschlossen. Dementsprechend ist es hier sinnvoll, nach einem Lösungsnetz zu fragen, dessen Prozessablaufsemantik dem Präfixabschluss der spezifizierten partiellen Sprache entspricht.

PROBLEMSPEZIFIKATION 5.2.3 (SYNTHESEPROBLEM FÜR PROZESSABLAUFSEMANTIK)
Problemspezifikation 5.2.3 (Syntheseproblem für Prozessablaufsemantik)
Eingabe: *Eine endliche partielle Sprache* $\mathcal{L}$.
Ausgabe: *Ein markiertes S/T-Netz* (N, m_0) *mit* $\mathfrak{PBpo}(N, m_0) = \mathrm{Pref}(\mathcal{L})$, *falls ein solches Lösungsnetz existiert, und eine „*notexists*“-Meldung sonst.*

In dieser Allgemeinheit ist das Problem sehr schwierig. Dies liegt daran, dass die üblichen zulässigen Stellentripel bzgl. $\mathcal{L}$ zwar die einzigen Kandidaten zum Hinzufügen zum Netz sind, dabei aber zwei Probleme bestehen, die sich auf der Ebene der Markenflüsse erklären lassen. Zum einen erlaubt zwar jedes zulässige Stellentripel eine Markenfluss-Verteilung, welche nur die spezifizierten kausalen Abhängigkeiten verwendet, es kann bzgl. desselben Stellentripels aber auch andere Markenfluss-Verteilungen geben, für die das nicht gilt. Ist dies der Fall, so darf das Stellentripel nicht hinzugefügt werden, da sonst Prozess-BPOs mit zu vielen kausalen Abhängigkeiten möglich wären. Dies zu überprüfen, ist aber schwierig. Zum anderen muss für jede in einer BPO spezifizierten kausalen Abhängigkeit ein Stellentripel hinzugefügt werden, welches einen positiven Markenfluss auf der entsprechenden Kante erlaubt. Hierbei ist u.U. für jede Kante ein eigenes Stellentripel nötig, da ein Stellentripel häufig nicht mehrere Kanten gleichzeitig abdeckt. Nur so lässt sich sicherstellen, dass es auch tatsächlich eine Prozess-BPO mit den gegebenen kausalen Abhängigkeiten gibt. Auch hierbei ergeben sich Probleme. Beispielsweise kann es sinnvoll sein, dasselbe Stellentripel mehrfach hinzuzufügen, d.h. für die Prozessablaufsemantik eines Netzes spielt ein mehrfaches Vorkommen strukturell identischer Stellen eine Rolle.
Daher wollen wir hier eine einfachere Version des Problems diskutieren. Es soll ein einssicheres S/T-Netz (N, m_0) mit $\mathfrak{PBpo}(N, m_0) = \mathrm{Pref}(\mathcal{L})$ synthetisiert werden. Es gilt der wichtige, wohlbekannte Zusammenhang, welcher beispielsweise in [131] erklärt wird, dass für einssichere Netze die Prozessablaufsemantik und die Minimalablaufsemantik übereinstimmen. Damit reduziert sich das Problem auf die Synthese eines einssicheren S/T-Netzes (N, m_0) mit $\mathfrak{MBpo}(N, m_0) = \mathrm{Pref}(\mathcal{L})$, im Prinzip also auf das schon gelöste Problem aus Problemspezifikation 5.2.2. Es bleibt nur noch, die Einssicherheit des gesuchten Netzes zu berücksichtigen. Dies bedeutet, dass nur zulässige Stellen zu dem zu synthetisierenden Netz hinzugefügt werden dürfen, welche die Eigenschaft der Einssicherheit bzgl. des spezifizierten Verhaltens garantieren. Auf linear algebraischer Ebene lässt sich eine entsprechende Forderung

nach Einssicherheit für alle Regionendefinitionen durch endlich viele, allerdings inhomogene Ungleichungen ausdrücken. Ein genaues Vorgehen zur Synthese einssicherer Netze wird im Rahmen der „Variante Beschränktheit“ in Unterabschnitt 5.4.7 beschrieben.

5.3 NETZKLASSEN

Der Fokus dieser Arbeit liegt auf der Synthese von Systemmodellen in der Form von Petrinetzen aus halbgeordneten Abläufen. Bei der Betrachtung halbgeordneter Abläufe spielt natürlich Nebenläufigkeit eine zentrale Rolle. Diese lässt sich durch Petrinetze auf der Systemebene in einer natürlichen Art und Weise repräsentieren. Insbesondere lässt sich das komplexe Zusammenspiel zwischen Nebenläufigkeit und Nichtdeterminismus intuitiv darstellen. Daher stellen Petrinetze auf der Systemebene ein sinnvolles Gegenstück zu halbgeordneten Abläufen auf der Szenarioebene dar. Außerdem lassen sich viele anwendungsorientierte Modellierungssprachen für nebenläufige Systeme auf Petrinetze zurückführen. Sie können häufig als Anwendungsdialekte von Petrinetzen betrachtet werden.
Es gibt allerdings nicht das Petrinetz, sondern der Begriff Petrinetz steht heute für viele Klassen von netzbasierten Modellen. Bisher haben wir die Synthese von S/T-Netzen betrachtet. S/T-Netze stellen die klassische Standarddefinition für Petrinetze dar. Diese fundamentale Petrinetzklasse kann als natürliche Erweiterung der ursprünglichen Definitionen von Petri verstanden werden. Sie erlaubt die Modellierung äußerst komplexer kausaler Abhängigkeiten. Neben S/T-Netzen gibt es aber viele weitere Petrinetzklassen, die typischerweise bestimmte für spezielle Bedürfnisse geeignete Abwandlungen der klassischen Petrinetzdefinition darstellen. Sie wollen keinesfalls die klassischen Petrinetze neu definieren, sondern es soll bei den Definitionen spezieller Petrinetzklassen ausgedrückt werden, dass diese Netzklassen sich in das allgemeiner zu betrachtende Konzept des Petrinetzes einordnen lassen. Neu entwickelte Petrinetzklassen erlauben häufig die Modellierung von Verhaltenseigenschaften nebenläufiger Systeme, die mit klassischen Methoden nicht beschreibbar sind. Daneben bieten sie oft intuitivere Modellierungseigenschaften für spezielle Modellierungszwecke. Auch etliche anwendungsorientierte Modellierungssprachen lassen sich nur durch entsprechende spezielle Petrinetzklassen darstellen. Für Anwendungen stellen daher nicht immer S/T-Netze das geeignete Modellierungskonzept dar, sondern häufig sind auch andere Petrinetzklassen von Interesse.
Dementsprechend betrachten wir in diesem Abschnitt die Synthese von Netzen weiterer Petrinetzklassen. Wir werden aber nicht spezielle einzelne Netzklassen diskutieren, wie es in einigen Arbeiten im Rahmen von Petrinetzsynthese gemacht wurde, z.B. [47, 185, 186, 146], sondern wir wollen die Synthese von Petrinetzen aus partiellen Sprachen parametrisch bzgl. der Petrinetzklasse präsentieren. In [16, 17, 18, 68] wurde dieser Weg für die Synthese von Petrinetzen aus (Schritt-) Transitionssystemen beschritten. Hierzu wurden in [16, 17] die schon in Abschnitt 3.2 diskutierten Netztypen eingeführt, welche eine parametrische Definition von Petrinetzen darstellen. Dadurch ließen sich viele bisher bekannte Syntheseresultate für Transitionssysteme als spezielle Instanzen der neuen allgemeinen Resultate wiederfinden.

Wir entwickeln in diesem Abschnitt ein ähnliches Vorgehen für partielle Sprachen. Wir verallgemeinern also die Syntheseprinzipien aus Kapitel 4 auf das Problem der Synthese eines Netzes eines beliebig vorgegebenen Netztyps aus einer partiellen Sprache. Netztypen stellen eine für ein solches Vorgehen sehr geeignete parametrische Petrinetzdefinition dar. Der lokale Charakter der auf Stellen und Transitionen basierenden Definition erlaubt insbesondere eine Verallgemeinerung von Lemma 4.3.1, also dem Prinzip, dass das Verhalten eines Netzes Schritt für Schritt durch das Hinzufügen von Stellen eingeschränkt werden kann. Damit kann für die Synthese im Rahmen von Netztypen größtenteils analog vorgegangen werden wie im Falle von S/T-Netzen im letzten Kapitel.
Für andere parametrische Petrinetzdefinitionen wie algebraische $(\mathcal{M},\mathcal{I})$-Netze [75, 131], Petrinetze über einer Gruppe [86] oder Marken-freie Netze [168] gestaltet sich ein Synthesevorgehen wesentlich komplizierter. Die Definition der Netztypen garantiert das für Regionendefinitionen wichtige Lokalitätsprinzip für Petrinetze auf eine natürliche Art und Weise. Damit stellt sie, wie in [16, 17, 18] dargelegt, eine geeignete einheitliche Petrinetzdefinition für Synthesezwecke dar. In [18] wird diskutiert, dass die Netzdefinition aus [86] in diesem Kontext problematisch ist. Algebraische Definitionen wie beispielsweise [75, 131] haben typischerweise keinen lokalen Fokus. Daher ist die Entwicklung entsprechender Regionenkonzepte für algebraische Netze kompliziert. Das Konzept der Marken-freien Netze [168] ist schließlich sehr ähnlich zu der Idee der Netztypen. Allerdings lassen sich Regionen für Netztypen leichter definieren.
Obwohl wir also grundsätzlich das Synthesevorgehen aus Kapitel 4 auf Netztypen übertragen können, ergibt sich noch ein wesentliches Problem. Es gibt bisher noch keine Definition einer Ablaufsemantik für Netztypen. Die Regionendefinitionen aus Kapitel 4 basieren aber wesentlich auf den verschiedenen Charakterisierungen aktivierter BPOs. Die Schritt-Transitions-Regionen verwenden die ursprüngliche Definition von Aktiviertheit aus Definition 3.3.8. BPO-Transitions-Regionen basieren auf der äquivalenten Charakterisierung aus Lemma 3.3.1. Markenfluss-Regionen verwenden schließlich das Prinzip der Markenfluss-BPOs. Wir verallgemeinern hier diese Charakterisierungen auf Netztypen. Allerdings sind hierzu gewisse Einschränkungen der allgemeinen Definition von Netztypen notwendig. Wir diskutieren diese und zeigen, dass sie keine wichtigen bekannten Netzklassen ausschließen. Diese Überlegungen geben insbesondere sehr interessante Einblicke über die Grenzen im Rahmen der Modellierung halbgeordneter Abläufe nebenläufiger Systeme durch BPOs.
Mit den verschiedenen Charakterisierungen aktivierter BPOs lassen sich dann völlig analog zum Fall von S/T-Netzen entsprechende Regionendefinitionen für Netztypen formulieren. Darauf aufbauend lassen sich dann für konkrete Netztypen, wie im letzten Kapitel dargestellt, Syntheseverfahren entwickeln. Der wesentliche Schritt zur in der Netzklasse parametrischen Betrachtung des Syntheseproblems für partielle Sprachen in diesem Abschnitt ist also die Einführung geeigneter Ablaufdefinitionen. Wie sich darauf aufbauend Regionendefinitionen und Syntheseverfahren analog zu Kapitel 4 entwickeln lassen, soll hier nur skizziert werden.
Wir beginnen in Unterabschnitt 5.3.1, die klassischen Aktiviertheitsdefinitionen von BPOs für S/T-Netze auf Netztypen zu verallgemeinern. In

Unterabschnitt 5.3.2 soll das Konzept der Markenflüsse auf Netztypen übertragen werden. Im letzten Unterabschnitt wird dann kurz gezeigt, wie sich diese Ergebnisse für die Synthese von Netzen eines Netztyps aus einer partiellen Sprache verwenden lassen.

5.3.1 *Aktivierte BPOs für Netztypen*

In diesem Unterabschnitt wird zum ersten mal eine Halbordnungssemantik für Netztypen eingeführt. Wir übertragen die verschiedenen klassischen Aktiviertheitsdefinitionen von BPOs für S/T-Netze auf Netztypen. Hierbei zeigt sich, dass bestimmte Charakterisierungen nur unter geeigneten Einschränkungen der sehr allgemeinen Netztypen-Definition sinnvoll möglich sind. Wir beginnen mit der Übertragung des ursprünglichen Aktiviertheitsbegriffs aus Definition 3.3.8 auf Netztypen. Dieser lässt sich analog für Netztypen formulieren. Die Interpretation, inwiefern eine derart aktivierte BPO gültiges Ablaufverhalten des Netzes darstellt, ist identisch wie im Falle von S/T-Netzen.

Definition 5.3.1 (Aktiviertheit)

Definition 5.3.1 (Aktiviertheit)
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes Netz vom Netztyp* $\mathcal{N} = (LS, LE, \mapsto)$. *Eine BPO* $bpo = (V, <, l)$ *mit* $l : V \to T$ *heißt aktiviert in* m_0 *bzgl.* N, *falls für jede Schrittlinearisierung* $bpo' \in Slin(bpo)$, *die assoziierte Schrittfolge* $\sigma(bpo')$ *in* m_0 *aktiviert ist.*

Eine Markierung m', *welche durch* $m_0 \xrightarrow{\sigma(bpo')} m'$ *für eine BPO* $bpo' \in Slin(bpo)$ *gegeben ist, heißt Folgemarkierung von* bpo.

Definition 5.3.2 (Ablaufsemantik)

Definition 5.3.2 (Ablaufsemantik)
Die partielle Sprache $\mathfrak{Bpo}(N, m_0)$ *aller in* m_0 *bzgl.* N *aktivierten BPOs heißt (Standard-) Ablaufsemantik eines markierten Netzes* (N, m_0) *vom Netztyp* $\mathcal{N} = (LS, LE, \mapsto)$.

Es lässt sich leicht zeigen, dass jede Sequentialisierung und jedes Präfix einer aktivierten BPO aktiviert ist. Außerdem ist zu beachten, dass verschiedene Schrittfolgen $\sigma(bpo')$ verschiedene Folgemarkierungen der BPO erzeugen können, d.h. eine BPO hat im Allgemeinen mehrere Folgemarkierungen.

Beispiel

Beispiel: *Als Beispiel hierzu betrachten wir den Netztyp* $(\{s, s'\}, \{0, 1, 2\}, \{s \xrightarrow{1} s, s \xrightarrow{2} s', s' \xrightarrow{1} s, s \xrightarrow{0} s, s' \xrightarrow{0} s'\})$, *wobei* 0 *das neutrale Element des Monoids von lokalen Ereignissen ist und die Operation des Monoids durch* $1 + 1 = 2$, $1 + 2 = 0$ *und* $2 + 2 = 1$ *gegeben ist. Weiter sei ein markiertes Netz* (N, m_0), $N = (P, T, W)$, *dieses Typs durch* $P = \{p\}$, $T = \{t_1, t_2\}$, $m_0(p) = s$, $W(p, t_1) = 1$, $W(p, t_2) = 2$ *gegeben. Dann ist die BPO* $bpo = (\{v_1, v_2\}, \emptyset, l)$, $l(v_i) = t_i$ *für* $i = 1, 2$ *bzgl.* (N, m_0) *aktiviert. Die Schrittfolge* $t_1 t_2$ *erzeugt die durch* $m'(p) = s'$ *definierte Folgemarkierung* m' *von* bpo, *während die Schrittfolge* $t_2 t_1$ *die durch* $m(p) = s \neq s'$ *definierte Folgemarkierung* m *von* bpo *erzeugt.*

Im Falle von S/T-Netzen kann die Aktiviertheit von BPOs, wie in Lemma 3.3.1 beschrieben, äquivalent durch die Betrachtung von Schnitten definiert werden. Eine Vorstufe bei der Herleitung dieses Lemmas stellt die Betrachtung von Co-Mengen dar. Es lässt sich nämlich unmittelbar zeigen, dass eine Abwandlung der Aussage von Lemma 3.3.1, bei der alle Co-Mengen anstelle nur der Schnitte betrachtet werden, gilt. Damit lässt sich dann in einem nächsten Schritt das eigentliche Lemma 3.3.1 beweisen. In dieser Reihenfolge gehen wir auch hier vor, d.h. das Ziel

ist zunächst, die Aktiviertheit einer BPO bzgl. eines gegebenen Netzes eines Netztyps dadurch zu charakterisieren, dass für jede Co-Menge C der BPO und jedes Präfix der Co-Menge die Folgemarkierung des Präfixes den Transitionsschritt $|C|_l$ aktiviert. Hierbei ergeben sich allerdings zwei Probleme. Zuerst einmal ist die Folgemarkierung eines Präfixes nicht eindeutig. Daher müssen alle möglichen Folgemarkierungen eines Präfixes betrachtet werden. Zweitens lässt sich eine Folgemarkierung nur dann definieren, wenn das Präfix selbst aktiviert ist. Bei S/T-Netzen konnten wir unabhängig davon, ob ein Präfix tatsächlich aktiviert ist, die Folgemarkierung eines Präfixes mithilfe der Ereignisse des Präfixes definieren. Bei Netztypen muss die Aktiviertheit entsprechender Präfixe nun vorausgesetzt werden.

LEMMA 5.3.1 *Lemma 5.3.1*

Sei (N, m_0), $N = (P, T, W)$, *ein markiertes Netz vom Netztyp* $\mathcal{N} = (LS, LE, \mapsto)$. *Eine BPO* $bpo = (V, <, l)$ *mit* $l : V \to T$ *ist genau dann bzgl.* (N, m_0) *aktiviert, wenn für jede nicht-leere Co-Menge* C *von* bpo *und jedes Präfix* $bpo' = (V', <', l')$ *von* C *gilt, dass* bpo' *aktiviert ist und* $|C|_l$ *in jeder Folgemarkierung von* bpo' *aktiviert ist.*

BEWEIS: Sei bpo aktiviert, dann ist auch das Präfix bpo' aktiviert. Sei $bpo'' = (V', <'', l') \in Slin(bpo')$, dann gibt es $bpo_s = (V, <_s, l) \in Slin(bpo)$ derart, dass $v' <_s c <_s v$ und $c \operatorname{co}_{<_s} c'$ für alle $v' \in V', c, c' \in C, v \in V \setminus (V' \cup C)$ und $<''=<_s|_{V' \times V'}$ gilt. Aus der Aktiviertheit von bpo folgt, dass die Schrittfolge $\sigma(bpo_s)$ aktiviert ist. Dies zeigt, dass $|C|_l$ in der durch bpo'' gegebenen Folgemarkierung von bpo' aktiviert ist.
Falls bpo nicht aktiviert ist, dann ist entweder jedes echte Präfix von bpo aktiviert oder dies ist nicht der Fall. Im zweiten Fall gibt es eine nicht-leere Co-Menge von bpo, welche ein nicht-aktiviertes Präfix besitzt. Im ersten Fall betrachten wir $bpo_s \in Slin(bpo)$, derart, dass $\sigma(bpo_s) = \tau_1 \dots \tau_n$ $(\tau_n \neq \emptyset)$ nicht aktiviert ist. Da jedes echte Präfix von BPO aktiviert ist, folgt, dass $\tau_1 \dots \tau_{n-1}$ aktiviert ist und τ_n in der Folgemarkierung von $\tau_1 \dots \tau_{n-1}$ nicht aktiviert ist. Sei C die nicht-leere Menge der maximalen Knoten von bpo_s. Diese ist eine Co-Menge von bpo. Sei weiter bpo' das von $V \setminus C$ definierte Präfix von bpo. Dieses ist ein Präfix von C. Dann ist die BPO bpo' aktiviert, da sie ein echtes Präfix von bpo ist. Allerdings ist $|C|_l = \tau_n$ in der durch $\tau_1 \dots \tau_{n-1}$ gegebenen Folgemarkierung von bpo' nicht aktiviert. □

Die zwei angesprochenen Probleme machen, wie dieses Lemma zeigt, den Aktiviertheitsbegriff sehr kompliziert. Um diesen zu vereinfachen, schränken wir die Definition eines Netztyps geeignet ein. Das Hauptproblem stellen die verschiedenen möglichen Folgemarkierungen eines Präfixes dar. Wir sind daher an Netzen interessiert, dessen aktivierte BPOs immer eine eindeutige Folgemarkierung besitzen. Wir formulieren eine schöne und natürliche Eigenschaft von Netzen, welche dies sicherstellt. Diese nennen wir schwache Zwischen-Zustands-Eigenschaft (SZZE). Ein markiertes Netz (N, m_0), $N = (P, T, W)$, eines Netztyps erfüllt die SZZE, wenn für alle erreichbaren Markierungen m, m', m'' und Transitionsschritte τ_1, τ_2 die folgende Beziehung gilt.

$$m \xrightarrow{\tau_1+\tau_2} m' \wedge m \xrightarrow{\tau_1\tau_2} m'' \implies m' = m''$$

Wenn also ein aktivierter Schritt in eine Folge von zwei Schritten, welche auch aktiviert ist, zerlegt werden kann, so führt diese Folge zur selben Markierung wie der Schritt.

Lemma 5.3.2

LEMMA 5.3.2
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes Netz vom Netztyp* $\mathcal{N} = (LS, LE, \mapsto)$, *welches die SZZE erfüllt. Dann hat jede bzgl.* (N, m_0) *aktivierte BPO* $bpo = (V, <, l)$ *eine eindeutige Folgemarkierung.*

BEWEIS: Wir betrachten $bpo', bpo'' \in Slin(bpo)$ mit $m_0 \xrightarrow{\sigma(bpo')} m'$ und $m_0 \xrightarrow{\sigma(bpo'')} m''$. Es ist $m' = m''$ zu zeigen.
Wir betrachten eine feste Reihenfolge der Knoten $V = \{v_1, \ldots, v_n\}$, derart, dass aus $v_i < v_j$ die Beziehung $i < j$ folgt. Wir zeigen, dass sowohl m' als auch m'' mit der Markierung m''' übereinstimmen, welche durch $m_0 \xrightarrow{l(v_1)\ldots l(v_n)} m'''$ gegeben ist. Hierzu formen wir $\sigma(bpo')$ iterativ in die Schaltfolge $l(v_1)\ldots l(v_n)$ um. Jede Iteration führt zu einer Schrittfolge σ von bpo. Diese ist aktiviert, da bpo aktiviert ist. Wir zeigen jeweils, dass sie $m_0 \xrightarrow{\sigma} m'$ erfüllt.
Zuerst transformieren wir $\sigma(bpo')$ zu einer Schaltfolge. Sei $\sigma(bpo') = \tau_1 \ldots \tau_k$. Falls ein τ_i nicht einelementig ist, dann kann τ_i in zwei nichtleere Schritte τ', τ'' zerlegt werden, i.e. $\tau_i = \tau' + \tau''$. Sei $m_0 \xrightarrow{\tau_1\ldots\tau_{i-1}} m_{i-1}$ und $m_{i-1} \xrightarrow{\tau_i} m_i$. Entsprechend der SZZE gilt $m_{i-1} \xrightarrow{\tau'\tau''} m_i$ (die Aktiviertheit von $\tau'\tau''$ wird durch die Aktiviertheit von bpo sichergestellt). Wir zerlegen alle Schritte auf diese Weise bis wir eine Schaltfolge $t_1 \ldots t_n$ von bpo mit $m_0 \xrightarrow{t_1\ldots t_n} m'$ erhalten.
Sei $bpo_{lin} = (V, <_{lin}, l)$ die Linearisierung von bpo mit $\sigma(bpo_{lin}) = t_1 \ldots t_n$. In bpo_{lin} sind die Knoten $\{v_1, \ldots, v_n\}$ derart total geordnet, dass sie die partielle Ordnungsbeziehung $<$ respektieren. Die Ordnung muss aber nicht der durch die Indizes $1, \ldots, n$ gegebenen Reihenfolge entsprechen. Es kann also vorkommen, dass $v_i <_{lin} v_j$ und $i > j$. Dies ist aber nur dann möglich, wenn $v_i \not< v_j$ und $v_j \not< v_i$, i.e. $v_i \,co_< v_j$. In einer solchen Situation vertauschen wir die Positionen der Knoten v_i und v_j in bpo_{lin}, um schließlich zu einer Linearisierung von bpo mit der Ordnung $v_1 \to v_2 \to \ldots \to v_n$ zu gelangen. Die unterschiedlichen Positionen der entsprechenden Knoten in den zwei Linearisierungen $v_1 \to v_2 \to \ldots \to v_n$ und bpo_{lin} können durch die Permutation π, für die gilt, dass $\pi(i)$ die Position des i-ten Knotens von bpo_{lin} in $v_1 \to v_2 \to \ldots \to v_n$ angibt, in Beziehung gesetzt werden, i.e. $\pi(i)$ ist der Index des i-ten Knotens von bpo_{lin} ($\pi^{-1}(i)$ ist also die Position von v_i in bpo_{lin}).
Sei also π die Permutation von $\{1, \ldots, n\}$, welche $v_{\pi(i)} <_{lin} v_{\pi(j)} \iff i < j$ erfüllt. Falls $\pi(i) = i$ für alle $i \in \{1, \ldots, n\}$, so sind wir fertig. Ansonsten betrachten wir den ersten Index i mit $\pi(i) \neq i$ (offensichtlich gilt $\pi^{-1}(i) > i$). Die Idee ist, eine Art „Bubble-Sort"-Verfahren anzuwenden, um den Knoten v_i von der Position $\pi^{-1}(i)$ zurück zur Position i zu sortieren und dies zu wiederholen bis es kein solches i mehr gibt. Da i der erste Index mit $\pi(i) \neq i$ ist, ergibt sich $j = \pi(\pi^{-1}(i) - 1) > i = \pi(\pi^{-1}(i))$ und damit $v_j \not< v_i$. Da $v_j <_{lin} v_i$ und bpo_{lin} eine Linearisierung von bpo ist, folgt $v_i \not< v_j$. Es gilt also $v_i \,co_< v_j$. Daher führt das Weglassen der Beziehung $v_j <_{lin} v_i$ von $<_{lin}$ zu einer Schrittlinearisierung von bpo. Die assoziierte Schrittfolge $t_1 \ldots t_{\pi^{-1}(i)-2}(t_{\pi^{-1}(i)-1} + t_{\pi^{-1}(i)}) \ldots t_n$ ist also aktiviert. Außerdem folgt aus der SZZE, dass $m_0 \xrightarrow{t_1\ldots t_{\pi^{-1}(i)-2}} \tilde{m}$, $\tilde{m} \xrightarrow{t_{\pi^{-1}(i)-1} t_{\pi^{-1}(i)}} \tilde{m}'$ und $\tilde{m} \xrightarrow{t_{\pi^{-1}(i)-1} + t_{\pi^{-1}(i)}} \tilde{m}'$. Somit wird die Folgemarkierung m' durch die Schrittfolge $t_1 \ldots t_{\pi^{-1}(i)-2}(t_{\pi^{-1}(i)-1} + t_{\pi^{-1}(i)}) \ldots t_n$ erhalten. Wir

können nun weiter eine Beziehung $v_i <_{lin} v_j$ zu $<_{lin}$ hinzufügen. Dies führt wiederum zu einer Linearisierung von bpo. Die assoziierte Schaltfolge $t_1 \dots t_{\pi^{-1}(i)-2} t_{\pi^{-1}(i)} t_{\pi^{-1}(i)-1} \dots t_n$ ist also wieder aktiviert. Mit der SZZE erhalten wir wiederum $\tilde{m} \xrightarrow{t_{\pi^{-1}(i)} t_{\pi^{-1}(i)-1}} \tilde{m}'$. Somit haben wir durch eine „Bubble-Sort"-Vertauschung v_i um eine Position nach vorne sortiert, ohne die Folgemarkierung der assoziierten Schrittfolge zu verändern. Durch wiederholtes Anwenden dieser Prozedur wird jeder Knoten v_i an die Position i sortiert. Dies zeigt $m_0 \xrightarrow{l(v_1)\dots l(v_n)} m'$. Da dasselbe Vorgehen für bpo'' durchgeführt werden kann, erhalten wir auch $m_0 \xrightarrow{l(v_1)\dots l(v_n)} m''$. Somit folgt $m' = m''$. □

Es lässt sich beobachten, dass es für ein markiertes Netz, welches die SZZE nicht erfüllt, immer eine BPO und zwei aktivierte Schrittfolgen der BPO gibt, so dass die zwei Schrittfolgen unterschiedliche Folgemarkierungen haben. Dies ist zwar auch für Netze, welche die SZZE erfüllen, möglich. In diesem Fall kommen dafür aber nur nicht-aktivierte BPOs in Frage. Unter Voraussetzung der SZZE lässt sich die Charakterisierung von aktivierten BPOs mithilfe von Co-Mengen nun folgendermaßen formulieren.

LEMMA 5.3.3 *Lemma 5.3.3*

Sei (N, m_0), $N = (P, T, W)$, *ein markiertes Netz vom Netztyp* $\mathcal{N} = (LS, LE, \mapsto)$, *welches die SZZE erfüllt, und sei* bpo $= (V, <, l)$ *mit* $l : V \to T$ *eine BPO.*
Falls bpo *bzgl.* (N, m_0) *aktiviert ist, dann folgt für jede nicht-leere Co-Menge* C *von* bpo *und jedes Präfix* bpo$' = (V', <', l')$ *von* C, *dass jede Schrittfolge* σ *von* bpo$'$ *aktiviert ist und* $|C|_l$ *in der Folgemarkierung von* σ *aktiviert ist.*
Falls es für jede nicht-leere Co-Menge C *von* bpo *und jedes Präfix* bpo$' = (V', <', l')$ *von* C *eine aktivierte Schrittfolge* σ *von* bpo$'$ *gibt, so dass* $|C|_l$ *in der Folgemarkierung von* σ *aktiviert ist, dann ist* bpo *bzgl.* (N, m_0) *aktiviert.*

BEWEIS: Die erste Aussage folgt direkt aus Lemma 5.3.1. Für die zweite Aussage betrachten wir eine BPO bpo, welche nicht bzgl. (N, m_0) aktiviert ist. Sei bpo$_p = (V_p, <_p, l_p)$ ein Präfix von bpo, welches nicht aktiviert ist und minimal mit dieser Eigenschaft ist, i.e. jedes echte Präfix von bpo$_p$ ist aktiviert. Nach Lemma 5.3.1 gibt es eine nicht-leere Co-Menge C von bpo$_p$ und ein Präfix bpo$' = (V', <', l')$ von C (bzgl. bpo$_p$ und bpo) derart, dass entweder bpo$'$ nicht aktiviert ist oder $|C|_l$ in einer Folgemarkierung von bpo$'$ nicht aktiviert ist. Da bpo$'$ ein echtes Präfix von bpo$_p$ ist, kann nur der zweite Fall zutreffen. Nach Lemma 5.3.2 ist die Folgemarkierung von bpo$'$ eindeutig, i.e. jede Schrittfolge σ von bpo$'$ ist aktiviert, aber $|C|_l$ ist in der Folgemarkierung von σ jeweils nicht aktiviert. Dies zeigt die Behauptung. □

Es muss hier also nur noch eine Schrittfolge für jedes Präfix einer Co-Menge betrachtet werden, da durch die SZZE die Eindeutigkeit der Folgemarkierung von aktivierten BPOs garantiert wird. Folgende Überlegung zeigt, dass ansonsten die zweite Aussage von Lemma 5.3.3 nicht gilt. Wir betrachten eine BPO bpo $= (V, <, l)$, welche bzgl. eines Netzes eines Netztyps aktiviert ist und zwei unterschiedliche Folgemarkierungen besitzt, wobei eine davon eine Transition t aktiviert, die andere aber nicht. Die zweite Aussage von Lemma 5.3.3 würde dann implizieren, dass die BPO bpo$' = (V \cup \{v\}, < \cup (V \times \{v\}), l')$, $l'|_V = l, l'(v) = t$, aktiviert ist, obwohl sie es tatsächlich gar nicht ist.

Beispiel

BEISPIEL: *Als eine konkrete Beispiel-BPO für dieses Phänomen lässt sich die im letzten Beispiel diskutierte BPO verwenden. Die Folgemarkierung* m *aktiviert hier* t_2*, aber die Folgemarkierung* m' *aktiviert* t_2 *nicht.*

Wir wollen nun eine Anforderung an Netztypen herleiten, welche die SZZE für alle Netze des Typs garantiert. Wir formulieren also die SZZE für Netztypen. Ein Netztyp $\mathcal{N} = (LS, LE, \mapsto)$ erfüllt die SZZE, wenn für alle Kombinationen von lokalen Zuständen $s, s', s'' \in LS$ und lokale Ereignisse $e_1, e_2 \in LE$ die folgende Beziehung gilt.

$$s \xrightarrow{e_1+e_2} s' \wedge s \xrightarrow{e_1 e_2} s'' \implies s' = s''$$

Lemma 5.3.4

LEMMA 5.3.4
Sei $\mathcal{N} = (LS, LE, \mapsto)$ *ein Netztyp. Falls* $\mathcal{N}$ *die SZZE erfüllt, dann erfüllt jedes markierte Netz des Typs* $\mathcal{N}$ *die SZZE. Falls* $\mathcal{N}$ *die SZZE nicht erfüllt, dann gibt es ein markierte Netz des Typs* $\mathcal{N}$*, welches die SZZE nicht erfüllt.*

BEWEIS: Wir beginnen mit dem Falle, dass $\mathcal{N}$ die SZZE erfüllt. Sei (N, m_0), $N = (P, T, W)$, ein markiertes Netz vom Netztyp $\mathcal{N}$. Seien m, m', m'' erreichbare Markierungen und τ_1, τ_2 Transitionsschritte, so dass $m \xrightarrow{\tau_1+\tau_2} m'$ und $m \xrightarrow{\tau_1 \tau_2} m''$. Nach der Schaltregel für Netztypen gelten für jede Stelle $p \in P$ die Beziehungen $m(p) \xrightarrow{\sum_{t\in T}(\tau_1+\tau_2)(t)W(p,t)} m'(p)$ und $m(p) \xrightarrow{\sum_{t\in T}\tau_1(t)W(p,t)} m_{mid}(p) \xrightarrow{\sum_{t\in T}\tau_2(t)W(p,t)} m''(p)$. Wir bezeichnen $e_i = \sum_{t\in T}\tau_i(t)W(p,t)$ für $i = 1, 2$, $s = m(p)$, $s' = m'(p)$ und $s'' = m''(p)$. Dann erhalten wir mit der SZZE $m'(p) = s' = s'' = m''(p)$.
Falls $\mathcal{N}$ die SZZE nicht erfüllt, dann gibt es lokale Zustände $s, s', s'' \in LS$ und lokale Ereignisse $e_1, e_2 \in LE$, so dass $s \xrightarrow{e_1+e_2} s'$, $s \xrightarrow{e_1 e_2} s''$ und $s' \neq s''$. Wir konstruieren ein Netz (N, m_0), $N = (P, T, W)$, vom Netztyp $\mathcal{N}$, welches nicht die SZZE erfüllt, indem wir $P = \{p\}$, $T = \{t_1, t_2\}$, $m_0(p) = s$, $W(p, t_1) = e_1$, $W(p, t_2) = e_2$ setzen. □

Die SZZE wird von allen üblichen Netzklassen erfüllt, d.h. für Standard-Netztypen, wie die in Abschnitt 3.2 diskutierten Beispiele, gilt die SZZE.

Beispiel

BEISPIEL: *Der Netztyp der S/T-Netze* $\mathcal{N}_{st}$*, der Netztyp der elementaren Netze* $\mathcal{N}_{en}$*, der Netztyp der STI-Netze unter der a-posteriori Semantik* $\mathcal{N}_{sti}$ *und der Netztyp der STI-Netze unter der a-priori Semantik* $\mathcal{N}_{\overleftarrow{sti}}$ *erfüllen die SZZE.*

Wir betrachten nun den Fall von zwei verschiedenen aktivierte BPOs, deren Knotenmengen dieselbe Multimenge an Transitionen definieren. Obwohl jede dieser BPOs eine eindeutige Folgemarkierung besitzt, falls SZZE gilt, so müssen diese für die zwei BPOs nicht übereinstimmen.

Beispiel

BEISPIEL: *Wir zeigen dies an dem Netztyp* $(\{s, s', s''\}, \{0, a, b, c\}, \{s \xrightarrow{a} s', s \xrightarrow{b} s'', s' \xrightarrow{b} s'', s'' \xrightarrow{a} s', s \xrightarrow{0} s, s' \xrightarrow{0} s', s'' \xrightarrow{0} s''\})$*, wobei* 0 *das neutrale Element des Monoids von lokalen Ereignissen ist und die Operation des Monoids durch* $a + a = a + b = a + c = b + b = b + c = c + c = c$ *gegeben ist. Dieser Netztyp erfüllt die SZZE. Wir betrachten das Netz* (N, m_0), $N = (P, T, W)$*, dieses Typs, welches durch* $P = \{p\}$, $T = \{t_a, t_b\}$, $m_0(p) = s$, $W(p, t_a) = a$, $W(p, t_b) = b$ *gegeben ist, und die BPOs* $bpo = (\{v_a, v_b\}, \{(v_a, v_b)\}, l)$, $l(v_a) = t_a$, $l(v_b) = t_b$ *und* $bpo' = (\{v'_a, v'_b\}, \{(v'_b, v'_a)\}, l')$, $l'(v'_a) = t_a$, $l'(v'_b) = t_b$*. Beide BPOs sind bzgl.* (N, m_0) *aktiviert. Weiter bestehen beide BPOs aus einem Ereignis der Transition* t_a *und einem Ereignis der Transition* t_b*, aber die Folgemarkierung*

m *von* bpo *ist durch* $m(p) = s''$ *gegeben, während die Folgemarkierung* m' *von* bpo$'$ *durch* $m'(p) = s'$ *gegeben ist.*

Wir schränken nun die SZZE derart weiter ein, dass die Folgemarkierung einer aktivierten BPO nur von der durch die Knotenmenge der BPO definierten Multimenge an Transitionen abhängt, aber nicht mehr von der Ordnungsrelation der BPO. Dies vereinfacht das Testen auf Aktiviertheit von BPOs, da dann in Lemma 5.3.3 wie für S/T-Netze nur die Transitionsvorkommen in bpo$'$ betrachtet werden müssen, nicht mehr aber Schrittfolgen von bpo$'$. Die entsprechend strengerer Eigenschaft bezeichnen wir als Parikh-Bild-Eigenschaft (PBE). Die PBE kann wiederum für Netze und Netztypen formuliert werden.
Ein markiertes Netz (N, m_0), $N = (P, T, W)$, eines Netztyps erfüllt die PBE, wenn für alle erreichbaren Markierungen m, m', m'' zusammen mit Transitionsschritten $\tau_1, \ldots, \tau_n$ und $\tau'_1, \ldots, \tau'_m$ die folgende Beziehung gilt.

$$m \xrightarrow{\tau_1 \ldots \tau_n} m' \wedge m \xrightarrow{\tau'_1 \ldots \tau'_m} m'' \wedge \tau_1 + \ldots + \tau_n = \tau'_1 + \ldots + \tau'_m$$
$$\Longrightarrow m' = m''.$$

Die PBE ist eine für Petrinetze typische Eigenschaft. Sie sagt aus, dass die Folgemarkierung einer aktivierten Schrittfolge durch die Anzahl des Vorkommens aller Transitionen in der Schrittfolge bestimmt ist. In anderen Worten hängt die Folgemarkierung nur von dem „Parikh-Bild" der Schrittfolge ab.
Ein Netztyp $\mathcal{N} = (LS, LE, \mapsto)$ erfüllt die PBE, wenn für alle Kombinationen von lokalen Zuständen $s, s', s'' \in LS$, eine Multimenge von lokalen Ereignissen $u \in \mathbb{N}^{LE}$ und zwei Partitionen $u = u_1 + \ldots + u_n = u'_1 + \ldots + u'_m$ von u mit den Bezeichnungen $e_i = \sum_{e \in u_i} u_i(e)e$ und $e'_i = \sum_{e \in u'_i} u'_i(e)e$ die folgende Beziehung gilt.

$$s \xrightarrow{e_1 \ldots e_n} s' \wedge s \xrightarrow{e'_1 \ldots e'_m} s'' \Longrightarrow s' = s''.$$

Es ist zu beachten, dass sowohl im Falle von Netzen als auch im Falle von Netztypen aus der PBE die SZZE folgt. Damit stellt die PBE tatsächlich eine strengere Forderung dar. Wir zeigen im nächsten Lemma den Zusammenhang der PBE für Netztypen und Netze. Im darauf folgenden Lemma beweisen wir dann das gewünschte Resultat für die PBE, dass die Folgemarkierung einer aktivierten BPO nur von den Transitionsanzahlen in der BPO abhängt.

LEMMA 5.3.5 *Lemma 5.3.5*
Sei $\mathcal{N} = (LS, LE, \mapsto)$ *ein Netztyp. Falls* $\mathcal{N}$ *die PBE erfüllt, dann erfüllt jedes markierte Netz des Typs* $\mathcal{N}$ *die PBE. Falls* $\mathcal{N}$ *die PBE nicht erfüllt, dann gibt es ein markierte Netz des Typs* $\mathcal{N}$*, welches die PBE nicht erfüllt.*

BEWEIS: Wir beginnen mit dem Falle, dass $\mathcal{N}$ die PBE erfüllt. Sei (N, m_0), $N = (P, T, W)$, ein markiertes Netz vom Netztyp $\mathcal{N}$. Seien m, m', m'' erreichbare Markierungen und $\tau_1, \ldots, \tau_n$ sowie $\tau'_1, \ldots, \tau'_m$ Transitionsschritte, so dass $m \xrightarrow{\tau_1 \ldots \tau_n} m'$, $m \xrightarrow{\tau'_1 \ldots \tau'_m} m''$ und $\tau_1 + \ldots + \tau_n = \tau'_1 + \ldots + \tau'_m$. Wir betrachten eine Stelle p und bezeichnen $e_i = \sum_{t \in T} \tau_i(t) \cdot W(p,t)$ und $e'_i = \sum_{t \in T} \tau'_i(t) W(p,t)$, $s = m(p)$, $s' = m'(p)$ und $s'' = m''(p)$. Nach der Schaltregel für Netztypen gelten die Beziehungen $s \xrightarrow{e_1 \ldots e_n} s'$ und $s \xrightarrow{e'_1 \ldots e'_m} s''$. Da $\mathcal{N}$ die PBE erfüllt, erhalten wir $m'(p) = s' = s'' = m''(p)$.

Falls $\mathcal{N}$ die PBE nicht erfüllt, dann gibt es lokale Zustände $s, s', s'' \in LS$, eine Multimenge von lokalen Ereignissen $u \in \mathbb{N}^{LE}$ und zwei Partitionen $u = u_1 + \ldots + u_n = u'_1 + \ldots + u'_m$ von u, so dass $s \xrightarrow{e_1 \ldots e_n} s'$, $s \xrightarrow{e'_1 \ldots e'_m} s''$ und $s' \neq s''$, wobei $e_i = \sum_{e \in u_i} u_i(e)e$ und $e'_i = \sum_{e \in u'_i} u'_i(e)e$. Wir konstruieren ein Netz (N, m_0), $N = (P, T, W)$, vom Netztyp $\mathcal{N}$, welches nicht die PBE erfüllt, indem wir $P = \{p\}$, $T = LE$, $m_0(p) = s$, $W(p, e) = e$ setzen. □

Lemma 5.3.6

LEMMA 5.3.6
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes Netz vom Netztyp* $\mathcal{N} = (LS, LE, \mapsto)$, *welches die PBE erfüllt, und seien* $bpo = (V, <, l), bpo' = (V', <', l')$ *zwei bzgl.* (N, m_0) *aktivierte BPOs mit* $|V|_l = |V'|_{l'}$. *Dann stimmt die eindeutige Folgemarkierung von* bpo *mit der eindeutigen Folgemarkierung von* bpo′ *überein.*

BEWEIS: Die Eindeutigkeit der Folgemarkierungen gilt nach Lemma 5.3.2, da aus PBE die SZZE folgt. Da bpo aktiviert ist, gibt es eine aktivierte Schrittfolge $\tau_1 \ldots \tau_n$ von bpo. Da bpo′ aktiviert ist, gibt es auch eine aktivierte Schrittfolge $\tau'_1 \ldots \tau'_m$ von bpo′. Da $|V|_l = |V'|_{l'}$, folgt $\tau_1 + \ldots + \tau_n = \tau'_1 + \ldots + \tau'_m$. Damit folgt aus PBE, dass die Folgemarkierungen der zwei Schrittfolgen übereinstimmen. Somit stimmen die eindeutigen Folgemarkierungen von bpo und bpo′ überein. □

Es lässt sich beobachten, dass es für ein markiertes Netz, welches die PBE nicht erfüllt, immer zwei aktivierte Schrittfolgen gibt, welche insgesamt aus den selben Anzahlen an Transitionsvorkommen bestehen, so dass die zwei Schrittfolgen unterschiedliche Folgemarkierungen haben. Allerdings sind die zugehörigen BPOs nicht notwendigerweise aktiviert.
Die meisten üblichen Netzklassen erfüllen über die SZZE hinaus auch noch die PBE, beispielsweise gilt für alle in Abschnitt 3.2 diskutierten Netztypen die PBE.

Beispiel

BEISPIEL: *Die Netztypen* $\mathcal{N}_{st}, \mathcal{N}_{en}, \mathcal{N}_{sti}$ *und* $\mathcal{N}_{\overleftarrow{sti}}$ *erfüllen neben der SZZE auch die PBE.*

Wir betrachten nun noch eine weitere zur PBE unabhängige Eigenschaft, welche die SZZE derart einschränkt, dass die Charakterisierung der Aktiviertheit von BPOs aus Lemma 5.3.3 so vereinfacht werden kann, dass nicht mehr jede Co-Menge und jedes Präfix einer Co-Menge betrachtet werden muss, sondern nur noch jeder Schnitt und das jeweils eindeutige Präfix eines Schnittes. Wir formulieren hierfür eine hinreichende Bedingung, die sog. Zwischen-Zustands-Eigenschaft (ZZE). Diese wird im Rahmen von klassischen Netzklassen häufig angeführt und hat in verschiedenen Bereichen eine wichtige Bedeutung [17, 18, 173, 146, 68]. Entsprechend stellt sie auch eine natürliche Einschränkung für unsere parametrische Petrinetzdefinition dar. Wir formulieren die ZZE wiederum für Netze und Netztypen.
Ein markiertes Netz (N, m_0), $N = (P, T, W)$, eines Netztyps erfüllt die ZZE, wenn für alle erreichbaren Markierungen m, m', m'' und Transitionsschritte τ_1, τ_2 die folgende Beziehung gilt.

$$m \xrightarrow{\tau_1 + \tau_2} m' \implies m \xrightarrow{\tau_1 \tau_2} m'.$$

Ein Netztyp $\mathcal{N} = (LS, LE, \mapsto)$ erfüllt die ZZE, wenn für alle Kombinationen von lokalen Zuständen $s, s', s'' \in LS$ und lokale Ereignisse $e_1, e_2 \in LE$ die folgende Beziehung gilt.

$$s \xrightarrow{e_1+e_2} s' \implies s \xrightarrow{e_1 e_2} s'.$$

Es ist zu beobachten, dass die ZZE die SZZE impliziert, allerdings ist sie unvergleichbar mit der PBE. Wir zeigen wieder zuerst den Zusammenhang der ZZE für Netztypen und Netze, bevor wir anschließend das zentrale Resultat für die ZZE beweisen.

LEMMA 5.3.7 *Lemma 5.3.7*
Sei $\mathcal{N} = (LS, LE, \mapsto)$ ein Netztyp. Falls $\mathcal{N}$ die ZZE erfüllt, dann erfüllt jedes markierte Netz des Typs $\mathcal{N}$ die ZZE. Falls $\mathcal{N}$ die ZZE nicht erfüllt, dann gibt es ein markierte Netz des Typs $\mathcal{N}$, welches die ZZE nicht erfüllt.

BEWEIS: Wir beginnen mit dem Falle, dass $\mathcal{N}$ die ZZE erfüllt. Sei (N, m_0), $N = (P, T, W)$, ein markiertes Netz vom Netztyp $\mathcal{N}$. Seien m, m' erreichbare Markierungen und τ_1, τ_2 Transitionsschritte, so dass $m \xrightarrow{\tau_1+\tau_2} m'$. Nach der Schaltregel für Netztypen gilt für jede Stelle $p \in P$ die Beziehung $m(p) \xrightarrow{\sum_{t\in T}(\tau_1+\tau_2)(t)W(p,t)} m'(p)$. Wir bezeichnen $e_i = \sum_{t\in T} \tau_i(t)W(p,t)$ für $i = 1, 2$, $s = m(p)$ und $s' = m'(p)$. Aus der ZZE folgt dann $s \xrightarrow{e_1 e_2} s'$. Damit können wir $m \xrightarrow{\tau_1\tau_2} m'$ folgern.
Falls $\mathcal{N}$ die ZZE nicht erfüllt, dann gibt es lokale Zustände $s, s' \in LS$ und lokale Ereignisse $e_1, e_2 \in LE$, so dass $s \xrightarrow{e_1+e_2} s'$ gilt, aber $s \xrightarrow{e_1 e_2} s'$ nicht gilt. Wir konstruieren ein Netz (N, m_0), $N = (P, T, W)$, vom Netztyp $\mathcal{N}$, welches nicht die ZZE erfüllt, indem wir $P = \{p\}$, $T = \{t_1, t_2\}$, $m_0(p) = s$, $W(p, t_1) = e_1$, $W(p, t_2) = e_2$ setzen. □

LEMMA 5.3.8 *Lemma 5.3.8*
Sei (N, m_0), $N = (P, T, W)$, ein markiertes Netz vom Netztyp $\mathcal{N} = (LS, LE, \mapsto)$, welches die ZZE erfüllt, und sei bpo $= (V, <, l)$ *mit $l : V \to T$ eine BPO.*
Falls bpo *bzgl. (N, m_0) aktiviert ist, dann folgt für jeden Schnitt C von* bpo*, dass jede Schrittfolge σ des Präfixes von C aktiviert ist und $|C|_l$ in der Folgemarkierung von σ aktiviert ist.*
Falls es für jeden Schnitt C von bpo *eine aktivierte Schrittfolge σ des Präfixes von C gibt, so dass $|C|_l$ in der Folgemarkierung von σ aktiviert ist, dann ist* bpo *bzgl. (N, m_0) aktiviert.*

BEWEIS: Da jeder Schnitt eine nicht-leere Co-Menge ist, ergibt sich die erste Aussage aus Lemma 5.3.3. Für die zweite Aussage betrachten wir eine BPO bpo, welche nicht bzgl. (N, m_0) aktiviert ist. Sei $bpo_p = (V_p, <_p, l_p)$ ein Präfix von bpo, welches nicht aktiviert ist und minimal mit dieser Eigenschaft ist, i.e. jedes echte Präfix von bpo_p ist aktiviert. Nach Lemma 5.3.1 gibt es eine nicht-leere Co-Menge C' von bpo_p und ein Präfix $bpo' = (V', <', l')$ von C' (bzgl. bpo_p und bpo) derart, dass entweder bpo' nicht aktiviert ist oder $|C'|_l$ in der nach Lemma 5.3.2 eindeutigen Folgemarkierung von bpo' nicht aktiviert ist. Da bpo' ein echtes Präfix von bpo_p ist, kann nur der zweite Fall zutreffen. Wir erweitern C' nun zu einem Schnitt C von bpo, indem wir maximale Knoten von bpo' und minimale Knoten von $V \setminus V'$ bzgl. $<$ hinzufügen. Wir schreiben $D = \{v \in V \setminus V' \mid v' < v \implies v' \in V'\}$ und $E = \{v \in V' \mid v \,co_< D \wedge (v < v' \implies v' \notin V')\}$. Die Menge $C = D \cup E$ ist ein Schnitt von bpo, welcher $C' \subseteq C$ erfüllt (da $C' \subseteq D$). Das

Präfix $bpo'' = (V'', <'', l'')$ von C erfüllt $V'' \subseteq V' = V'' \cup E$, und ist daher entsprechend der Minimalitätseigenschaft von bpo_p aktiviert. Wir nehmen nun an, dass $|C|_l$ in der eindeutigen Folgemarkierung von bpo'' aktiviert ist. Nach der ZZE ist dann auch die Schrittfolge $|E|_l|D|_l$ in der Folgemarkierung von bpo'' aktiviert. Die Markierung, welche durch das Schalten des Schrittes $|E|_l$ in der Folgemarkierung von bpo'' erreicht wird, muss die eindeutige Folgemarkierung von bpo' sein. Der Schritt $|D|_l$ ist also in der Folgemarkierung von bpo' aktiviert. Da $C' \subseteq D$, folgt wiederum aus der ZZE, dass $|C'|_l$ in der Folgemarkierung von bpo' aktiviert ist. Dies ist ein Widerspruch. Somit kann $|C|_l$ in der Folgemarkierung von bpo'' nicht aktiviert sein, wobei bpo'' das Präfix von C ist. Da bpo'' aktiviert ist, gilt, dass jede Schrittfolge σ von bpo'' aktiviert ist, aber $|C|_l$ in der Folgemarkierung von σ nicht aktiviert ist. Dies zeigt die Behauptung. □

Die ZZE ist in dem Sinne notwendig für die Charakterisierung der Aktiviertheit dieses Lemmas, dass die zweite Aussage des Lemmas für ein Netz (N, m_0), $N = (P, T, W)$, welches die ZZE nicht erfüllt, aber die SZZE erfüllt, nicht gilt. Dies lässt sich folgendermaßen einsehen. Für (N, m_0) gibt es Schritte τ_1, τ_2 und eine erreichbare Markierung m, so dass $m \xrightarrow{\tau_1+\tau_2} m'$ aber nicht $m \xrightarrow{\tau_1\tau_2}$. Sei $bpo' = (V', <', l')$ eine schrittweise lineare BPO, so dass $m_0 \xrightarrow{\sigma(bpo')} m$. Dann würde aus der zweiten Aussage von Lemma 5.3.8 folgen, dass eine BPO der Form $bpo = (V' \cup V, <' \cup (V' \times V), l)$, $l|_{V'} = l'$, $|V|_l = \tau_1 + \tau_2$ aktiviert ist, obwohl sie nicht aktiviert ist.

Beispiel

BEISPIEL: *Als Beispiel für diese Situation betrachten wir den Netztyp* $(\{s\}, \{0, 1, 2\}, \{s \xrightarrow{0} s\})$, *wobei* 0 *das neutrale Element des Monoids von lokalen Ereignissen ist und die Operation des Monoids durch* $1 + 1 = 2$, $1 + 2 = 0$ *und* $2 + 2 = 1$ *gegeben ist (dasselbe Monoid wurde auch im Beispiel im Anschluss an Definition 5.3.1 betrachtet). Weiter sei ein markiertes Netz* (N, m_0), $N = (P, T, W)$, *dieses Typs durch* $P = \{p\}$, $T = \{t_1, t_2\}$, $m_0(p) = s$, $W(p, t_1) = 1$, $W(p, t_2) = 2$ *gegeben. Die BPO* $bpo = (\{v_1, v_2\}, \emptyset, l)$, $l(v_i) = t_i$ *für* $i = 1, 2$ *ist bzgl.* (N, m_0) *nicht aktiviert, da die Schrittfolgen* t_1t_2 *und* t_2t_1 *nicht aktiviert sind. Allerdings ist* $t_1 + t_2$ *bzgl.* (N, m_0) *aktiviert und dieser Schritt entspricht dem einzigen Schnitt der BPO (in obiger Notation gilt hier* $m = m_0$ *und damit* $V' = \emptyset$*).*

Im Allgemeinen sind solche Netze, welche die SZZE, aber nicht die ZZE erfüllen, in unserem Kontext problematisch. Für solche Netze muss es einen in einer erreichbaren Markierung aktivierten Transitionsschritt geben, so dass eine sequentielle Zerlegung des Schrittes nicht mehr aktiviert ist. Dies steht nicht im Einklang zu unseren grundsätzlichen, intuitiven Annahmen im Rahmen der Modellierung von Abläufen von Netzen mit BPOs.

Beispiel

BEISPIEL: *Wie gesagt, ist die ZZE eine typische und viel betrachtete Eigenschaft von Netzklassen. Sie wird beispielsweise von den Netztypen* $\mathcal{N}_{st}$, $\mathcal{N}_{en}$ *und* $\mathcal{N}_{sti}$ *erfüllt. Allerdings gilt sie für den wichtigen Netztyp* $\mathcal{N}_{\overleftarrow{sti}}$ *der STI-Netze unter der a-priori Semantik nicht. Für diesen sind aber auch, wie in Abschnitt 3.3 diskutiert, BPOs als Mittel zur Ablaufbeschreibung nicht geeignet. Es gibt aber auch noch ein weiteres wichtiges Beispiel, in dem die ZZE nicht erfüllt ist. So gilt die ZZE Für Netze mit speziellen Schritt-Aktivierungsregeln wie beispielsweise Lokalitäten nicht (vgl. [146, 68]).*

Abschließend zeigen wir mit folgenden Beispielen, dass es tatsächlich jeweils Netztypen gibt, welche die SZZE nicht erfüllen bzw. die SZZE,

aber nicht die ZZE und die PBE, erfüllen bzw. die ZZE, aber nicht die PBE, erfüllen bzw. die PBE, aber nicht die ZZE, erfüllen bzw. beide die PBE und die ZZE erfüllen.

BEISPIEL: *Beispiel*

- *Der Netztyp des Beispiels nach Definition 5.3.1 erfüllt die SZZE nicht, da* $s \xrightarrow{(1)(2)} s'$ *und* $s \xrightarrow{1+2} s$.
- *Der Netztyp des Beispiels nach Lemma 5.3.8 erfüllt die SZZE, aber nicht die ZZE, da* $s \xrightarrow{1+2} s$, *aber nicht* $s \xrightarrow{(1)(2)}$.
- *Der Netztyp des zweiten Beispiels nach Lemma 5.3.4 erfüllt die SZZE, aber nicht die PBE, da* $s \xrightarrow{(a)(b)} s''$ *und* $s \xrightarrow{(b)(a)} s'$.
- *Eine Kombination der vorherigen zwei Netztypen führt zu dem Netztyp* $(\{s, s', s''\}, \{0, 1, 2, a, b, c\}, \{s \xrightarrow{a} s', s \xrightarrow{b} s'', s' \xrightarrow{b} s'', s'' \xrightarrow{a} s', s \xrightarrow{0} s, s' \xrightarrow{0} s', s'' \xrightarrow{0} s''\})$, *wobei* 0 *das neutrale Element des Monoids von lokalen Ereignissen ist und die Operation des Monoids durch* $1+1 = 2$, $1+2 = 0$, $2+2 = 1$ *und* $x+y = c$ *für* $x \in \{1, 2, a, b, c\}, y \in \{a, b, c\}$ *gegeben ist. Dieser Netztyp erfüllt die SZZE, aber weder die ZZE noch die PBE.*
- *Der Netztyp des zweiten Beispiels nach Lemma 5.3.4, welcher nicht die PBE erfüllt, erfüllt die ZZE.*
- *Der Netztyp des Beispiels nach Lemma 5.3.8, welcher nicht die ZZE erfüllt, erfüllt die PBE. Auch der Netztyp* $\mathcal{N}_{\overleftarrow{sti}}$ *erfüllt die PBE, aber nicht die ZZE.*
- *Die Netztypen* $\mathcal{N}_{st}$, $\mathcal{N}_{en}$ *und* $\mathcal{N}_{sti}$ *erfüllen die PBE und die ZZE.*

5.3.2 *Markenfluss-BPOs für Netztypen*

In diesem Unterabschnitt verallgemeinern wir das Konzept der Markenflüsse aus Abschnitt 4.3 auf Netztypen. Um den Fluss von Marken für Netztypen definieren zu können, müssen wir eine geeignete algebraische Struktur der lokalen Zustände sowie geeignete Beziehungen zwischen den Mengen der lokalen Ereignisse und der lokalen Zustände voraussetzen. Das Ziel ist auch hier die Herleitung eines geeigneten Begriffes einer Markenfluss-BPO. Dieser ermöglicht dann wiederum eine äquivalente Charakterisierung der Aktiviertheit von BPOs.
Bei S/T-Netzen sind Markierungen durch die Anzahlen an Marken in den Stellen eines Netzes gegeben. In Netztypen werden anstelle von Markenanzahlen in Stellen lokale Zustände von Stellen betrachtet. Dem Fluss von Marken in S/T-Netzen müsste also eine Art von Fluss von lokalen Zuständen entsprechen. Bei S/T-Netzen herrscht ein Markenfluss zwischen zwei Schaltereignissen vor, wenn Marken von dem einen Ereignis produziert und dann von dem anderen Ereignis konsumiert werden. Im Allgemeinen lässt sich bei Netztypen nicht sinnvoll von einem Fluss von lokalen Zuständen basierend auf einer Art von Produzieren und Konsumieren von lokalen Zuständen durch Schaltereignisse sprechen. Daher definieren wir sog. Fluss-Netztypen. Fluss-Netztypen sind spezielle Netztypen, für welche sich ein entsprechendes Flussverhalten für lokale Zustände definieren lässt. Die Hauptidee ist hierbei,

die Menge der lokalen Zustände mit einer geeigneten algebraischen Struktur zu versehen, so dass lokale Zustände addiert und in Beziehung gesetzt werden können. Insbesondere soll es eine Menge von generierenden lokalen Zuständen geben, welche einzelnen Marken von S/T-Netzen entsprechen. Die generierenden lokalen Zustände sind also die eigentlich fließenden Objekte. Außerdem werden die Mengen der lokalen Ereignisse und der lokalen Zustände durch geeignete Morphismen verknüpft. Diese modellieren einen lokalen Fluss, welcher durch ein lokales Ereignis entsteht. Dieser lokale Fluss muss konsistent zur partiell definierten Veränderung von lokalen Zuständen durch lokale Ereignisse sein. Im Anschluss an die nun folgende formale Definition eines Fluss-Netztyps werden wir erklären, dass jede dieser Einschränkungen notwendig ist, um einen sinnvollen Markenfluss-Begriff für Netztypen zu ermöglichen.

Definition 5.3.3 (Fluss-Netztyp)

Definition 5.3.3 (Fluss-Netztyp)
Ein Fluss-Netztyp $(\mathcal{N}, f)$ *ist ein Netztyp* $\mathcal{N} = (LS, LE, \mapsto)$ *zusammen mit einer Fluss-Abbildung* $f = (f_1, f_2) : LE \to LS \times LS$ *mit folgenden Eigenschaften:*

- LS *ist ein freies kommutatives Monoid,*
- f_1, f_2 *sind Monoid-Morphismen,*
- $(s, e, s') \in \mapsto \implies f_1(e) \leqslant s \wedge s' = s - f_1(e) + f_2(e)$.

Der Effekt des Schaltens eines Schrittes nebenläufiger Transitionen auf einen lokalen Zustand ergibt sich durch die Effekte der entsprechenden lokalen Ereignisse. Hierbei nehmen wir nun an, dass, falls ein Schritt von lokalen Ereignissen, welcher ja wiederum ein lokales Ereignis definiert, in einem lokalen Zustand aktiviert ist, dann jedes der lokalen Ereignisse des Schrittes einen Teil des lokalen Zustandes konsumiert, modelliert durch f_1, und einen Teil des nachfolgenden lokalen Zustandes produziert, modelliert durch f_2. Dementsprechend lässt sich der lokale Zustand in verschiedene Anteile zerlegen, welche selbst wieder lokale Zustände sind. Diese lassen sich umgekehrt wieder zu dem ursprünglichen lokalen Zustand komponieren. Natürlich ist die Komposition lokaler Zustände assoziativ und kommutativ, da der Effekt des Schaltens eines Schrittes nebenläufiger Transitionen auf einen lokalen Zustand unabhängig von irgendeiner Reihenfolge der Betrachtung der Transitionen sein muss. Wir benötigen auch einen Null-Fluss. Dementsprechend ist ein neutraler lokaler Zustand notwendig. Somit ist es gerechtfertigt vorauszusetzen, dass die Menge der lokalen Zustände ein kommutatives Monoid ist. Die Notwendigkeit der Zerlegbarkeit lokaler Zustände zeigt, dass generierende lokale Zustände sinnvoll sind. Daher wird ein freies kommutatives Monoid von lokalen Zuständen betrachtet. Für die Abbildungen f_1 und f_2 wird gefordert, dass sie Morphismen sind, um zu garantieren, dass von einem Transitionsschritt konsumierte bzw. produzierte lokale Zustände konsistent zu den von den einzelnen Transitionen des Schrittes konsumierten bzw. produzierten lokalen Zuständen sind. Die letzte Eigenschaft von Fluss-Netztypen stellt sicher, dass die Veränderung $(s, e, s') \in \mapsto$ eines lokalen Zustandes s zu einem lokalen Zustand s' durch ein lokales Ereignis e konsistent zu dem Fluss $f(e)$ von e ist und dass s' mithilfe des Flusses $f(e)$ aus s berechnet werden kann. Die Abbildung f wird später den Markenfluss auf BPOs festlegen.

Bevor wir zu dem Konzept der Markenflüsse kommen, sollen noch einige Grundlagen zu Fluss-Netztyp geklärt werden. Zuerst einmal lassen sich die bekannten Netzklassen, also die verbreiteten Erweiterungen der klassischen Petrinetzdefinitionen, als Instanziierungen der Fluss-Netztyp-Definition gewinnen. Hierzu ergänzen wir einfach die Repräsentation der Netzklasse als Netztyp auf natürliche Art und Weise durch eine geeignete Fluss-Abbildung. Als Beispiele betrachten wir hier wieder die Netztypen $\mathcal{N}_{st}$, $\mathcal{N}_{en}$, $\mathcal{N}_{sti}$ und $\mathcal{N}_{\overleftarrow{sti}}$.

BEISPIEL: *Die Netztypen $\mathcal{N}_{st}$, $\mathcal{N}_{sti}$ und $\mathcal{N}_{\overleftarrow{sti}}$ können intuitiv mit geeigneten Fluss-Abbildungen versehen werden, so dass sich ein entsprechender Fluss-Netztyp ergibt (es wird jeweils das freie kommutative Monoid $(\mathbb{N}, +, 0)$ für die lokalen Zustände betrachtet):* *Beispiel*

- $f_{st}(i,j) = (i,j)$,
- $f_{sti}(i,j,k) = (i,j)$,
- $f_{\overleftarrow{sti}}(i,j,k) = (i,j)$.

Auch $\mathcal{N}_{en}$ kann zu einem Fluss-Netztyp erweitert werden. Wir setzen hierzu $LS = \mathbb{N} \supset \{0,1\}$ und $LE = \mathbb{N} \times \mathbb{N}$, wobei $(0,0)$, $(1,0)$ und $(0,1)$ zu nop, in *und* out *korrespondieren und alle anderen lokalen Ereignisse* failure *entsprechen. Eine entsprechende Fluss-Abbildung ist dann durch $f_{en}(i,j) = (i,j)$ gegeben.*

In Bezug zu den im letzten Unterabschnitt diskutierten Eigenschaften von Netztypen gilt, dass die Anforderungen an Fluss-Netztypen die PBE und damit natürlich auch die SZZE garantieren. Insbesondere ist also die Folgemarkierung einer aktivierten BPO eindeutig und hängt nur von den Anzahlen der Transitionsvorkommen in der BPO ab.

LEMMA 5.3.9 *Lemma 5.3.9*
Ein Fluss-Netztyp $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, erfüllt die PBE.

BEWEIS: Seien s, s', s'' lokale Zustände, u eine Multimenge von lokalen Ereignissen und $u = u_1 + \ldots + u_n = u'_1 + \ldots + u'_m$ zwei Partitionen von u. Wir bezeichnen $e_i = \sum_{e \in u_i} u_i(e)e$ und $e'_i = \sum_{e \in u'_i} u'_i(e)e$. Sei weiter $s \xrightarrow{e_1 \ldots e_n} s'$ und $s \xrightarrow{e'_1 \ldots e'_m} s''$. Wir schreiben $s = s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} s_n = s'$. Da f_1 und f_2 Monoid-Morphismen sind, folgt $s_i = s_{i-1} + f_2(e_i) - f_1(e_i) = s_{i-1} + f_2(\sum_{e \in u_i} u_i(e)e) - f_1(\sum_{e \in u_i} u_i(e)e) = s_{i-1} + \sum_{e \in u_i} u_i(e) f_2(e) - \sum_{e \in u_i} u_i(e) f_1(e)$ für jedes $i = 1, \ldots, n$. Durch Zusammenfügen all dieser Gleichungen ergibt sich $s' = s + \sum_{i=1}^{n} (\sum_{e \in u_i} u_i(e) f_2(e)) - \sum_{i=1}^{n} (\sum_{e \in u_i} u_i(e) f_1(e)) = s + \sum_{e \in u} u(e) f_2(e) - \sum_{e \in u} u(e) f_1(e)$. Analog erhalten wir auch $s'' = s + \sum_{e \in u} u(e) f_2(e) - \sum_{e \in u} u(e) f_1(e)$. □

Nun lässt sich bei Fluss-Netztypen die durch die Anzahlen der Transitionsvorkommen in einer BPO bestimmte Folgemarkierung einer aktivierten BPO folgendermaßen sehr einfach berechnen.

LEMMA 5.3.10 *Lemma 5.3.10*
Sei (N, m_0), $N = (P, T, W)$, ein markiertes Netz eines Fluss-Netztyps $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$. Die Folgemarkierung m einer bzgl. (N, m_0) aktivierten BPO $bpo = (V, <, l)$ ist gegeben durch

$$m(p) = m_0(p) + \sum_{v \in V} f_2(W(p, l(v))) - \sum_{v \in V} f_1(W(p, l(v))) \text{ für } p \in P.$$

BEWEIS: Jede Schrittfolge $\sigma = \tau_1 \dots \tau_n$ von bpo ist bzgl. (N, m_0) aktiviert, i.e. es gibt Markierungen $m_1, \dots, m_n$, so dass $m_0 \xrightarrow{\tau_1} m_1 \xrightarrow{\tau_2} \dots \xrightarrow{\tau_n} m_n$, wobei $(m_{i-1}(p), \sum_{t\in T} \tau_i(t) \cdot W(p,t), m_i(p)) \in \mapsto$ für $i = 1, \dots, n$ und $p \in P$. Entsprechend der Definition eines Fluss-Netztyps folgt $m_i(p) = m_{i-1}(p) - f_1(\sum_{t\in T} \tau_i(t) \cdot W(p,t)) + f_2(\sum_{t\in T} \tau_i(t) \cdot W(p,t)) = m_{i-1}(p) - \sum_{t\in T} \tau_i(t) \cdot f_1(W(p,t)) + \sum_{t\in T} \tau_i(t) \cdot f_2(W(p,t))$ für $i = 1, \dots, n$ und $p \in P$. Für die Folgemarkierung m_n von σ berechnen wir daraus $m_n(p) = m_0(p) + \sum_{i=1}^{n}(\sum_{t\in T} \tau_i(t) \cdot f_2(W(p,t))) - \sum_{i=1}^{n}(\sum_{t\in T} \tau_i(t) \cdot f_1(W(p,t))) = m_0(p) + \sum_{t\in T} |V|_l(t) \cdot f_2(W(p,t)) - \sum_{t\in T} |V|_l(t) \cdot f_1(W(p,t)) = m_0(p) + \sum_{v\in V} f_2(W(p, l(v))) - \sum_{v\in V} f_1(W(p, l(v)))$ für $p \in P$. Per Definition gilt $m = m_n$. Dies zeigt die Behauptung. □

Für einen Fluss-Netztyp $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, definieren wir nun einen Markenfluss einer BPO bpo $= (V, <, l)$ wie folgt analog zum Falle von S/T-Netzen. Eine Funktion $x :<\to LS$ heißt Markenfluss-Funktion (auf bpo bzgl. $(\mathcal{N}, f)$). Für einen Knoten $v \in V$ bezeichnen wir $Zu_x(v) = \sum_{v'<v} x(v', v)$ als Marken-Zufluss von v (bzgl. x) und $Ab_x(v) = \sum_{v<v'} x(v, v')$ als Marken-Abfluss von v (bzgl. x). Ein partiell geordneter Markenfluss bzgl. $(\mathcal{N}, f)$ ist ein Paar $\mu = (\text{bpo}, X)$, wobei bpo $= (V, <, l)$ eine $*$-BPO und X eine Menge von Markenfluss-Funktionen auf bpo bzgl. $(\mathcal{N}, f)$ ist. Wir definieren nun die Begriffe des Markenfluss-Prozesses und der Markenfluss-BPO für Fluss-Netztypen.

Definition 5.3.4 (Markenfluss-Prozess)

DEFINITION 5.3.4 (MARKENFLUSS-PROZESS)
Sei $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, ein Fluss-Netztyp. Ein partiell geordneter Markenfluss $\mu = (\text{bpo}, X)$, bpo $= (V, <, l)$, bzgl. $(\mathcal{N}, f)$ ist ein Markenfluss-Prozess eines markierten Netzes (N, m_0), $N = (P, T, W)$, vom Netztyp $\mathcal{N}$, falls $l(V \setminus \{q_{bpo}, s_{bpo}\}) \subseteq T$ gilt und eine bijektive Funktion $\phi : X \to P$ existiert, so dass

(i) $\forall x \in X : Ab_x(q_{bpo}) = m_0(\phi(x))$.

(ii) $\forall x \in X, \forall v \in V \setminus \{q_{bpo}, s_{bpo}\} : Ab_x(v) = f_2(W(\phi(x), l(v))) \wedge Zu_x(v) = f_1(W(\phi(x), l(v)))$.

Ein partiell geordneter Markenfluss ist also ein Markenfluss-Prozess, wenn jede Markenfluss-Funktion zu einer Stelle des Netzes korrespondiert. Ein Markenfluss, welcher zu einer Kante (v, v') der BPO zugeordnet ist, repräsentiert den lokalen Zustand, welcher durch das Schalten von $l(v)$ in der Stelle produziert und durch das Schalten von $l(v')$ konsumiert wird. Der Quellenknoten repräsentiert eine Transition, welche die Anfangsmarkierung des Netzes produziert, während der Senkenknoten eine Transition darstellt, welche die Folgemarkierung der BPO konsumiert.

Definition 5.3.5 (Markenfluss-BPO)

DEFINITION 5.3.5 (MARKENFLUSS-BPO)
Sei $\mu = (\text{bpo}, X)$ ein Markenfluss-Prozess eines markierten Netzes (N, m_0), $N = (P, T, W)$, vom Netztyp $\mathcal{N}$. Die $*$-Einschränkung bpo_μ der $*$-BPO bpo heißt Markenfluss-BPO von (N, m_0) (bzgl. μ).

Die Definitionen verallgemeinern die entsprechenden Markenfluss-Konzepte für S/T-Netze. Im Falle des Netztyps $\mathcal{N}_{st}$ ergeben sich die Formulierungen aus Kapitel 4. An dieser Stelle wird nun auch formal klar, warum alle Einschränkungen von Fluss-Netztypen aus Definition 5.3.3 konzeptuell notwendig sind. Wir benötigen die Flussabbildung f mit den entsprechenden Eigenschaften zum Rechnen mit Flüssen und

wir benötigen ein kommutatives Monoid von lokalen Zuständen, um auch mit diesen geeignet rechnen zu können. Die einzige Forderung, die sich nicht ohne weiteres formal motivieren lässt, ist die Betrachtung eines sogar freien kommutativen Monoids von lokalen Zuständen. Wir zeigen an einem Beispiel, dass wir Generatoren der Menge der lokalen Zustände benötigen, um zu jeder aktivierten BPO einen entsprechenden Markenfluss-Prozess finden zu können. Solche Generatoren gibt es gerade für freie kommutative Monoide. Genauer gesagt ist die entscheidende Charakteristik eines freien kommutativen Monoids, dass sich jedes Element bis auf die Reihenfolge eindeutig durch die Elemente der endlichen Menge der Generatoren des Monoids darstellen lässt.

Beispiel

BEISPIEL: *Das kommutative Monoid* $(M, +, (0,0))$, *wobei* $M = \mathbb{N} \times \mathbb{N} \setminus \{(0,1),(1,0)\}$ *gilt und die Operation* $+$ *die komponentenweise Addition ist, besitzt keine entsprechenden Generator-Elemente. Die Elemente* $(1,1)$, $(2,0)$ *und* $(0,2)$ *können nicht als Summe anderer Elemente von* M *dargestellt werden und müssten somit Generatoren sein. Allerdings ließe sich dann das Element* $(2,2)$ *auf zwei Arten durch Generatoren darstellen, es gilt nämlich* $(2,2) = 2 \cdot (1,1)$ *und* $(2,2) = (2,0) + (0,2)$. *Wir definieren* $\mathcal{N} = (M, M \times M, \mapsto)$ *mit* $(n,(i,j),n') \in \mapsto$ *genau dann, wenn* $n \geqslant i$ *und* $n' = n - i + j$. *Weiter setzen wir* $f(i,j) = (i,j)$. *Wir betrachten das markierte Netz* (N, m_0), $N = (P, T, W)$, *vom Netztyp* $\mathcal{N}$, *welches durch* $T = \{t_1, t_2, t_3, t_4\}$, $P = \{p\}$, $W(p,t_1) = ((0,0),(1,1))$, $W(p,t_2) = ((0,0),(1,1))$, $W(p,t_3) = ((2,0),(0,0))$, $W(p,t_4) = ((0,2),(0,0))$, $m_0(p) = (0,0)$ *gegeben ist. Dann ist die BPO* $\mathrm{bpo} = (\{\nu_1,\nu_2,\nu_3,\nu_4\},\{(\nu_1,\nu_3),(\nu_1,\nu_4),(\nu_2,\nu_3),(\nu_2,\nu_4)\},l)$ *mit* $l(\nu_i) = t_i$ *bzgl.* (N, m_0) *aktiviert. Allerdings lässt sich in obigen Notationen kein Markenfluss-Prozess finden, so dass* bpo *die zugehörige Markenfluss-BPO ist. Das Problem hierbei ist, dass sich keine geeignete Markenfluss-Verteilung finden lässt, da* ν_1 *und* ν_2 *jeweils den lokalen Zustand* $(1,1)$ *produzieren. Dadurch entsteht zwar insgesamt der lokale Zustand* $(2,2)$, *welcher ausreicht, um* ν_3 *und* ν_4 *in einem Schritt zu aktivieren, da diese Ereignisse* $(2,0)$ *und* $(0,2)$ *konsumieren und* $(2,0) + (0,2) = (2,2)$ *gilt. Aber* ν_3 *konsumiert dann jeweils den Anteil* $(1,0)$ *sowohl von* ν_1 *als auch von* ν_2. *Da aber* $(1,0) \notin M$, *ist eine solche Fluss-Verteilung der lokalen Zustände nicht gültig. Somit gibt es keine Verteilung von lokalen Zustände auf die Kanten von* $\mathrm{bpo}^\star$, *so dass sich ein Markenfluss-Prozess ergibt. Bei genauer Betrachtung liegt die Problematik hier tatsächlich an dem Fehlen von entsprechenden Generatoren der lokalen Zustände.*

Im Gegensatz zu diesem Beispiel gilt für die von uns gewählten Fluss-Netztypen schon, dass jede aktivierte BPO eine Markenfluss-BPO ist. Diese entscheidende enge Beziehung von Markenfluss-BPOs zu dem Begriff der Aktiviertheit von BPOs ist das zentrale Resultat des Unterabschnittes und wird im folgenden Satz gezeigt. Der Satz verallgemeinert das wichtige Resultat aus [142], dass jede aktivierte BPO eines S/T-Netzes eine Sequentialisierung einer Prozess-BPO ist.
Letzteres Resultat haben wir auch in Abschnitt 4.3 zum Beweis, dass jede aktivierte BPO eines S/T-Netzes eine Markenfluss-BPO ist, verwendet. Es erfordert, wie in [216] dargestellt, einen sehr komplizierten Beweis. Ein alternativer Beweis in [216, 217] greift dafür zentral auf den bekannten Heiratssatz von Hall aus der Graphentheorie zurück. In [131] wird ein weiterer, wiederum eigenständiger Beweis für das Resultat vorgestellt. Dieser ist etwas einfacher als der ursprüngliche Beweis in [142], aber immer noch mehrere Seiten lang. Es ist daher nicht verwunderlich, dass auch der Beweis unseres folgenden Satzes

für Fluss-Netztypen aufwändig ist. Der Beweis verwendet die zentralen Ideen des angesprochenen Beweises aus [131] in dem neuen Kontext von Fluss-Netztypen.

Satz 5.3.1

SATZ 5.3.1
Sei (N, m_0), $N = (P, T, W)$, *ein markiertes Netz eines Fluss-Netztyps* $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, *und sei* $bpo = (V, <, l)$ *eine bzgl.* (N, m_0) *aktivierte BPO. Dann ist* bpo *eine Markenfluss-BPO von* (N, m_0).

BEWEIS: Sei LS das durch $\mathbb{N}^A$ gegebene freie kommutative Monoid über A. Wir führen einen Widerspruchsbeweis, d.h. wir nehmen an, dass es eine Stelle p gibt, für die es keine Markenfluss-Funktion $x :<^* \rightarrow LS$ mit folgenden Eigenschaften gibt (die Eigenschaften sind hier anders gegliedert als in Definition 5.3.4):

(i) $\forall v \in V : Zu_x(v) = f_1(W(p, l(v)))$.

(ii) $Ab_x(q_{bpo}) = m_0(p)$ und $\forall v \in V : Ab_x(v) = f_2(W(p, l(v)))$.

Wir setzen $V^* = \{v_0, \ldots, v_{|V^*|}\}$, so dass $v_i <^* v_j$ die Beziehung $i < j$ impliziert. Insbesondere gilt also $v_0 = q_{bpo}$ und $v_{|V^*|} = s_{bpo}$. Wir betrachten die Menge $\mathcal{X}$ von Markenfluss-Funktionen $x :<^* \rightarrow LS$, welche (i) sowie die Beziehungen $Ab_x(q_{bpo}) \geqslant m_0(p)$ und $Ab_x(v) \geqslant f_2(W(p, l(v)))$ für alle $v \in V$ erfüllen. Diese Menge ist nicht leer, da beispielsweise die Funktion x, welche durch $x(q_{bpo}, v) = f_1(W(p, l(v)))$ für jedes $v \in V$, $x(v, s_{bpo}) = f_2(W(p, l(v)))$ für jedes $v \in V$, $x(q_{bpo}, s_{bpo}) = m_0(p)$ und $x(v, v') = 0$ für alle $v < v'$, $v, v' \in V$ in $\mathcal{X}$ ist. Nach Annahme gibt es keine Funktion aus $\mathcal{X}$, welche (ii) erfüllt.
Wir sprechen davon, dass $x \in \mathcal{X}$ die Eigenschaft (ii) für einen Index i nicht erfüllt, wenn $i = 0$ und $Ab_x(v_i) > m_0(p)$ oder wenn $i > 0$ und $Ab_x(v_i) > f_2(W(p, l(v_i)))$. Wir bezeichnen mit k_x den kleinsten Index, für den eine Funktion $x \in \mathcal{X}$ die Eigenschaft (ii) nicht erfüllt. Sei $\mathcal{X}_{sup} \subseteq \mathcal{X}$ die nicht-leere Menge aller Markenfluss-Funktionen $x \in \mathcal{X}$, für welche k_x maximal ist, d.h. es gilt $\forall x', x'' \in \mathcal{X}_{sup} : k_{x'} = k_{x''}$ und $\forall x \in \mathcal{X}, \forall x' \in \mathcal{X}_{sup} : k_x \leqslant k_{x'}$. Wir bezeichnen $sup = k_x$ für $x \in \mathcal{X}_{sup}$. Nach Annahme gilt $sup < |V^*|$ (natürlich ist der Fall $sup = 0$ möglich). Wir wählen schließlich eine Markenfluss-Funktion $x_0 \in \mathcal{X}_{sup}$, welche einen minimalen Wert $Ab_{x_0}(v_{sup})$ besitzt, i.e. $\forall x \in \mathcal{X}_{sup} : Ab_x(v_{sup}) \not< Ab_{x_0}(v_{sup})$.
Im Folgenden konstruieren wir aus x_0 eine Co-Menge C von bpo, so dass $|C|_l$ in der Folgemarkierung des Präfixes $bpo' = (D', <|_{D' \times D'}, l|_{D'})$, $D' = \{v \in V \mid v < C\}$, von C nicht aktiviert ist. Da bpo bzgl. (N, m_0) aktiviert ist, ist auch solch ein Präfix von bpo aktiviert. Entsprechend Lemma 5.3.10 gilt für die eindeutige Folgemarkierung m von bpo':

$$m(p) = m_0(p) + \sum_{v \in D'} f_2(W(p, l(v))) - \sum_{v \in D'} f_1(W(p, l(v))).$$

Um zu zeigen, dass $|C|_l$ in m nicht aktiviert ist, werden wir zeigen, dass es keine Markierung m' gibt, so dass $(m(p), \sum_{t \in T} |C|_l(t) \cdot W(p, t), m'(p)) \in \mapsto$. Entsprechend der Definition eines Fluss-Netztyps genügt es hierzu, die folgende Beziehung zu beweisen: $m(p) \not\geqslant f_1(\sum_{t \in T} |C|_l(t) \cdot W(p, t)) = \sum_{t \in T} |C|_l(t) \cdot f_1(W(p, t)) = \sum_{v \in C} f_1(W(p, l(v)))$. Diese Beziehung lässt sich wiederum in folgende Ungleichung umformen:

$$(*) \quad m_0(p) + \sum_{v \in D'} f_2(W(p, l(v))) - \sum_{v \in D'} f_1(W(p, l(v))) \not\geqslant$$

$$\sum_{v \in C} f_1(W(p, l(v))).$$

Eine Herleitung von $(*)$ würde also zeigen, dass $|C|_l$ in der Folgemarkierung von bpo′ nicht aktiviert ist. Dies widerspricht nach Lemma 5.3.3 der Aktiviertheit von bpo.
Wir definieren nun die Menge C. Hierzu definieren wir zuerst eine Menge D. Es wird sich zeigen, dass C eine Co-Menge von bpo ist und dass D das minimale Präfix von C bzgl bpo* ist, i.e. $D = D' \cup \{q_{bpo}\}$.
Wir betrachten ein $a \in A$, so dass $Ab_{x_0}(v_{sup})(a) > f_2(W(p, l(v_{sup})))(a)$ (bzw. $Ab_{x_0}(v_{sup})(a) > m_0(p)(a)$, falls $sup = 0$). Sei D die Menge aller Knoten $v \in V^*$, so dass eine Folge von Knoten $\sigma(v) = v^0 w^1 v^1 \dots w^k v^k$ mit $v^0 = v_{sup}$ und $v^k = v$ sowie

(c1) $\forall j \neq m : w^j \neq w^m \wedge v^j \neq v^m$, und

(c2) $\forall j : x_0(v^j, w^{j+1})(a) > 0 \wedge v^j <^* w^j$

existiert.
Aus $Ab_{x_0}(v_{sup})(a) > 0$ folgt, dass $v_0 = q_{bpo}$ in D enthalten ist. Außerdem folgt für $k = 0$, dass $v_{sup} \in D$. Der Knoten $s_{bpo} = v_{|V^*|}$ ist nicht in D enthalten, da $s_{bpo} \neq v_{sup}$ und es keinen Knoten w mit $s_{bpo} <^* w$ gibt.
Wir definieren

$$C \quad = \quad \{w \in V^* \setminus D \mid \exists v \in D : x_0(v, w)(a) > 0\}$$

Die Menge C repräsentiert den Transitionsschritt, welcher zu viele Marken von $l(v_{sup})$ konsumiert. Wir zeigen in einer Reihe von Beweisschritten, dass C eine Co-Menge von bpo ist, für welche, wie beschrieben, die Beziehung $(*)$ gilt. Die Idee hierbei ist folgende. Falls dies nicht der Fall wäre, dann könnte der Markenfluss x_0 entlang der Folgen $\sigma(v)$ derart umverteilt werden, dass der Marken-Abfluss von v_{sup} bzgl. a reduziert wird, wobei der Marken-Abfluss von v_{sup} bzgl. $a' \neq a$, der Marken-Zufluss von allen Knoten und der Marken-Abfluss von Knoten mit einem Index $i < sup$ unverändert bleiben. Dies würde also der Wahl von x_0 widersprechen.

Behauptung 1: $v_j \in D \implies j \leqslant sup$

Angenommen $j > sup$. Dann ist es folgendermaßen möglich, eine Markenfluss-Funktion $x \in \mathfrak{X}_{sup}$ mit $Ab_x(v_{sup}) < Ab_{x_0}(v_{sup})$ zu konstruieren. Sei $\sigma(v_j) = v^0 w^1 v^1 \dots w^k v^k$, dann setzen wir

$$\begin{aligned} \forall i \quad &: \quad x(v^i, w^i)(a) = x_0(v^i, w^i)(a) + 1 \\ \forall i \quad &: \quad x(v^i, w^{i+1})(a) = x_0(v^i, w^{i+1})(a) - 1 \\ \text{sonst} \quad &: \quad x(v, v')(a') = x_0(v, v')(a'). \end{aligned}$$

Dies widerspricht der Wahl von x_0.

Behauptung 2: $v_j \in D \implies x_0(v_j, s_{bpo})(a) = 0$

Aus Behauptung 1 folgt $j \leqslant sup$. Angenommen $x_0(v_j, s_{bpo})(a) > 0$. Dann ist es folgendermaßen möglich, eine Markenfluss-Funktion $x \in$

$\mathcal{X}_{sup}$ mit $Ab_x(v_{sup}) < Ab_{x_0}(v_{sup})$ zu konstruieren. Im Falle $j < sup$ sei $\sigma(v_j) = v^0 w^1 v^1 \dots w^k v^k$, dann setzen wir

$$\begin{aligned} & x(v^j, s_{bpo})(a) = x_0(v^j, s_{bpo})(a) - 1, \\ \forall i \;:\; & x(v^i, w^i)(a) = x_0(v^i, w^i)(a) + 1, \\ \forall i \;:\; & x(v^i, w^{i+1})(a) = x_0(v^i, w^{i+1})(a) - 1, \\ sonst \;:\; & x(v, v')(a') = x_0(v, v')(a'). \end{aligned}$$

Im Falle $j = sup$ setzen wir $x(v^j, s_{bpo})(a) = x_0(v^j, s_{bpo})(a) - 1$ und $x(v, v')(a') = x_0(v, v')(a')$ sonst. Dies widerspricht der Wahl von x_0. Diese Behauptung zeigt insbesondere, dass $s_{bpo} \notin C$, i.e. $C \subseteq V$.

Behauptung 3: $\forall v \in V^* : (\exists w \in C : v <^* w) \iff v \in D$

$\Longrightarrow$: Sei $w \in C$ mit $v <^* w$. Wir konstruieren eine Folge $\sigma(v) = v_{sup} \dots v$, welche (C1) und (C2) erfüllt. Nach der Definition von C gibt es einen Knoten $v' \in D$ mit $x_0(v', w)(a) > 0$. Sei $\sigma(v') = v^0 w^1 v^1 \dots w^k v^k$. Im Falle $v = v^j$ für ein $j \in \{0, \dots, k\}$ folgt $v \in D$. Wir unterscheiden die folgenden verbleibenden Fälle:

- $(\exists j \in \{0, \dots, k\} : w^j = w)$: Dann erfüllt $v_{sup} w^1 v^1 \dots w^j v$ die Eigenschaften (C1) und (C2).
- $(\forall j \in \{0, \dots, k\} : w^j \neq w)$: Dann erfüllt $v_{sup} w^1 v^1 \dots w^k v' w v$ die Eigenschaften (C1) und (C2).

$\Longleftarrow$: Sei $v \in D$, dann finden wir folgendermaßen $w \in C$ mit $v <^* w$. Falls $v = v_{sup}$, dann gibt es $w = v_j$, $j > sup$, $x_0(v_{sup}, v_j)(a) > 0$. Nach Behauptung 1 folgt dann $v_j \notin D$ und somit $w = v_j \in C$. Falls $v \neq v_{sup}$, dann betrachten wir $\sigma(v) = v_{sup} w^1 v^1 \dots w^k v^k$. Nach der Definition von C kann der Knoten w_k entweder in C oder in D sein, da $x_0(v^{k-1}, w^k)(a) > 0$ und $v^{k-1} \in D$. Wir unterscheiden diese beiden Fälle:

- $w^k \in C$: Dann gilt $v = v^k <^* w^k \in C$.
- $w^k \in D$: Dann betrachten wir einen bzgl. $<^*$ maximalen Knoten $\tilde{v}$ in der Menge $\{v' \in D \mid v <^* v'\}$ (diese Menge ist nicht leer, da w^k in ihr enthalten ist). Falls $\tilde{v} = v_{sup}$, dann gibt es, wie schon gezeigt, ein w mit $v <^* \tilde{v} <^* w \in C$. Ansonsten sei $\sigma(\tilde{v}) = v_{sup} \tilde{w}^1 \tilde{v}^1 \dots \tilde{w}^l \tilde{v}^l$. Dann gilt $\tilde{w}^l \notin D$, da ansonsten $\tilde{v}$ nicht maximal wäre. Somit folgt $\tilde{w}^l \in C$, da $x_0(\tilde{v}^{l-1}, \tilde{w}^l)(a) > 0$ und $\tilde{v}^{l-1} \in D$. Folglich gilt $v <^* \tilde{v} <^* \tilde{w}^l \in C$.

Behauptung 3 zeigt insbesondere, dass C eine Co-Menge ist, da $v <^* w \in C \implies v \in D \implies v \notin C$. Außerdem folgt $D = \{v \in V^* \mid v <^* C\}$, i.e. $D' = D \setminus \{q_{bpo}\} = \{v \in V \mid v < C\}$.

Behauptung 4: C *erfüllt* $(*)$

Falls $sup = 0$ und damit nach Behauptung 1 die Beziehung $D = \{v_0\}$ gilt, folgt mit Behauptung 3 für $v \in C$ die Gleichung $f_1(W(p, l(v))) = Zu_{x_0}(v) = x_0(v_0, v)$. Damit folgt $\sum_{v \in C} f_1(W(p, l(v)))(a) = \sum_{v \in C} x_0(v_0, v)(a) = Ab_{x_0}(v_0)(a)$ unter Verwendung der Definition von C. Nach Voraussetzung gilt $Ab_{x_0}(v_0)(a) > m_0(p)(a)$ und somit entsprechend der letzten Gleichung $\sum_{v \in C} f_1(W(p, l(v)))(a) > m_0(p)(a)$. Mit

$D = \{v_0\}$ folgt weiter $m_0(p)(a) + \sum_{v \in D'} f_2(W(p, l(v)))(a) - \sum_{v \in D'} f_1(W(p, l(v)))(a) < \sum_{v \in C} f_1(W(p, l(v)))(a)$. Dies zeigt $(*)$.
Sei nun $sup > 0$. Dann gilt $Ab_{x_0}(v_0) = m_0(p)$ und entsprechend Behauptung 1 auch $\forall v \in D \setminus \{v_0, v_{sup}\} : Ab_{x_0}(v) = f_2(W(p, l(v)))$. Weiter gilt nach Annahme die Beziehung $\forall v \in D' \cup C : Zu_{x_0}(v) = f_1(W(p, l(v)))$ und $Ab_{x_0}(v_{sup})(a) > f_2(W(p, l(v_{sup})))(a)$. Insgesamt berechnen wir $m_0(p)(a) + \sum_{v \in D'} f_2(W(p, l(v)))(a) - \sum_{v \in D'} f_1(W(p, l(v)))(a) - \sum_{v \in C} f_1(W(p, l(v)))(a) < \sum_{v \in D}(\sum_{v <^* v'} x_0(v, v')(a)) - \sum_{v \in D}(\sum_{v' <^* v} x_0(v', v)(a)) - \sum_{v \in C}(\sum_{v' <^* v} x_0(v', v)(a)) = 0$. Wir weisen im Folgenden noch die Gültigkeit der letzten Umformung nach. Sie gilt, da jeder Summand $x_0(v, v')(a)$ entweder 0 ist (Fall 1) oder genau einmal positiv und einmal negativ einfließt (Fall 2). Mit Behauptung 3 lässt sich einsehen, dass nur Summanden $x_0(v, v')(a)$ mit $v \in D$ vorkommen. Für $(v, v') \in D \times (D \cup C)$ gilt Fall 2. Für $(v, v') \in D \times (V^* \setminus D)$ mit $x_0(v, v')(a) > 0$ gilt per Definition $v' \in C$. In der Situation, dass Fall 1 nicht zutrifft, ist somit Fall 2 sichergestellt. Folglich ist obige Ungleichung gültig und wir erhalten $m_0(p)(a) + \sum_{v \in D'} f_2(W(p, l(v)))(a) - \sum_{v \in D'} f_1(W(p, l(v)))(a) < \sum_{v \in C} f_1(W(p, l(v)))(a)$. Dies zeigt wiederum $(*)$.
Somit ergibt sich also, wie oben schon erklärt, mit Lemma 5.3.3 ein Widerspruch zur Aktiviertheit von bpo, da sich folgern lässt, dass $|C|_l$ in der Folgemarkierung des Präfixes bpo' der Co-Menge C bzgl. bpo nicht aktiviert ist. □

Der Satz zeigt, dass die Existenz eines geeigneten Markenflusses auch in dem hier gewählten allgemeinen Kontext eine notwendige Bedingung für die Aktiviertheit einer BPO ist. Für eine äquivalente Charakterisierung der Aktiviertheit von BPOs durch Markenflüsse müsste wie im Falle von S/T-Netzen auch eine Umkehrung der Aussage des letzten Satzes gelten, d.h. aus der Existenz eines entsprechenden Markenflusses müsste sich die Aktiviertheit einer BPO folgern lassen. Diese Umkehrung gilt aber nicht und lässt sich auch durch Zusatzforderungen nicht ohne Weiteres sicherstellen. Der Grund hierfür kann mithilfe der Ausführungen aus Unterabschnitt 5.2.1 deutlich gemacht werden. Hier werden zwar BGOs im Rahmen von STI-Netzen mit der a-priori Semantik betrachtet, analoge Zusammenhänge lassen sich aber auch für BPOs im Rahmen von STI-Netzen mit der a-posteriori Semantik herleiten. Für diese Netzklasse stellt die Existenz eines geeigneten Markenflusses zwar eine notwendige Bedingung für die Aktiviertheit dar, ist alleine aber noch keine hinreichende Bedingung. Um die Inhibitorkanten zu berücksichtigen, muss hierzu noch die entsprechende Zusatzbedingung (INH) für die Inhibitorgewichte berücksichtigt werden. Auch für andere Netzklassen wie Netze mit Lesekanten und Netze mit Kapazitäten ist, wie in [156] beschrieben, eine Charakterisierung der Aktiviertheit von BPOs durch Markenflüsse nur durch die Berücksichtigung geeigneter Netzklassen-spezifischer Zusatzbedingungen möglich. Die Zusatzbedingungen zur Existenz geeigneter Markenflüsse, welche eine Charakterisierung der Aktiviertheit erlauben, sind also für unterschiedliche Netzklassen verschieden. Diese verschiedenen Zusatzbedingungen lassen sich auf der allgemeinen Ebene der Fluss-Netztypen, welche viele verschiedene Netzklassen umfasst, nicht einheitlich zusammenfassen. Daher diskutieren wir hier Einschränkungen von Fluss-Netztypen, für die wir eine äquivalente Charakterisierung der Aktiviertheit von BPOs durch Markenflüsse herleiten können. Hierbei ist es schwierig, geeignete Einschränkungen zu finden, welche möglichst allgemein sind.

Inbesondere sollen sie in dem Sinne nicht zu restriktiv sein, dass nicht nur mehr oder weniger S/T-Netze neu erfunden werden.
Die einfachste Situation ist, ausschließlich solche Fluss-Netztypen zu betrachten, welche auch die zur dritten Forderung aus Definition 5.3.3 umgekehrte Implikation erfüllen, wie es beispielsweise für den Fluss-Netztyp $\mathcal{N}_{st}$, also für S/T-Netze, der Fall ist. Für solche Fluss-Netztypen hängt der lokale Aktiviertheitsbegriff für lokale Ereignisse nur von der Fluss-Funktion ab. Für Netze eines solchen Typs ergibt sich daher auch ohne Zusatzbedingungen die zu Satz 5.3.1 umgekehrte Aussage.

Lemma 5.3.11

LEMMA 5.3.11
Sei $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, *ein Fluss-Netztyp, welcher folgende Eigenschaft erfüllt:* $f_1(e) \leqslant s \wedge s' = s - f_1(e) + f_2(e) \implies (s, e, s') \in \mapsto$. *Sei* (N, m_0), $N = (P, T, W)$, *ein markiertes Netz vom Fluss-Typ* $(\mathcal{N}, f)$ *und sei* $bpo = (V, <, l)$ *eine Markenfluss-BPO von* (N, m_0). *Dann ist* bpo *bzgl.* (N, m_0) *aktiviert.*

BEWEIS: Sei $\mu = (bpo^*, X)$ ein zu bpo gehöriger Markenfluss-Prozess. Wir nehmen an, dass bpo nicht aktiviert ist. Sei $bpo_p = (V_p, <_p, l_p)$ ein Präfix von bpo, welches nicht aktiviert ist und minimal mit dieser Eigenschaft ist, i.e. jedes echte Präfix von bpo_p ist aktiviert. Nach Lemma 5.3.1 gibt es eine nicht-leere Co-Menge C von bpo_p und ein Präfix $bpo' = (V', <', l')$ von C (bzgl. bpo_p und bpo) derart, dass entweder bpo' nicht aktiviert ist oder $|C|_l$ in der nach Lemma 5.3.2 eindeutigen Folgemarkierung m von bpo' nicht aktiviert ist. Da bpo' ein echtes Präfix von bpo_p ist, kann nur der zweite Fall zutreffen. Nach Lemma 5.3.10 gilt $m(p) = m_0(p) + \sum_{v \in V'} f_2(W(p, l'(v))) - \sum_{v \in V'} f_1(W(p, l'(v)))$ für $p \in P$. Wir bezeichnen $V'_m = V' \cup \{q_{bpo}\}$. Entsprechend Definition 5.3.4 gilt für $x \in X$ mit $\phi(x) = p$ Folgendes: $m(p) = Ab_x(q_{bpo}) + \sum_{v \in V'} Ab_x(v) - \sum_{v \in V'} Zu_x(v) = \sum_{v \in V'_m} (\sum_{v <^* v'} x(v, v')) - \sum_{v \in V'} (\sum_{v' <^* v} x(v', v)) = \sum_{(v,v') \in <^* \cap (V'_m \times (V^* \setminus V'_m))} x(v, v') \geqslant \sum_{(v,v') \in <^* \cap (V'_m \times C)} x(v, v') = \sum_{v \in C} Zu_x(v) = \sum_{v \in C} f_1(W(p, l(v))) = f_1(\sum_{t \in T} |C|_l(t) \cdot W(p, t))$. Mit der hier zusätzlich für $(\mathcal{N}, f)$ geforderten Eigenschaft, können wir für $m'(p) = m(p) - f_1(\sum_{t \in T} |C|_l(t) \cdot W(p, t)) + f_2(\sum_{t \in T} |C|_l(t) \cdot W(p, t))$ folgern, dass $(m(p), \sum_{t \in T} |C|_l(t) \cdot W(p, t), m'(p)) \in \mapsto$. Somit ist $|C|_l$ in m aktiviert, ein Widerspruch. □

Es ist zu beachten, dass die Anforderung an Fluss-Netztypen des letzten Lemmas die ZZE garantiert. Die Anforderung aus Lemma 5.3.11 ist aber sehr restriktiv. Sie wird beispielsweise vom Netztyp $\mathcal{N}_{sti}$ bzw. generell von Inhibitornetzen und ähnlichen Netzklassen nicht erfüllt.
Daher versuchen wir nun, eine geeignete Einschränkung für Fluss-Netztypen zu finden, welche es erlaubt, die Charakteristika von Inhibitornetzen und ähnlichen Netzklassen auf einer allgemeinen Ebene zu berücksichtigen. Eine Möglichkeit, um solche Netzklassen abzudecken, ist, Fluss-Netztypen $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, mit einer Blockade-Funktion $b : LS \times LE \to \{0, 1\}$ zu versehen, für die folgende Beziehung gilt:

$$(f_1(e) \leqslant s \wedge s' = s - f_1(e) + f_2(e) \wedge b(s, e) = 0) \iff (s, e, s') \in \mapsto .$$

Wir fordern also auch hier im Wesentlichen die zur dritten Forderung aus Definition 5.3.3 umgekehrte Implikation, allerdings unter der Berücksichtigung der Blockade-Funktion. Nimmt diese den Wert 1 an, so

verhindert sie die Aktiviertheit eines lokalen Ereignisses in einem lokalen Zustand. Im Falle, dass sie den Wert 0 hat, wird die Aktiviertheit dann wie im letzten Lemma von der Fluss-Funktion bestimmt.

BEISPIEL: *Für STI-Netze lassen sich Blockade-Funktionen folgendermaßen kanonisch definieren. Für die Fluss-Netztypen* $(\mathcal{N}_{sti}, f_{sti})$ *und* $(\mathcal{N}_{\overleftarrow{sti}}, f_{\overleftarrow{sti}})$ *setzen wir* $b_{sti}(n,(i,j,k)) = 0 \iff n+j \leqslant k$ *und* $b_{\overleftarrow{sti}}(n,(i,j,k)) = 0 \iff n \leqslant k$. *Auch für den Fluss-Netztyp* $(\mathcal{N}_{en}, f_{en})$ *der elementaren Netze lässt sich eine Blockade-Funktion durch* $b_{en}(n,(i,j)) = 0 \iff n+j \leqslant 1$ *definieren. Damit erfüllen all diese Netztypen die betrachtete Anforderung an Fluss-Netztypen.* *Beispiel*

Mit dieser Einschränkung ist die Aktiviertheit eine BPO durch die Kombination der Fluss-Funktion und der Blockade-Funktion bestimmt. Die Fluss-Funktion und die Blockade-Funktion modellieren unabhängig voneinander Bedingungen für die Aktiviertheit. Die Bedingungen der Fluss-Funktion lassen sich gemäß Lemma 5.3.11 und Satz 5.3.1 durch Markenfluss-Funktionen darstellen. Die Anforderung an die BPO ist hier, dass sie eine Markenfluss-BPO ist. Für die Bedingungen der Blockade-Funktion muss eine zusätzliche Anforderung an die BPO gestellt werden. Es wird gefordert, dass die BPO die sog. Blockade-Eigenschaft erfüllt. Diese wird derart formuliert, dass eine BPO genau dann aktiviert ist, wenn sie eine Markenfluss-BPO ist, welche die Blockade-Eigenschaft erfüllt. Die Blockade-Eigenschaft definieren wir in einer natürlicher Weise. Eine BPO bpo erfüllt die Blockade-Eigenschaft bzgl. eines markierten Netzes (N, m_0), $N = (P, T, W)$, vom Fluss-Netztyp $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, wenn es für jede nichtleere Co-Menge C von bpo und jedes Präfix bpo′ von C eine aktivierte Schrittfolge von bpo′ mit Folgemarkierung m gibt, so dass $b(m(p), \sum_{t \in T} |C|_l(t) \cdot W(p,t)) = 0$ für jede Stelle $p \in P$. Falls $\mathcal{N}$ die ZZE erfüllt, genügt es bei der Blockade-Eigenschaft, alle Schnitte anstelle aller nicht-leeren Co-Mengen zu betrachten. Eine entsprechende Charakterisierung der Aktiviertheit von BPOs ergibt sich nun als Folgerung aus Lemma 5.3.11 und Satz 5.3.1.

LEMMA 5.3.12 *Lemma 5.3.12*

Sei $(\mathcal{N}, f)$, $\mathcal{N} = (LS, LE, \mapsto)$, *ein Fluss-Netztyp mit einer zugehörigen Blockade-Funktion* b. *Sei* (N, m_0), $N = (P, T, W)$, *ein markiertes Netz vom Fluss-Netztyp* $(\mathcal{N}, f)$ *und sei* bpo $= (V, <, l)$ *eine BPO mit* $l : V \to T$. *Es gilt, dass* bpo *genau dann aktiviert ist, wenn* bpo *eine Markenfluss-BPO von* (N, m_0) *ist, welche die Blockade-Eigenschaft bzgl.* (N, m_0) *erfüllt.*

BEWEIS: Die „genau dann"-Aussage folgt aus Satz 5.3.1 und Lemma 5.3.1. Die „wenn"-Aussage lässt sich analog wie Lemma 5.3.11 beweisen, wobei zusätzlich die Blockade-Eigenschaft berücksichtigt werden muss. □

Die Blockade-Eigenschaft ist eine allgemein formulierte Anforderung, welche für viele Netzklassen, beispielsweise STI-Netze oder elementare Netze, verwendet werden kann. Für spezielle solche Netzklassen lässt sie sich häufig noch vereinfachen, indem Spezifika der Netzklassen ausgenutzt werden. Eine solche Vereinfachung findet sich beispielsweise im Rahmen der Überlegungen aus Unterabschnitt 5.2.1 zu STI-Netzen mit der a-priori Semantik (allerdings für BGOs). Weiter sei auf die Aktiviertheitsüberlegungen in [156] hingewiesen. Dort sind Vereinfachungen für Inhibitornetze und Netze mit Kapazitäten formuliert.

5.3.3 *Synthese für Netztypen*

In diesem Unterabschnitt skizzieren wir, wie sich aufbauend auf den Grundlagen der letzten beiden Unterabschnitte die Syntheseprinzipien aus Kapitel 4 auf Netztypen übertragen lassen. Wir betrachten also das folgende Problem, welches eine Verallgemeinerung des Syntheseproblems aus Kapitel 4 darstellt. Die Verallgemeinerung besteht darin, dass das Problem mithilfe von Netztypen parametrisch in der Netzklasse formuliert ist.

Problemspezifikation 5.3.1 (Syntheseproblem für Netztypen)

Problemspezifikation 5.3.1 (Syntheseproblem für Netztypen)
Eingabe: *Eine endliche partielle Sprache $\mathcal{L}$ und ein Netztyp $\mathcal{N} = (\mathsf{LS}, \mathsf{LE}, \mapsto)$.*
Ausgabe: *Ein markiertes Netz (N, m_0) vom Typ $\mathcal{N}$ mit $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$, falls ein solches Lösungsnetz existiert, und eine „*notexists*"-Meldung sonst.*

Dieses Problem soll mit den selben Verfahren wie das Syntheseproblem in Kapitel 4 gelöst werden. Zuerst einmal lässt sich der Begriff der zulässigen Stelle für Netztypen analog wie für S/T-Netze formulieren, wobei anstatt eines Stellentripels nun ein Stellenpaar betrachtet wird. In dem allgemeinen Kontext von Netztypen muss nur eine Komponente für die Anfangsmarkierung und eine Komponente für die Gewichtsfunktion des zugehörigen atomaren Netzes berücksichtigt werden. Ausgehend von dem Begriff des zulässigen Stellenpaares lassen sich dann alle weiteren in Abschnitt 4.3 für zulässige Stellentripel und S/T-Netze durchgeführten Überlegungen analog auch auf Stellenpaare von Netztypen übertragen.
Somit lässt sich sinnvoll über Verallgemeinerungen der verschiedenen Regionendefinitionen aus Abschnitt 4.3 auf Netztypen diskutieren. Den einzelnen Regionendefinitionen aus Abschnitt 4.3 liegt jeweils eine spezielle Charakterisierung der Aktiviertheit von BPOs von S/T-Netzen zugrunde. Unter geeigneten Einschränkungen gelten entsprechende Charakterisierungen, wie gezeigt, auch für Netztypen. Sind derartige Einschränkungen gewährleistet, so lassen sich die entsprechenden Regionendefinitionen analog auf Netztypen übertragen.
Schritt-Transitions-Regionen liegt die ursprüngliche Aktiviertheits-Charakterisierung basierend auf Schrittfolgen zugrunde, welche wir auch für Netztypen formuliert haben. Daher lässt sich die Definition einer Schritt-Transitions-Region analog auf Netztypen übertragen, wobei eine Region in diesem Fall dann ein Vektor mit einem LS-Eintrag und m Einträgen aus LE ist und die Forderung (ST) entsprechend der Schaltregel für Netztypen abgeändert werden muss.
BPO-Transitions-Regionen liegt die Charakterisierung der Aktiviertheit basierend auf Schnitten zugrunde. Eine entsprechende Charakterisierung haben wir in Lemma 5.3.8 für Netztypen, welche die ZZE erfüllen, formuliert. Für solche Netztypen lässt sich dann also ähnlich wie im Falle von Schritt-Transitions-Regionen eine Definition von BPO-Transitions-Regionen formulieren. Hier besteht allerdings noch das Problem, dass in einer entsprechenden Definition zur Berücksichtigung der Folgemarkierung von Präfixen von Schnitten Schrittfolgen der Präfixe verwendet werden müssen. Diese müssen also in die Forderung (BPOT) einfließen. Dies kann analog wie bei Schritt-Transitions-Regionen durchgeführt werden. Für Netztypen, welche die PBE erfüllen wie beispielsweise Fluss-Netztypen ist dies häufig nicht nötig und es kann ähnlich wie in der Formulierung von (BPOT) für S/T-Netze die

Berechnung der Folgemarkierung von Präfixen von Schnitten ohne einen Zwischenschritt über Schrittfolgen durchgeführt werden. Insbesondere bei Fluss-Netztypen ist dies der Fall. Hier lässt sich mit Lemma 5.3.10 eine zur Forderung (BPOT) für S/T-Netze analoge Forderung für Netztypen gewinnen.
Markenfluss-Regionen verwenden schließlich die Äquivalenz von aktivierten BPOs und Markenfluss-BPOs. Für Fluss-Netztypen gilt hier die Implikation, dass eine aktivierte BPO eine Markenfluss-BPO ist. Für eine Charakterisierung der Aktiviertheit von BPOs sind zusätzliche Einschränkungen notwendig. Hierzu können die Voraussetzungen aus Lemma 5.3.11 verwendet werden. In diesem Fall gilt, dass eine Markenfluss-BPO aktiviert ist. Dann lässt sich für Fluss-Netztypen aufbauend auf den Definitionen des letzten Unterabschnittes vollkommen analog wie in Kapitel 4 die Konsistenz von atomaren partiell geordneten Markenflüssen, also die Forderungen (ANF), (ZU) und (AB), definieren und damit eine sinnvolle Definition von Markenfluss-Regionen ableiten.
Eine weitere Möglichkeit für eine Charakterisierung der Aktiviertheit von BPOs stellt Lemma 5.3.12 dar. In diesem Fall gilt, dass eine BPO genau dann aktiviert ist, wenn sie eine Markenfluss-BPO ist, welche die Blockade-Eigenschaft erfüllt. Zur Definition von Markenfluss-Regionen kann dann genauso vorgegangen werden wie gerade erklärt, nur dass zusätzlich noch die Blockade-Eigenschaft sichergestellt werden muss. Hierzu kann eine Zusatzforderung ähnlich der Forderung (INH) für Markenfluss-Regionen von STI-Netzen formuliert werden (vgl. Unterabschnitt 5.2.1). Die Folgemarkierung von Präfixen kann dabei mit Lemma 5.3.10 analog wie für STI-Netze diskutiert aus dem Markenfluss berechnet werden.
Um nun aufbauend auf einer Regionendefinition für Netztypen ein Verfahren zur Lösung des betrachteten Syntheseproblems wie in Kapitel 4 zu entwickeln, ist eine linear algebraische Charakterisierung der Regionendefinition nötig. Eine solche lässt sich aber auf der allgemeinen Ebene von Netztypen nicht gewinnen. Auf der Ebene von Netztypen ist gar nicht von irgendwelchen Zahlenwerten die Rede, sondern die Grundlage von Netztypen bilden Monoide. Daher ist für Netztypen im Allgemeinen eine Übersetzung der Anforderungen von entsprechenden Regionendefinitionen in konkrete Ungleichungen nicht möglich. Eine linear algebraische Charakterisierung muss somit spezifisch für spezielle Netztypen formuliert werden. Dabei können natürlich u.U. ähnliche Netztypen einheitlich behandelt werden. Beispielsweise lassen sich unter Voraussetzung von Lemma 5.3.11 Markenfluss-Regionen, welche dann analog wie in Kapitel 4 definiert sind, auch analog wie in Kapitel 4 linear algebraisch charakterisieren, indem jede Komponente des freien kommutativen Monoids LS einzeln betrachtet wird. Normalerweise schlagen wir hier jedoch vor, basierend auf den allgemeinen parametrischen Regionendefinitionen für jeden zu betrachtenden Netztyp einzeln eine linear algebraische Charakterisierung zu entwickeln. Dies ist sinnvoll, da sich konkrete Berechnungsverfahren typischerweise nicht auf der allgemeinen algebraischen Ebene widerspiegeln lassen.
Ist schließlich für einen konkreten Netztyp eine linear algebraische Charakterisierung einer Regionendefinition gewonnen, so sollte dann geprüft werden, inwiefern mit dieser Charakterisierung die Verfahren aus Abschnitt 4.4 zur Berechnung einer geeigneten Repräsentation eines gesättigt zulässigen Netzes verwendbar sind. Hier sind möglicherweise leichte Modifikationen notwendig, generell sind die vorge-

stellten Verfahren aber auch in diesem Rahmen die Kandidaten für ein zielführendes Berechnungsverfahren. Nach der Berechnung eines Repräsentationsnetzes ist dann wiederum ein Übereinstimmungstest notwendig, um zu prüfen, ob das synthetisierte Netz das spezifizierte Verhalten aufweist. Hierzu bieten sich als Kandidaten auch wieder die entsprechenden Verfahren aus Kapitel 4 an. Dabei müssen auch hier bei den einzelnen Berechnungsschritten Spezifika des betrachteten Netztyps berücksichtigt werden.
Insgesamt ist es also zur Lösung des Syntheseproblems aus Spezifikation 5.3.1 an einigen Stellen notwendig, individuelle, vom betrachteten Netztyp abhängige, Berechnungsschritte zu entwickeln, da das Konzept des Netztyps für spezielle Berechnungen zu abstrakt ist. Jedoch lassen sich zumindest Regionendefinitionen, welche die Basis für alle Syntheseverfahren darstellen, auf allgemeiner Ebene für Netztypen formulieren.

5.4 VARIANTEN DER FRAGESTELLUNG

Wie erklärt, stellen Szenarien in frühen Modellierungsphasen häufig das intuitivste und einfachste Modellierungskonzept für Systeme verschiedenster Anwendungsbereiche dar. Dies liegt insbesondere daran, dass Benutzer einzelne Szenarien häufig besser kennen als das Gesamtsystem. Daher ist es eine typische Situation, dass das Verhalten eines Systems in der Form von halbgeordneten Abläufen modelliert wird. Allerdings ist eine solche Szenario-Spezifikation für die finalen Modellierungsziele weniger geeignet. Zu Zwecken der Dokumentation, der Analyse, der Simulation, der Optimierung, des Designs oder schließlich der Implementierung eines Systems sind normalerweise integrierte zustandsbasierte Modelle erwünscht. Um die Kluft zwischen der Szenario-Sicht eines Systems und einem integrierten Systemmodell zu überbrücken, gibt es die wichtige Herausforderung, Verfahren zu entwickeln, welche ein Systemmodell automatisch aus einer Szenario-Spezifikation eines Systems konstruieren.
Mit diesem Problem beschäftigen sich die Syntheseverfahren dieser Arbeit, wobei Petrinetze als Systemmodell und BPOs zur Modellierung von Szenarien verwendet werden. Das bisher betrachtete klassische Syntheseproblem stellt die Frage, ob es ein Netz gibt, welches das durch die spezifizierten BPOs gegebene Verhalten aufweist, und im positiven Fall erfordert das Problem die Konstruktion eines solchen Netzes. Allerdings entspricht, wie schon in Abschnitt 4.1 diskutiert, dieses klassische Syntheseproblem häufig nicht genau den praktischen Problemstellungen in realistischen Anwendungskontexten. Dies liegt daran, dass die tatsächlichen praktischen Fragestellungen sehr vielfältig sind. Häufig müssen problemspezifische Anforderungen berücksichtigt werden, so dass die Standard-Syntheseverfahren zur Lösung der Probleme nicht ausreichen. Aus diesem Grund diskutieren wir in diesem Abschnitt Varianten der klassischen Synthesefragestellung, welche viele typische praktische Anforderungen an Systemmodelle abdecken. Dies führt zu einem großen Repertoire an Syntheseverfahren, welches für verschiedenste Anwendungsmöglichkeiten von Synthese entsprechende Lösungsansätze zur Verfügung stellt.
Die wesentlichen Beschränkungen der klassischen Synthesefragestellung im Hinblick auf Anwendbarkeit werden im Folgenden an Beispielen veranschaulicht. In diesem Zusammenhang motivieren wir die

Varianten der klassischen Fragestellung, welche wir im weiteren Verlauf des Abschnittes im Detail diskutieren. Wir betrachten hierzu wieder die partielle Sprache aus Abbildung 4. Die erste Frage im Rahmen des klassischen Syntheseproblems ist nun, ob es ein Netz mit dem spezifizierten Ablaufverhalten gibt. Da dies der Fall ist, muss dann ein entsprechendes Netz berechnet werden. Ein solches Netz ist beispielsweise das Netz aus Abbildung 3. Aber diese Lösung ist natürlich nicht eindeutig. Auch die Netze der Abbildungen 55 und 42 sind Lösungsnetze. Wie schon im Rahmen des gesättigt zulässigen Netzes diskutiert, gibt es tatsächlich unendlich viele zulässige Stellen und damit auch unendlich viele mögliche Lösungsnetze für unsere Beispiel-Sprache. Nun kann für bestimmte Anwendungen nicht jedes dieser Netze geeignet sein. Daher beschäftigen sich einige der modifizierten Fragestellungen dieses Abschnittes mit entsprechenden zusätzlichen Anforderungen an das zu synthetisierende Netz. Auf diese Weise kann ausgeschlossen werden, dass ein besonders ungünstiges Lösungsnetz berechnet wird. Beispielsweise darf ein Netz für viele Anwendungen nicht zu groß sein. In diesem Fall ist es sinnvoll, eine obere Schranke für die Größe des Netzes vorauszusetzen. Dies kann über eine Schranke für die Anzahl der Stellen des Netzes realisiert werden (Variante Stellen-Schranke). Durch eine solche lässt sich in obigem Beispiel ausschließen, dass ein Netz mit vielen Stellen wie das aus Abbildung 42 berechnet wird. In diesem Zusammenhang ist es auch möglich, nach einem Lösungsnetz mit einer minimalen Anzahl an Stellen zu suchen. Im Beispiel wäre das Netz aus Abbildung 55 eine solche Lösung. Anstelle der Betrachtung der Anzahl an Stellen lassen sich mit mathematischen Optimierungsverfahren auch die Kantengewichte und die Markierungen des Lösungsnetzes minimieren (Variante Optimierung). Im Beispiel kann mit einem solchen Verfahren das Netz aus Abbildung 3 berechnet werden. Dieses ist bzgl. der Kantengewichte und Markierungen kleiner als das Netz aus Abbildung 55, obwohl es mehr Stellen besitzt. In etlichen Anwendungen spielen schließlich spezielle Anforderungen an die Zustände des zu modellierenden Systems eine Rolle. In diesem Fall kann gefordert werden, dass bestimmte Abläufe des zu synthetisierenden Netzes zu den selben Markierungen und damit zu den selben Zuständen führen (Variante Zustände). Beispielsweise kann für obige partielle Sprache gefordert werden, dass die Endmarkierung des synthetisierten Netzes nach den zwei spezifizierten Abläufen bpo_1 und bpo_2 übereinstimmt. Diese Synthesefrage hat allerdings eine negative Antwort, da es kein Netz mit dieser Eigenschaft, welches das spezifizierte Ablaufverhalten aufweist, gibt.

Nun führen wir das Beispiel noch einen Schritt weiter. Wir gehen von der partiellen Sprache in Abbildung 54 aus. Für diese muss die Frage nach einem Netz mit dem spezifizierten Verhalten negativ beantwortet werden. In einem solchen Fall gibt es bzgl. der klassischen Problemspezifikation keine Anforderungen an einen Synthesealgorithmus, welche sicherstellen, dass ein sinnvolles Systemmodell erzeugt wird. Tatsächlich gibt es nicht einmal die Anforderung, dass überhaupt ein Netz berechnet wird. Für praktische Anwendungen liegt der Fokus im Rahmen der Verwendung eines Syntheseverfahrens aber häufig auf der Konstruktion eines geeigneten Systemmodells, und nicht auf der Entscheidung des Syntheseproblems. Es soll also, auch wenn das Syntheseproblem eine negative Antwort besitzt, ein Netz erzeugt werden, welches das spezifizierte System sinnvoll modelliert. Daher betrachten

wir hier auch Synthesefragestellungen, welche von einem Entscheidungsproblem abstrahieren und sich stattdessen mit der Konstruktion eines geeigneten Netzes für beliebige Szenario-Spezifikationen beschäftigen. Ein Beispiel hierfür ist die Konstruktion einer in einem gewissen Sinne besten Approximation für das gewünschte Verhalten durch ein Netz. Hierzu eignet sich ein Netz mit minimalem zusätzlichen Verhalten (Variante beste obere Approximation). Dann wird auch für den Fall, dass sich das Verhalten nicht exakt abbilden lässt, ein Modell mit sehr ähnlichem Verhalten berechnet. Für obiges Beispiel ist solch ein Netz in Abbildung 57 gegeben. Ähnlich kann auch ein Netz mit minimal weniger Verhalten als spezifiziert betrachtet werden (Variante beste untere Approximation). Ein entsprechendes Beispiel-Netz für die betrachtete partielle Sprache ist das Netz aus Abbildung 3. Ein wichtiger Punkt im Rahmen der Überlegungen dieses Absatzes ist, dass Spezifikationen generell dazu tendieren, unvollständig zu sein. Der Fall, dass das klassische Syntheseproblem eine negative Antwort besitzt, ist also ganz typisch. In diesem Fall lässt sich allerdings kaum sagen, wie ein geeignetes Systemmodell aussehen soll. Ein sinnvolles Vorgehen zur Konstruktion eines Netzes ist also schwierig, da wir hier insbesondere auch keine heuristischen Verfahren und entsprechende Problemstellungen betrachten wollen. Allerdings ist die Forderung nach einer exakten Reproduktion der Spezifikation, wie sie im klassischen Syntheseproblem gefordert wird, für unvollständige Spezifikationen offensichtlich zu restriktiv. Ein sinnvoller Ansatz, um im Rahmen einer exakten Fragestellung solche Situationen zu behandeln, besteht darin, dass wir die strenge Forderung des klassischen Syntheseproblems abschwächen. Eine Möglichkeit hierzu ist die schon diskutierte Möglichkeit der Suche nach einer approximativen Lösung. Eine andere Möglichkeit, welche wir ebenfalls als eine Synthesevariante dieses Abschnittes diskutieren, ist die Vorgabe eines Toleranzbereichs für das spezifizierte Verhalten. Dies kann durch die Angabe einer unteren und oberen Schranke für das Verhalten des Netzes erzielt werden (Variante Sprach-Schranken). Beispielsweise könnte die partielle Sprache aus Abbildung 54 nur die obere Schranke darstellen, während die partielle Sprache aus Abbildung 4 eine untere Schranke für das Ablaufverhalten des gesuchten Netzes repräsentiert. Dann wäre natürlich wiederum das Netz aus Abbildung 3 ein Lösungsnetz für das Problem.
Wir sind in diesem Abschnitt also an Varianten der klassischen Synthesefragestellung interessiert. Dabei werden sowohl Varianten betrachtet, welche die Anforderungen an das zu synthetisierende Netz abschwächen, als auch Varianten, welche zusätzliche Anforderungen an das Netz stellen. Die Varianten berücksichtigen typische über die klassische Fragestellung hinausgehende Syntheseaspekte. Es ergibt sich hier also ein im Gegensatz zum klassischen Syntheseproblem abgeänderter funktionaler Zusammenhang zwischen Eingabe und Ausgabe. Für die Ausgabe wird nicht mehr ein Netz gesucht, welches exakt die Abläufe der Eingabe widerspiegelt. Hier werden nun andere Zusammenhänge des Netzes zur Eingabe berücksichtigt. Dabei sind bei einigen Varianten zusätzlich zu einer partiellen Sprache weitere Eingabeparameter notwendig.
Ein wichtiger Punkt bei der Lösung der modifizierten Synthesefragestellungen ist, dass sich alle betrachteten Varianten mit einem ähnlichen Vorgehen wie das klassische Problem lösen lassen. Wir skizzieren hier für jede Variante umfassend mögliche Lösungsverfahren.

Für diese lassen sich die ursprünglichen Regionendefinitionen aus Kapitel 4 direkt verwenden. Wir gehen in diesem Abschnitt jeweils nur auf die Verwendung der intuitiven Schritt-Transitions-Regionen ein. Für BPO-Transitions-Regionen und Markenfluss-Regionen lässt sich analog vorgehen. Für diese Regionendefinitionen ändern sich nur die jeweils zu betrachtenden Ungleichungssysteme entsprechend der linear algebraischen Charakterisierungen. Im Gegensatz zu den Regionendefinitionen müssen die Algorithmen zur Berechnung von Repräsentationsnetzen aus Kapitel 4 für die verschiedenen Varianten der Synthesefragestellung allerdings geeignet adaptiert werden. Auch bestimmte Zusatzverfahren wie die Übereinstimmungstests aus Kapitel 4 sind in einigen Fällen wieder notwendig. Wir zeigen hierzu für jede Variante die wesentlichen Ideen zur Anpassung der Synthesealgorithmen aus Abschnitt 4.4 an die neue Problemstellung. Dabei gehen wir insbesondere darauf ein, inwiefern das Verfahren zur Berechnung einer Schrittseparations-Repräsentation und das Verfahren zur Berechnung einer Erzeugendensystem-Repräsentation zur Lösung der Varianten abgeändert werden müssen. Auf die Anwendung des Verfahrens zur Berechnung einer BPO-Separations-Repräsentation gehen wir nicht näher ein, da sich die Überlegungen im Rahmen der Berechnung einer Schrittseparations-Repräsentation analog wie in Abschnitt 4.4 auf die Berechnung einer BPO-Separations-Repräsentation übertragen lassen. Es ist hier noch anzumerken, dass das Grundprinzip der Berechnung einer Schrittseparations-Repräsentation für alle Varianten genutzt werden kann, dies aber für die Berechnung einer Erzeugendensystem-Repräsentation nicht der Fall ist.
Insgesamt betrachten wir in diesem Abschnitt die sechs Hauptvarianten der klassischen Synthesefragestellung, die wir in obigen Ausführungen angesprochen haben. Dabei diskutieren wir auch einige interessante Versionen dieser Varianten. Jede der sechs Varianten wird in einem Unterabschnitt behandelt. Es wird jeweils gezeigt, wie sich die Variante mit den Mitteln des letzten Kapitels lösen lässt. In einem weiteren Unterabschnitt wird der Vollständigkeit halber kurz noch eine Reihe weiterer Varianten angesprochen. Während einige der diskutierten Varianten völlig neu sind, wurden zu anderen Varianten ähnliche Fragestellungen schon in anderen Bereichen der Theorie der Synthese von Petrinetzen behandelt. Aber in letzterem Fall wurden wenn überhaupt bisher meist nur sehr problemspezifische Lösungsverfahren angegeben, welche zum einen nicht ohne Weiteres auf die Synthese aus partiellen Sprachen übertragbar sind und zum anderen nicht die verschiedenen möglichen Lösungsverfahren, also insbesondere die verschiedenen Repräsentations-Berechnungen, aus Kapitel 4 berücksichtigen. Auf derartig verwandte Ansätze wird in den jeweiligen Unterabschnitten eingegangen.

5.4.1 *Variante Sprach-Schranken*

Das Ablaufverhalten eines zu modellierenden Systems lässt sich manchmal nicht exakt spezifizieren. Dies kann an Design-Alternativen oder einem gewissen Design-Spielraum bei der Systemmodellierung liegen, aber auch an unvollständigen Informationen über das System oder Unsicherheiten bzgl. des Verhaltens des Systems. Dieses Problem trifft insbesondere auf frühe Modellierungsphasen zu, in denen die Informationen über das System typischerweise noch vage sind und das

Modellierungsziel noch eher die Erstellung eines Prototyps ist. In solchen Fällen ist es wünschenswert, einen gewissen Toleranzbereich für das Ablaufverhalten des Systems zu spezifizieren. Anstelle der Spezifikation einer partiellen Sprache, welche exakt das Verhalten eines zu synthetisierenden Netzes darstellen soll, sind wir daran interessiert, das nach unserem Wissen minimal und maximal mögliche Ablaufverhalten des zu synthetisierenden Netzes durch zwei partielle Sprachen zu spezifizieren. Diese zwei Sprachen repräsentieren also eine untere und eine obere Schranke für das Ablaufverhalten eines Netzes. Es wird nach einem Netz gesucht, welches mindestens die Abläufe der unteren Schranke aufweist und nur Abläufe der oberen Schranke beinhaltet. Wir suchen also einen Algorithmus, welcher folgender Spezifikation genügt.

Problemspezifikation 5.4.1 (Syntheseproblem mit Sprach-Schranken)

PROBLEMSPEZIFIKATION 5.4.1 (SYNTHESEPROBLEM MIT SPRACH-SCHRANKEN)
Eingabe: *Zwei endliche partielle Sprachen $\mathcal{L}, \mathcal{L}'$ mit $\mathcal{L} \subseteq \mathcal{L}'$.*
Ausgabe: *Ein markiertes S/T-Netz (N, m_0) mit $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0) \subseteq PS(\mathcal{L}')$, falls ein solches Lösungsnetz existiert, und eine „*notexists*"-Meldung sonst.*

Zur Lösung dieses Problems können wie bisher Regionen von $\mathcal{L}$ betrachtet werden, da als Lösungsnetze nur Netze mit ausschließlich bzgl. $\mathcal{L}$ zulässigen Stellen in Frage kommen. Für falsche Fortsetzungen oder einen Übereinstimmungs- bzw. Enthaltenstest wird allerdings $\mathcal{L}'$ verwendet. In [63, 64] wird die Sprach-Schranken-Variante für klassische reguläre Sprachen betrachtet und im Rahmen der Berechnung einer Art Erzeugendensystem-Repräsentation gelöst.
In unserer Situation lässt sich die Variante auch durch Berechnung einer Erzeugendensystem-Repräsentation (N, m_0) von $\mathcal{L}$ lösen. Ein Test, ob $\mathfrak{Bpo}(N, m_0) \subseteq PS(\mathcal{L}')$, kann mit den Übereinstimmungstest-Verfahren aus Abschnitt 4.5 durchgeführt werden. Ein solcher Test löst das betrachtete Problem, da (N, m_0) entsprechend Lemma 4.4.10 eine beste obere Approximation eines Netzes an das durch $\mathcal{L}$ spezifizierte Verhalten darstellt. Gilt für (N, m_0) die Beziehung $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0) \subseteq PS(\mathcal{L}')$ nicht, so kann diese auch für kein anderes markiertes S/T-Netz gelten. Auch das Verfahren zur Berechnung einer Schrittseparations-Repräsentation lässt sich derart anpassen, dass es Problemspezifikation 5.4.1 genügt. Hierzu wird für jede falsche Schrittfortsetzung $\sigma = \tau_1 \dots \tau_j \in \mathcal{L}'_{ws}$ bzgl. $\mathcal{L}'$ nach einer schrittseparierenden Region bzgl. $\mathcal{L}$ gesucht und, falls eine solche existiert, ein entsprechend zulässiges Stellentripel hinzugefügt. Falls es keine solche Region gibt, sind zwei Fälle möglich. In dem Fall $\tau_1 \dots \tau_{j-1} \in \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}$ folgt, dass das Problem eine negative Antwort besitzt, da dann in jedem Netz, dessen Ablaufverhalten $PS(\mathcal{L})$ beinhaltet, die Schrittfolge $\sigma \notin \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}'))\}$ aktiviert ist. In dem Fall $\tau_1 \dots \tau_{j-1} \notin \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}$ kann diese Schrittfolge $\tau_1 \dots \tau_{j-1}$ einfach als zu verhindernde falsche Fortsetzung betrachtet werden, i.e. sie wird zur Menge der betrachteten falschen Schrittfortsetzungen $\mathcal{L}'_{ws}$ hinzugefügt. Die Idee ist hierbei, ein Präfix der falschen Fortsetzung σ zu verhindern. Dies ist natürlich nur möglich, wenn das entsprechende Präfix nicht durch $\mathcal{L}$ spezifiziert ist. Wenn der erste der beiden betrachteten Fälle niemals eintritt, so wird für jede falsche Schrittfortsetzung bzgl. $\mathcal{L}'$ ein Präfix ausgeschlossen. In dieser Situation lässt sich folgern, dass für das konstruierte Netz (N, m_0) gilt,

dass es entweder das betrachtete Problem positiv löst oder es kein Lösungsnetz gibt. Dies kann entschieden werden, indem wiederum $\mathfrak{Bpo}(N, m_0) \subseteq PS(\mathcal{L}')$ mit den Übereinstimmungstest-Verfahren aus Abschnitt 4.5 geprüft wird.

Version Steuerung: In [64, 67] wird diskutiert, dass ein solches Problem insbesondere im Rahmen der Theorie und der Praxis der Steuerungssynthese relevant ist [189]. Die Steuerungssynthese ist ein typisches Problemfeld, in dem ein gewisser Spielraum für das Verhalten des Zielsystems vorgegeben wird. Im einfachsten Fall spezifizieren zwei Schranken $\mathcal{L}, \mathcal{L}'$ das minimale und maximale erwartete Verhalten eines Ausgangssystems, welches durch ein Netz (N_p, m_p) mit $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N_p, m_p)$ gegeben ist. Es wird dann nach einem Netz (N_c, m_c) gesucht, welches das Ausgangssystem derart steuert, dass $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}((N_p, m_p) \times (N_c, m_c)) \subseteq PS(\mathcal{L}')$, wobei $(N_p, m_p) \times (N_c, m_c)$ das Netz ist, welches sich durch Verschmelzen der Transitionen von (N_p, m_p) und (N_c, m_c) ergibt. Diese Fragestellung lässt sich lösen, indem das betrachtete Syntheseproblem mit den Sprach-Schranken $\mathcal{L}, \mathcal{L}'$ gelöst wird. Falls dieses eine positive Antwort besitzt, so stellt ein entsprechendes Lösungsnetz ein geeignetes Steuerungsnetz dar. Ansonsten gibt es kein zugehöriges Steuerungsnetz. Fortschrittlichere Fragestellungen im Rahmen der Steuerungssynthese, welche sich teilweise auch auf Petrinetz-Syntheseprobleme zurückführen lassen, finden sich beispielsweise in [15, 66, 193, 21, 67].

Version Unerwünscht: Ein typisches Szenario ist, dass die obere Schranke $\mathcal{L}'$ nicht direkt, sondern indirekt beispielsweise durch eine endliche Menge S von unerwünschten Schrittfolgen gegeben ist. Es wird also ein Netz (N, m_0) gesucht mit $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$, in dem keine Schrittfolge aus S aktiviert ist. Im Falle des Vorgehens mithilfe einer Erzeugendensystem-Repräsentation (N, m_0) bzgl. $\mathcal{L}$ kann dann einfach der Test $\mathfrak{Bpo}(N, m_0) \subseteq PS(\mathcal{L}')$ dadurch ersetzt werden, dass für jede Schrittfolge $\sigma \in S$ geprüft wird, ob sie bzgl. (N, m_0) aktiviert ist. Ist dies für eine Schrittfolge der Fall, so kann wiederum mit Lemma 4.4.10 eine negative Antwort auf die neue Fragestellung gegeben werden. Ist dies für keine Schrittfolge der Fall, so löst (N, m_0) das Problem positiv. Im Falle des Vorgehens mithilfe einer Schrittseparations-Repräsentation ist die Menge der zu verhindernden Schrittfolgen direkt durch S gegeben, d.h. die Menge S wird anstelle von $\mathcal{L}'_{ws}$ betrachtet.

Anstelle einer Menge von unerwünschten Schrittfolgen kann natürlich auch eine endliche Menge $\tilde{\mathcal{L}}$ von unerwünschten Abläufen in der Form von BPOs spezifiziert werden. In diesem Fall entspricht also das Komplement der Menge $\{bpo \mid \exists bpo' \in \tilde{\mathcal{L}} : bpo' \in PS(bpo)\}$ der oberen Schranke $\mathcal{L}'$ aus Problemspezifikation 5.4.1, wobei diese obere Schranke dann unendlich sein kann. Eine Lösung dieses Problems kann wie im Falle von unerwünschten Schrittfolgen gewonnen werden. Bei der Verwendung einer Erzeugendensystem-Repräsentation müssen die BPOs aus $\tilde{\mathcal{L}}$ auf Aktiviertheit geprüft werden. Im Falle einer Schrittseparations-Repräsentation muss der Zusammenhang zwischen aktivierten BPOs und Schrittfolgen aus Definition 3.3.8 geeignet verwendet werden, d.h. für jede BPO aus $\tilde{\mathcal{L}}$ wird eine Schrittlinearisierung gesucht, welche durch eine geeignete zulässige Stelle verhindert werden kann.

Version Übereinstimmung: In einigen Anwendungskontexten wird die Abweichung des Verhaltens eines Systems von erwünschtem Verhalten durch ein Übereinstimmungsmaß bewertet. In unserem Kontext

ist es also beispielsweise möglich, dass eine obere Schranken für das Verhalten eines zu synthetisierenden Petrinetzes mithilfe eines Maßes μ gegeben ist, welches den Grad der Übereinstimmung einer partiellen Sprache und eines S/T-Netzes misst. Wir gehen hierzu davon aus, dass eine untere Verhaltensschranke durch eine partielle Sprache $\mathcal{L}$ gegeben ist und eine obere Verhaltensschranke durch einen Wert μ_0, welcher eine untere Schranke für die Übereinstimmung von $\mathcal{L}$ zu dem zu synthetisierenden Netz bzgl. μ darstellt, spezifiziert ist. Für eine gegeben partielle Sprache $\mathcal{L}$ und eine entsprechende Schranke μ_0 ergibt sich also die Frage nach einem S/T-Netz (N, m_0), welches $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$ und $\mu((N, m_0), \mathcal{L}) \geqslant \mu_0$ erfüllt. Wir betrachten hier nun den Fall, dass das Maß μ für obere Approximationen der betrachteten Sprache in dem Sinne monoton ist, dass aus $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0) \subseteq \mathfrak{Bpo}(N', m_0')$ die Beziehung $\mu((N, m_0), \mathcal{L}) \geqslant \mu((N', m_0'), \mathcal{L})$ folgt. In diesem Fall lässt sich das formulierte Problem dadurch lösen, dass eine entsprechend der Formulierung in Lemma 4.3.3 beste obere Approximation (N, m_0) an $\mathcal{L}$ berechnet wird. Mit einem Test, ob $\mu((N, m_0), \mathcal{L}) \geqslant \mu_0$, lässt sich das Problem dann lösen, da im Falle eines negativen Testergebnisses direkt gefolgert werden kann, dass es kein Lösungsnetz gibt. Eine entsprechende beste obere Approximation ist beispielsweise durch eine Erzeugendensystem-Repräsentation von $\mathcal{L}$ gegeben. In Unterabschnitt 5.4.4 wird auch eine Möglichkeit zur Berechnung eines geeigneten Approximationsnetzes mithilfe einer Schrittseparations-Repräsentation erläutert. Da das Netz in diesem Fall durch schrittweises Hinzufügen zulässiger Stellentripel erzeugt wird, kann es sinnvoll sein, in jedem Schritt zu prüfen, ob das bisher konstruierte Netz (N, m_0) schon die Forderung $\mu((N, m_0), \mathcal{L}) \geqslant \mu_0$ erfüllt.
Ein einfaches monotones Übereinstimmungsmaß ist beispielsweise durch $\mu((N, m_0), \mathcal{L}) = 1 - (|\mathfrak{Bpo}(N, m_0) \setminus PS(\mathcal{L})|)/|\mathfrak{Bpo}(N, m_0)|$ gegeben. Für klassische Sprachen finden sich in [197] Beispiele für fortschrittlichere Übereinstimmungsmaße. So ist das dort eingeführte Maß der Richtigkeit des Verhaltens a_B in unserem Kontext sehr interessant. Insbesondere ist es in unserem Sinne monoton.

5.4.2 *Variante Stellen-Schranke*

Im Rahmen der formalen Modellierung mit Petrinetzen sind wichtige Modellierungsziele häufig eine automatische Analyse der Modelle, eine Simulation der Modelle, oder sogar eine Implementierung der Modelle in der Form von Programm-Code oder Hardware-Bausteinen. Aus Gründen der Performance darf in diesen Fällen die Größe eines Modells typischerweise eine gewisse obere Grenze nicht überschreiten. Neben den genannten Modellierungszielen sollen Modelle oft auch händisch untersucht, analysiert und beurteilt werden. In solchen Fällen ist es umso wichtiger, dass Modelle nicht zu groß bzw. zu komplex sind, da Anwender und Analysten normalerweise an interpretierbaren und kontrollierbaren Referenzmodellen, welche auch von modellierungsunerfahrenen Managern schnell verstanden werden können, interessiert sind. Für automatisch durch Syntheseverfahren erzeugte Modelle ist das Kriterium der Verständlichkeit und Lesbarkeit besonders entscheidend, um händische Feinabstimmungen und eine langfristige Wartung des Modells zu ermöglichen.
Folglich ist eine wichtige Anforderung an ein zu synthetisierendes Petrinetz die Einhaltung einer oberen Schranke für die Größe des Netzes.

Die Größe eines Netzes kann durch die Anzahl seiner Komponenten gemessen werden. Während die Anzahl der Transitionen eines zu synthetisierenden Petrinetzes durch die Anzahl der Aktivitäten in der Verhaltens-Spezifikation, i.e. der betrachteten partiellen Sprache, vorgegeben ist, stellt eine Beschränkung der Anzahl der Stellen des Netzes eine natürliche Anforderung dar. Wir betrachten also die Situation, dass neben der Verhaltens-Spezifikation in Form einer partiellen Sprache zusätzlich eine obere Schranke für die Anzahl der Stellen spezifiziert wird. Hierbei stellt sich u.U. die Frage welche Größenordnung für eine solche Schranke sinnvoll ist. Diese lässt sich aber im Allgemeinen nicht befriedigend beantworten, da dies sicherlich von mehreren Faktoren wie dem Modellierungszweck, der Anzahl an Aktivitäten des zu modellierenden Systems oder der Größe der Verhaltens-Spezifikation abhängt.

PROBLEMSPEZIFIKATION 5.4.2 (SYNTHESEPROBLEM MIT STELLEN-SCHRANKE)

Problemspezifikation 5.4.2 (Syntheseproblem mit Stellen-Schranke)

Eingabe: *Eine endliche partielle Sprachen $\mathcal{L}$ und eine Schranke $b \in \mathbb{N}^+$.*
Ausgabe: *Ein markiertes S/T-Netz (N, m_0), $N = (P, T, W)$, mit $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$ und $|P| \leqslant b$, falls ein solches Lösungsnetz existiert, und eine „*`notexists`*“-Meldung sonst.*

Mit dem Konzept separierender Stellen kann das Problem dadurch gelöst werden, dass alle b-elementigen Partitionen $\{\mathcal{L}_{ws}^1 \dots \mathcal{L}_{ws}^b\}$ der Menge der falschen Schrittfortsetzungen $\mathcal{L}_{ws}$ betrachtet werden. Falls es für eine dieser Partitionen möglich ist, für jede der b Mengen der Partition ein zulässiges Stellentripel zu finden, welches alle falschen Schrittfortsetzungen der Menge separiert, so ist ein Netz mit entsprechenden b Stellen eine Schrittseparations-Repräsentation bzgl. $\mathcal{L}$. Mit einem der bekannten Übereinstimmungstests lässt sich prüfen, ob ein solches Netz ein Lösungsnetz ist. Ist dies nicht der Fall, so lässt sich folgern, dass das Problem eine negative Antwort besitzt. Ist oben Genanntes für keine Partition möglich, so lässt sich direkt eine negative Antwort auf das Syntheseproblem geben. Es ist noch anzumerken, dass die Frage nach einem zulässigen Stellentripel, welches eine Menge von falschen Schrittfortsetzungen separiert, dadurch beantwortet werden kann, dass ein Ungleichungssystem gelöst wird, welches eine $\mathbf{b}_\sigma^{SSS}$- bzw. $\mathbf{b}_\sigma^{BPOSS}$- bzw. $\mathbf{b}_\sigma^{MSS}$-Ungleichung für jede falsche Schrittfortsetzung σ der betrachteten Menge enthält. Hierzu können wie bisher üblich Techniken der linearen Optimierung verwendet werden. Ein Problem des beschriebenen Ansatzes ist, dass die Anzahl der zu betrachtenden Partitionen exponentiell ist.
Daher schlagen wir hier noch eine fortschrittlichere Vorgehensweise vor. Diese baut auf Ideen aus [48] auf. In dieser Arbeit geht es, wie dort erklärt, nicht um Regionen-basierte Synthese, sondern um einen verwandten Ansatz zur Konstruktion von Petrinetzen. Dennoch lassen sich die Grundprinzipien dieser Arbeit auch in unserem Zusammenhang verwenden. Das Syntheseproblem mit Stellen-Schranke kann mithilfe von Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \dots, t_m\}$ gelöst werden, indem das folgende ganzzahlige Zulässigkeitsproblem gelöst wird.

- $\mathbf{A}_\mathcal{L}^{ST} \cdot \mathbf{x}^i \geqslant \mathbf{o}$, $i \in \{1, \dots, b\}$;
- $-k \cdot s_{\sigma,i} + \mathbf{b}_\sigma^{SSS} \cdot \mathbf{x}^i < \mathbf{o}$, $i \in \{1, \dots, b\}$, $\sigma \in \mathcal{L}_{ws}$;
- $\sum_{i=1}^b s_{\sigma,i} \leqslant b - 1$, $\sigma \in \mathcal{L}_{ws}$;

- $\mathbf{x}^i \in \mathbb{N}^{2m+1}$, $i \in \{1, \ldots, b\}$;
- $k \in \mathbb{N}$;
- $s_{\sigma,i} \in \{0, 1\}$, $i \in \{1, \ldots, b\}$, $\sigma \in \mathcal{L}_{ws}$.

Die Vektoren $\mathbf{x}^i$ repräsentieren b Schritt-Transitions-Regionen über die Ungleichungen $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x}^i \geqslant \mathbf{0}$. Falls $s_{\sigma,i} = 0$, dann ist die Beschränkung $-k \cdot s_{\sigma,i} + \mathbf{b}_{\sigma}^{SSS} \cdot \mathbf{x}^i < \mathbf{0}$ aktiv, so dass sich die übliche Ungleichung zum Separieren der falschen Schrittfortsetzung σ durch die Region $\mathbf{x}^i$ ergibt. Falls $s_{\sigma,i} = 1$, dann ist die Beschränkung erfüllt, wenn k groß genug gewählt wird. Somit ist die Beschränkung in diesem Fall redundant. Weiterhin folgt aus der Ungleichung $\sum_{i=1}^{b} s_{\sigma,i} \leqslant b - 1$, dass zumindest ein $s_{\sigma,i}$ Null ist. Damit ist für jede falsche Schrittfortsetzung σ eine der betrachteten Beschränkungen aktiv, so dass σ durch eine der Regionen $\mathbf{x}^i$ separiert wird. Falls das Ungleichungssystem eine Lösung besitzt, so definiert eine solche folglich eine Schrittseparations-Repräsentation von $\mathcal{L}$ mit b Stellen. Das betrachtete Syntheseproblem lässt sich dann mit einem Übereinstimmungstest lösen. Falls das Ungleichungssystem keine Lösung besitzt, so hat das Problem eine negative Antwort. Gibt es nämlich ein Lösungsnetz für das Problem, so ergibt sich folgendermaßen eine Lösung des Ungleichungssystems. Die Vektoren $\mathbf{x}^i$ werden korrespondierend zu den Stellen des Netzes gewählt. Weiter wird $s_{\sigma,i} = 0$ gesetzt, falls das zu $\mathbf{x}^i$ korrespondierende Stellentripel die falsche Schrittfortsetzung σ separiert. Andernfalls wird $s_{\sigma,i} = 1$ festgelegt. Indem k groß genug gewählt wird, lassen sich alle Beschränkungen $-k \cdot s_{\sigma,i} + \mathbf{b}_{\sigma}^{SSS} \cdot \mathbf{x}^i < \mathbf{0}$ erfüllen. Da jede falsche Schrittfortsetzung durch eine Stelle des Netzes separiert wird, sind auch die Ungleichungen $\sum_{i=1}^{b} s_{\sigma,i} \leqslant b - 1$ erfüllt.
Generell kann das betrachtete ganzzahlige Zulässigkeitsproblem, wie in Abschnitt 3.4 beschrieben, durch Methoden der ganzzahligen linearen Optimierung gelöst werden. Diese weisen im Allgemeinen eine exponentielle Komplexität auf. Hinzu kommt, dass das zu lösende Ungleichungssystem sehr groß ist. Allerdings gibt es entsprechende praxisorientierte Optimierungs-Werkzeuge, welche auch mit sehr großen Probleminstanzen zurecht kommen.
Das Prinzip der Berechnung einer Erzeugendensystem-Repräsentation ist im Rahmen der Problemstellung dieses Unterabschnittes nur zu Vorverarbeitungszwecken, wie im letzten Punkt von Bemerkung 4.4.4 erklärt, nützlich.
Version Minimal: Ein Algorithmus zur Lösung des Syntheseproblems mit Stellen-Schranke kann genutzt werden, um ein Lösungsnetz für das klassische Syntheseproblem zu berechnen, welches eine minimale Anzahl an Stellen besitzt. Hierzu kann so vorgegangen werden, dass zuerst geprüft wird, ob das klassische Problem positiv lösbar ist. Im negativen Fall ergibt sich auch für die hier betrachtete Problemversion eine negative Antwort. Im positiven Fall wird dann das Syntheseproblem mit der Stellen-Schranke $b = 1$, dann mit $b = 2$, usw., betrachtet bis für ein b ein Lösungsnetz gefunden wird. Dieses ergibt dann eine Lösung der Problemversion. Da die Existenz eines endlichen Lösungsnetzes sichergestellt ist, terminiert das Verfahren.
Wie oben lassen sich auch wieder die Ideen aus [48] verwenden, um diese Problemversion mithilfe eines ganzzahligen linearen Optimierungsproblems zu lösen. Hierzu wird zuerst wieder das klassische Syntheseproblem gelöst. Hat dieses eine positive Lösung, so wird die

Anzahl der Stellen b des berechneten Lösungsnetzes betrachtet. Die Problemversion lässt sich mithilfe von Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ lösen, indem das folgende ganzzahlige lineare Optimierungsproblem gelöst wird.

- $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x}^i \geqslant \mathbf{o}$, $i \in \{1, \ldots, b\}$;
- $-k \cdot s_{\sigma,i} + \mathbf{b}_{\sigma}^{SSS} \cdot \mathbf{x}^i < \mathbf{o}$, $i \in \{1, \ldots, b\}$, $\sigma \in \mathcal{L}_{ws}$;
- $\sum_{i=1}^{b} s_{\sigma,i} \leqslant b - 1$, $\sigma \in \mathcal{L}_{ws}$;
- $k \cdot z_i - \sum_{j=0}^{2n} x_j^i \geqslant 0$, $i \in \{1, \ldots, b\}$;
- $\mathbf{x}^i \in \mathbb{N}^{2m+1}$, $i \in \{1, \ldots, b\}$;
- $k \in \mathbb{N}$;
- $z_i, s_{\sigma,i} \in \{0, 1\}$, $i \in \{1, \ldots, b\}$, $\sigma \in \mathcal{L}_{ws}$;
- $\min! \sum_{i=1}^{b} z_i$.

Das Ungleichungssystem entspricht dem oben diskutierten System zur Berechnung einer Schrittseparations-Repräsentation mit b Stellen (eine solche existiert nach Voraussetzung) ergänzt um binäre Variablen z_i und Ungleichungen $k \cdot z_i - \sum_{j=0}^{2n} x_j^i \geqslant 0$. Falls $z_i = 1$, so lässt sich letztere Beschränkung erfüllen, indem k groß genug gewählt wird. Falls $z_i = 0$, so ist die Beschränkung nur dann erfüllt, wenn $\mathbf{x}^i = 0$, i.e. $\mathbf{x}^i$ definiert in diesem Fall das redundante Null-Stellentripel. Nun wird in dem betrachteten Optimierungsproblem die Summe $\sum_{i=1}^{b} z_i$ minimiert. Somit erhalten so viele z_i wie möglich den Wert Null. Folglich entsprechen so viele $\mathbf{x}^i$ wie möglich dem Nullvektor. Zu solchen Vektoren korrespondierende Stellen werden natürlich von der zu konstruierenden Schrittseparations-Repräsentation weggelassen. Somit wird die Anzahl der Stellen des Netzes durch dieses Optimierungsproblem minimiert. Es wird also eine Schrittseparations-Repräsentation mit einer minimalen Anzahl an Stellen berechnet. Dies löst offenbar die betrachtete Problemversion.

Version Subnetz: Die Forderung nach einem Netz mit einer minimalen Anzahl an Stellen kann zu der Forderung, dass es kein Subnetz gibt, welches auch das Syntheseproblem löst, abgeschwächt werden. Es wird also nach einem Lösungsnetz für das klassische Syntheseproblem gesucht, so dass kein Subnetz des Netzes auch das Syntheseproblem löst. Diese Fragestellung kann gelöst werden, indem zuerst das klassische Syntheseproblem gelöst wird. Falls es ein Lösungsnetz für dieses Problem gibt, so werden alle Stellen des synthetisierten Netzes in beliebiger Reihenfolge untersucht. Es wird für jede Stelle geprüft, ob sie weggelassen werden kann, ohne das Verhalten des Netzes zu ändern. Hierzu kann entweder geprüft werden, ob auch ohne die Stelle alle falschen Schrittfortsetzungen ausgeschlossen sind, oder es kann geprüft werden, ob die Stelle eine implizite Stelle ist. Im positiven Fall wird die Stelle dann aus dem Netz entfernt. Anschließend werden die Stellen des verbleibenden Netzes untersucht. Es ist zu beachten, dass die Anzahl der Stellen des resultierenden Netzes von der Reihenfolge, in der die Stellen untersucht werden, abhängt [65]. In jedem Fall entsteht aber ein Netz, welches die formulierte Problemversion löst.

5.4.3 *Variante Zustände*

Spezifikationen von Systemen enthalten in einigen Fällen explizite Informationen über Zustände des Systems, beispielsweise über Fehlerzustände oder normal terminierende Zustände. Diese können in etlichen Modellierungssprachen auch berücksichtigt werden (vgl. z.B. UML). Eine partielle Sprache erlaubt erst einmal nur die Spezifikation der Szenarien eines Systems. Szenarien beschreiben das beobachtbare Verhalten eines Systems, d.h. die Durchführung der Aktivitäten des Systems, auf der einfachen und intuitiven Ebene von einzelnen Systemabläufen. Bei diesem Vorgehen werden eigentlich weder Verzweigungsverhalten noch Systemzustände berücksichtigt. Daher stellen Szenarien eine sehr klare benutzerorientierte Sicht auf ein System dar und können einfach spezifiziert werden. Eine typische Situation ist nun, dass ein Benutzer neben dem Wissen über Szenarien des Systems auch teilweise Wissen über Zustände des Systems hat. Normalerweise sind sicherlich nicht alle Systemzustände umfassend bekannt, aber es gibt u.U. partielle Informationen über Zustände. Solche sollen nun berücksichtigt werden. Sei also eine Szenario-Spezifikation in der Form einer partiellen Sprache $\mathcal{L}$ gegeben. Dann erlauben wir hierzu folgende ergänzende Spezifikationstechnik. Es kann für beliebige Teilabläufe angegeben werden, dass sie zum selben Zustand führen sollen. Ein Teilablauf ist dabei natürlicherweise durch ein Präfix eines Ablaufes aus $\mathcal{L}$ gegeben. Zusätzlich zu einer partiellen Sprache $\mathcal{L}$ sind also paarweise disjunkte Teilmengen $\mathcal{L}_1, \ldots, \mathcal{L}_l \subseteq \mathrm{Pref}(\mathcal{L})$ gegeben. Die Teilabläufe einer solcher Teilmenge sollen dann alle in dem zu synthetisierenden Netz dieselbe Folgemarkierung erzeugen.

Problemspezifikation 5.4.3 (Syntheseproblem mit Zuständen)

PROBLEMSPEZIFIKATION 5.4.3 (SYNTHESEPROBLEM MIT ZUSTÄNDEN)
Eingabe: *Eine endliche partielle Sprachen $\mathcal{L}$ und eine endliche Folge paarweise disjunkter partieller Sprachen $\mathcal{L}_1, \ldots, \mathcal{L}_l \subseteq \mathrm{Pref}(\mathcal{L})$.*
Ausgabe: *Ein markiertes S/T-Netz (N, m_0) mit $\mathfrak{Bpo}(N, m_0) = \mathrm{PS}(\mathcal{L})$, so dass für $j \in \{1, \ldots, l\}$ und $\mathrm{bpo}, \mathrm{bpo}' \in \mathcal{L}_j$ die Folgemarkierungen der bzgl. (N, m_0) aktivierten BPOs* bpo *und* bpo' *übereinstimmen, falls ein solches Lösungsnetz existiert, und eine „*notexists*"-Meldung sonst.*

Anzumerken ist hier, dass, falls zwei BPOs bpo und bpo' in einem Netz dieselbe Folgemarkierung erzeugen, dann das mögliche Folgeverhalten des Netzes übereinstimmt. Somit ergibt sich als notwendige Bedingung für die Lösbarkeit des betrachteten Problems, dass für entsprechende BPOs bpo und bpo' das in $\mathcal{L}$ spezifizierte Folgeverhalten übereinstimmen muss.
Im Allgemeinen gilt, dass die zusätzlichen Anforderungen bzgl. Systemzuständen zusätzliche Gleichungen für die linear algebraischen Charakterisierungen von Regionen definieren. Im Falle von Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ ergeben sich folgendermaßen zusätzliche Einschränkung. Wir fixieren $\mathrm{bpo}_j = (V_j, <_j, l_j) \in \mathcal{L}_j$ für jedes $j \in \{1, \ldots, l\}$. Dann betrachten wir für jede BPO $\mathrm{bpo} = (V, <, l) \in \mathcal{L}_j$, $\mathrm{bpo} \neq \mathrm{bpo}_j$, einen Zeilenvektor $\mathbf{s}^j_{\mathrm{bpo}} = (s^j_{\mathrm{bpo},0}, \ldots, s^j_{\mathrm{bpo},2m})$, so dass genau dann $\mathbf{s}^j_{\mathrm{bpo}} \cdot \mathbf{x} = \mathbf{o}$ gilt, wenn das Schalten von bpo und bpo_j zur selben Anzahl an Marken in $p_{\overrightarrow{x}}$ führt. Zur Matrix $\mathbf{A}^{ST}_{\mathcal{L}}$ fügen

wir dementsprechend die Zeilen $\mathbf{s}^j_{bpo}$ und $-\mathbf{s}^j_{bpo}$ hinzu. Wir definieren hierzu

$$\mathbf{s}^j_{bpo,i} = \begin{cases} 0 & \text{für } i = 0 \\ |V|_l(t_i) - |V_j|_{l_j}(t_i) & \text{für } i = 1,\ldots,m \\ -|V|_l(t_{i-m}) + |V_j|_{l_j}(t_{i-m}) & \text{für } i = m+1,\ldots,2m \end{cases}$$

Das Syntheseproblem mit Zuständen lässt sich nun dadurch lösen, dass diese erweiterte Matrix anstelle der ursprünglichen Matrix $\mathbf{A}^{ST}_{\mathcal{L}}$ im Rahmen der Verfahren zur Berechnung einer Schrittseparations-Repräsentation und einer Erzeugendensystem-Repräsentation verwendet wird. Ansonsten kann analog vorgegangen werden wie im Falle des klassischen Syntheseproblems.

Version Zustandsseparation: Das Syntheseproblem mit Zuständen kann derart verallgemeinert werden, dass nicht nur übereinstimmende Zustände $\mathcal{L}_1,\ldots,\mathcal{L}_l$ sondern zusätzlich auch sich unterscheidende Zustände spezifiziert werden. Eine solche Spezifikation legt also für gewisse Paare $(\mathcal{L}_i,\mathcal{L}_j)$ fest, dass die zwei durch $\mathcal{L}_i$ und $\mathcal{L}_j$ definierten Markierungen verschieden sein müssen. Hierzu müssen diese beiden Markierung in mindestens einer Stelle verschieden sein. Im Falle der Berechnung einer Schrittseparations-Repräsentation kann daher für jedes solche Paar versucht werden, eine derart zustandsseparierende zulässige Stelle zu berechnen. Dies ist auf ähnliche Weise möglich wie die Berechnung schrittseparierender Stellentripel. Für jedes spezifizierte Paar $(\mathcal{L}_i,\mathcal{L}_j)$ wird ein Ungleichungssystem betrachtet, welches die obig betrachtete erweiterte linear algebraische Charakterisierung von Regionen um eine Ungleichung $\mathbf{s}^j_{bpo_i} \cdot \mathbf{x} \neq \mathbf{0}$ ($bpo_j \in \mathcal{L}_j$, $bpo_i \in \mathcal{L}_i$) erweitert. Hat dieses System eine Lösung, so separiert das zugehörige zulässige Stellentripel die Markierungen von $\mathcal{L}_i$ und $\mathcal{L}_j$. Hat es keine Lösung, so lassen sich die beiden Markierungen unter den gegebenen Voraussetzungen nicht separieren und es kann eine negative Antwort auf das betrachtete Problem gegeben werden. Ansonsten kann wiederum völlig analog wie im Falle des klassischen Syntheseproblems vorgegangen werden.

Im Rahmen der Berechnung einer Erzeugendensystem-Repräsentation lässt sich zeigen, dass ein mit der betrachteten erweiterten linear algebraischen Charakterisierung von Regionen berechnetes Netz (N, m_0) entweder das Problem positiv löst oder das Problem eine negative Lösung besitzt. Um zu überprüfen, ob (N, m_0) das Problem löst, muss wie üblich ein Übereinstimmungstest durchgeführt werden. Im negativen Fall kann analog zum klassischen Syntheseproblem eine negative Antwort auf das Problem gegeben werden. Im positiven Fall muss zusätzlich noch für jedes spezifizierte Paar $(\mathcal{L}_i,\mathcal{L}_j)$ geprüft werden, ob die BPOs bpo_j und bpo_i ($bpo_j \in \mathcal{L}_j$, $bpo_i \in \mathcal{L}_i$) in (N, m_0) zu unterschiedlichen Folgemarkierungen führen. Falls dies der Fall ist, so löst (N, m_0) das Problem. Ist dies nicht der Fall, so separiert keine zulässige Stelle, welche konform zu den Zustandsforderungen aus Problemspezifikation 5.4.3 ist, die Markierungen von $\mathcal{L}_i$ und $\mathcal{L}_j$, da jede solche Stelle eine nicht-negative Linearkombination der Stellen von (N, m_0) ist. Somit kann dann eine negative Antwort auf das Problem gegeben werden.

Zu dieser Version sei noch angemerkt, dass eine vollständige Spezifikation, welche Elemente aus $Pref(\mathcal{L})$ zu demselben Zustand führen sollen und welche zu verschiedenen Zuständen führen sollen, zu einem zu dem klassischen Syntheseproblem für Schritt-Transitionssysteme sehr ähnlichem Problem führt [17, 18] (vgl. auch Unterabschnitt 4.3.4).

5.4.4 *Variante beste obere Approximation*

Bei den bisher betrachteten Syntheseproblemen ist die Entscheidung, ob es ein Netz mit gewissen Eigenschaft gibt, ein zentraler Aspekt. Falls die Probleme eine positive Antwort besitzen, so wird auch die Berechnung eines entsprechendes Netzes gefordert. Falls sie allerdings eine negative Antwort haben, so gibt es über die Lösung des Entscheidungsproblems hinaus keine weiteren Forderungen. Für viele praktische Anwendungsszenarien von Synthese spielt ein solches Entscheidungsproblem aber eine untergeordnete Rolle. Oft ist es wichtig, für beliebige Spezifikationen ein sinnvolles Systemmodell zu synthetisieren, unabhängig davon, ob sich ein gewisses Syntheseproblem exakt lösen lässt. Die Frage ist dann allerdings, wie der Begriff „sinnvoll“ definiert werden kann, d.h. unter welchen Bedingungen stellt ein Netz, welches einer Spezifikation nicht genügt dennoch ein sinnvolles Systemmodell dar. Dies hängt sicherlich vom Anwendungskontext ab. Eine geeignete Möglichkeit besteht darin, ein Netz zu berechnen, welches eine in einem gewissen Sinne beste Approximation an die Spezifikation darstellt. Im Rahmen des klassischen Syntheseproblems ergibt sich die Suche nach einem S/T-Netz, dessen Ablaufsemantik eine beste Approximation an die gegeben Sprache unter allen Ablaufsemantiken von S/T-Netzen darstellt, als natürliche Synthesefragestellung. Falls das klassische Syntheseproblem eine positive Antwort besitzt, so wird auch bei dieser Fragestellung die Konstruktion eines exakten Lösungsnetzes gefordert. Falls es keine exakte Lösung gibt, so wird gewissermaßen die Berechnung eines besten Kandidaten für ein Lösungsnetz verlangt.
Generell erscheint ein solches Vorgehen sinnvoll, da sich praktische System-Spezifikationen häufig nicht exakt realisieren lassen und in vielen praktischen Anwendungsfeldern ohnehin Näherungslösungen ausreichend sind. Daneben ist die Synthese von besten Approximationen auch von einer rein theoretischen Sichtweise her interessant, da wichtige Entscheidungsprobleme für Petrinetze entscheidbar sind und effizient gelöst werden können.
Nun lässt sich aber immer noch unterscheiden, ob die Sprache des zu synthetisierenden Netzes eine beste obere oder eine beste untere Approximation an die Spezifikation darstellen soll. Eine beste obere Approximation ist häufig sinnvoll, da in vielen Fällen das explizit spezifizierte Verhalten auf jeden Fall in dem Ziel-System berücksichtigt werden soll. Dies gilt insbesondere dann, wenn die Sprache von einem Experten spezifiziert ist. In diesem Fall handelt es sich normalerweise ausschließlich um erwünschtes Systemverhalten. Die spezifizierte Sprache soll also in dem Ablaufverhalten des zu synthetisierenden Netzes beinhaltet sein. Die Konstruktion eines besten oberen Approximations-Netzes stellt dies sicher und garantiert darüber hinaus, dass das Netz nur solches zusätzliches Verhalten aufweist, welches nötig ist, um ein Petrinetzmodell zu erhalten. Somit kann eine solche Approximation als eine natürliche Vervollständigung der spezifizierten Sprache durch ein Petrinetz betrachtet werden. Eine solche Ergänzung der spezifizierten Sprache ist vielfach notwendig, da selbst in qualitativ hochwertigen Spezifikationen einige Verhaltensweisen bzw. Szenarien eines Systems auch von Experten übersehen oder vergessen werden, d.h. Spezifikationen sind oft unvollständig (ein gutes Beispiel hierfür ist das in Unterabschnitt 2.2.2 diskutierte Process-Mining).

Formal bedeutet die Berechnung einer besten oberen Approximation, dass ein S/T-Netz konstruiert wird, dessen Ablaufsemantik in jeder Ablaufsemantik eines S/T-Netzes, welche die spezifizierte Sprache enthält, beinhaltet ist und selbst auch die spezifizierte Sprache enthält. Entsprechend der Überlegungen des letzten Kapitels wissen wir schon, dass ein solches Lösungsnetz für jede partielle Sprache existiert (Lemma 4.4.10). Die Ablaufsemantik eines Lösungsnetzes stellt in diesem Fall die eindeutige kleinste Ablaufsemantik eines S/T-Netzes dar, welche die gegebene partielle Sprache enthält. Dies sind sehr schöne Eigenschaften für obere Petrinetz-Approximationen. Für untere Approximationen gelten entsprechende Zusammenhänge nicht.

PROBLEMSPEZIFIKATION 5.4.4 (SYNTHESEPROBLEM BESTE OBERE APPROXIMATION)
Eingabe: *Eine endliche partielle Sprachen* $\mathcal{L}$.
Ausgabe: *Ein markiertes S/T-Netz* (N, m_0), *so dass* $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$ *und* $(\forall (N' m_0') : (\mathfrak{Bpo}(N, m_0) \setminus \mathfrak{Bpo}(N', m_0') \neq \emptyset) \implies (PS(\mathcal{L}) \not\subseteq \mathfrak{Bpo}(N', m_0')))$.

Problemspezifikation 5.4.4 (Syntheseproblem beste obere Approximation)

Wir haben in Abschnitt 4.4 gezeigt, dass das gesättigt zulässige Netz und damit auch eine Erzeugendensystem-Repräsentation von $\mathcal{L}$ die formulierten Anforderungen an eine beste obere Approximation erfüllen. Das hier betrachtete Problem ist also für jede endliche partielle Sprache lösbar. Die Synthese einer besten oberen Netz-Approximation wurde für klassische Sprachen schon in [65] im Rahmen der Berechnung einer Art von Erzeugendensystem-Repräsentation angesprochen.
Wie gesagt, kann das hier betrachtete Syntheseproblem nun dadurch gelöst werden, dass eine Erzeugendensystem-Repräsentation (N, m_0), wie in Abschnitt 4.4 beschrieben, berechnet wird. Daneben lässt sich auch das Verfahren zur Berechnung einer Schrittseparations-Repräsentation (N, m_0) nutzen. Eine solche löst das Problem im Allgemeinen zwar nicht, aber sie stellt eine obere Approximation an $\mathcal{L}$ dar, i.e. $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N, m_0)$. Der Grund dafür, dass es sich bei (N, m_0) nicht um eine beste obere Approximation handeln muss, ist entsprechend Lemma 4.4.2 folgendermaßen. Es ist möglich, dass es kein bzgl. einer falschen Schrittfortsetzung $\sigma = \tau_1 \dots \tau_j$ schrittseparierendes zulässiges Stellentripel gibt, aber ein bzgl. $\tau_1 \dots \tau_{j-1}(\tau_j + t)$ oder $\tau_1 \dots \tau_j t$ schrittseparierendes Stellentripel existiert. Ein solches wird aber nicht notwendigerweise zu (N, m_0) hinzugefügt.
Daher lässt sich das betrachtete Problem mit folgender Ergänzung des Verfahrens zur Berechnung einer Schrittseparations-Repräsentation lösen. Falls es bei diesem Verfahren für eine falsche Schrittfortsetzung $\sigma = \tau_1 \dots \tau_j$ kein schrittseparierendes zulässiges Stellentripel gibt, so behandeln wir σ wie eine von der Sprache erzeugte Schrittfolge. Das bedeutet, dass wir in diesem Fall für jedes $t \in T$ die Schrittfolgen $\tau_1 \dots \tau_{j-1}(\tau_j + t)$ und $\tau_1 \dots \tau_j t$ zur Menge der falschen Schrittfortsetzungen $\mathcal{L}_{ws}$ hinzufügen. Auf diese Weise wird dann versucht, für diese Schrittfolgen schrittseparierende Stellentripel zu finden. Wenn dies wiederum nicht möglich ist, so setzt sich das Vorgehen entsprechend fort und es werden jeweils entsprechend längere Schrittfolgen als falsche Schrittfortsetzungen behandelt.
Bei diesem Algorithmus kann die Menge $\mathcal{L}_{ws}$ wachsen. Folgendermaßen lässt sich zeigen, dass der Algorithmus dennoch terminiert. Eine Schrittfolge $\sigma = \tau_1 \dots \tau_j$, in der eine Transition t öfter vorkommt als die Maximalanzahl an Vorkommen von t in einer BPO aus $\mathcal{L}$, i.e.

$(\tau_1 + \ldots + \tau_j)(t) > n^t$, kann immer durch das zulässige Stellentripel st^t separiert werden (vgl. Lemma 4.5.1 und Lemma 4.5.2). Somit ist die Größe einer Schrittfolge $\sigma = \tau_1 \ldots \tau_j$, welche zu $\mathcal{L}_{ws}$ hinzugefügt wird, dadurch begrenzt, dass die Multimenge $\tau_1 + \ldots + \tau_j$ höchstens $\sum_{t \in T} n^t + 1$ Elemente besitzt. Ausgehend von einer falschen Schrittfortsetzung werden also nur endlich viele weitere Schrittfolgen im Rahmen des skizzierten Vorgehens betrachtet.
Für ein mit dem beschriebenen Algorithmus konstruiertes Netz (N, m_0) lässt sich nachprüfen, dass jede Schrittfolge $\sigma \in \mathfrak{Step}(N, m_0) \setminus \{\sigma(bpo) \mid bpo \in Slin(Pref(\mathcal{L}))\}$ auch in jedem Netz (N', m_0') mit $PS(\mathcal{L}) \subseteq \mathfrak{Bpo}(N', m_0')$ aktiviert ist (ähnlich wie im Beweis zu Satz 4.4.1). Hieraus ergibt sich für (N, m_0) die gewünschte Eigenschaft einer besten oberen Approximation aus Problemspezifikation 5.4.4.

5.4.5 *Variante beste untere Aproximation*

Anstelle einer oberen Approximation können wir, wie im letzten Unterabschnitt ausgeführt, auch untere Approximationen an eine gegebene Sprache betrachten. Ein unteres Approximations-Netz besitzt ausschließlich Abläufe, welche in der Sprache spezifiziert sind. Eine beste untere Approximation ist ein unteres Approximations-Netz mit einer maximalen Anzahl an Abläufen. Eine Schwierigkeit hierbei ist, dass es keine eindeutige größte Ablaufsemantik eines S/T-Netzes gibt, welche in der gegebenen partiellen Sprache beinhaltet ist.
Es gibt realistische Szenarien, in denen die Berechnung einer besten unteren Approximation im Vergleich zu einer besten oberen Approximation wünschenswert ist. Dies ist dann der Fall, wenn es problematischer ist, gewisses nicht-spezifiziertes Verhalten in einem Systemmodell zuzulassen als spezifiziertes Verhalten wegzulassen. Eine typische Situation hierfür ist eine in dem Sinne vollständige Spezifikation, dass alle nicht-spezifizierten Abläufe potentiell fehlerhaftes Verhalten darstellen, welches auf jeden Fall verhindert werden soll. Solche Spezifikationen kommen insbesondere im Rahmen von sicherheitskritischen Systemen wie medizinischen Systemen oder (eingebetteten) Verkehrssystemen vor, denn bei derartigen Systemen können Abläufe, welche nicht explizit entsprechend der Spezifikation erlaubt sind, fatale Folgen haben. Weiter tendieren auch Spezifikationen von Systemen, welche nur wenige möglicherweise lange Abläufe besitzen, dazu, vollständig zu sein. Bei solchen Systemen ist es einfach, das gesamte erwünschte Verhalten in einer Spezifikation zu berücksichtigen, so dass alle nicht-spezifizierten Abläufe möglicherweise Fehler verursachen. Wenn es in solchen Fällen also nicht möglich ist, eine Spezifikation durch ein Petrinetz zu reproduzieren, so ist es sinnvoller, einige der spezifizierten erwünschten Abläufe wegzulassen und somit die Funktionalität des modellierten Systems einzuschränken als nicht-spezifizierte, potentiell fehlerhafte Abläufe in dem modellierten System zu erlauben. Natürlich sollen aber möglichst wenige der erwünschten Abläufe weggelassen werden. Somit ergibt sich die Problematik der Berechnung einer besten unteren Approximation. Ein weiterer Grund, welcher in manchen Fällen für die Berechnung einer unteren Approximation spricht, ist, dass Spezifikationen einerseits aufgrund von menschlichem Versagen insbesondere bei Zeitdruck und andererseits aufgrund von technischen Störeinflüssen und Verzerrungen im Rahmen automatischer Verfahren ohnehin auch fehlerhafte Abläufe enthalten können.

Für eine formale Problemspezifikation muss nun berücksichtigt werden, dass für beste untere Approximationen im Gegensatz zu besten oberen Approximationen bestimmte Eigenschaften nicht gelten. Insbesondere gibt es im Allgemeinen keine Ablaufsemantik eines S/T-Netzes, welche in einer gegebenen partiellen Sprache beinhaltet ist und alle anderen Ablaufsemantiken von S/T-Netzen, welche auch in der gegebenen partiellen Sprache beinhaltet sind, enthält. Daher ist es sinnvoll, die Problemspezifikation auf eine weniger restriktive Weise zu formulieren als im Falle des Syntheseproblems beste obere Approximation. Wir legen hier die Maximalität eines unteren Approximations-Netzes durch die Anzahl der Abläufe fest.

PROBLEMSPEZIFIKATION 5.4.5 (SYNTHESEPROBLEM BESTE UNTERE APPROXIMATION)

Problemspezifikation 5.4.5 (Syntheseproblem beste untere Approximation)

Eingabe: *Eine endliche partielle Sprachen* $\mathcal{L}$.
Ausgabe: *Ein markiertes S/T-Netz* (N, m_0), *so dass* $\mathfrak{Bpo}(N, m_0) \subseteq PS(\mathcal{L})$ *und* $(\forall (N' m_0') : (|\mathfrak{Bpo}(N', m_0')| > |\mathfrak{Bpo}(N, m_0)|) \implies (PS(\mathcal{L}) \not\subseteq \mathfrak{Bpo}(N', m_0')))$.

Dieses Problem kann offenbar gelöst werden, indem das klassische Syntheseproblem für jede präfix- und sequentialisierungsabgeschlossene Teilmenge von $PS(\mathcal{L})$ gelöst wird. Hierzu können die üblichen Verfahren aus Kapitel 4 verwendet werden. Interessant sind dann die bzgl. der Anzahl der Elemente maximalen Teilmengen von $PS(\mathcal{L})$, für die das klassische Syntheseproblem eine positive Antwort besitzt. Ein für eine solche Teilmenge synthetisiertes Netz erfüllt die Anforderungen an eine beste untere Approximation.
Für eine schrittabgeschlossene partielle Sprache $\mathcal{L}$ kann dieses Vorgehen im Rahmen der Verwendung einer Schrittseparations-Repräsentation folgendermaßen deutlich beschleunigt werden. Zuerst wird eine Schrittseparations-Repräsentation von $\mathcal{L}$ berechnet und dabei alle falschen Schrittfortsetzungen, welche nicht ausgeschlossen werden können, gespeichert. Das berechnete Netz bildet den Ausgangspunkt. Im Weiteren sollen noch die gespeicherten falschen Fortsetzungen ausgeschlossen werden. Hierzu betrachten wir die präfix- und sequentialisierungsabgeschlossenen Teilmengen von $PS(\mathcal{L})$ in absteigender Reihenfolge bzgl. der Anzahl der Elemente. Sobald wir eine Menge gefunden haben, für die gilt, dass sich alle gespeicherten Schrittfolgen mit bzgl. dieser Menge zulässigen Stellentripeln verhindern lassen, so führt das Hinzufügen der entsprechenden Stellentripel zum Ausgangsnetz zu einer Lösung des betrachteten Problems.
Version Subnetz: Eine schwächere Forderung ergibt sich, wenn bei obiger Problemspezifikation nicht alle Netze $(N' m_0')$ mit $(|\mathfrak{Bpo}(N', m_0')| > |\mathfrak{Bpo}(N, m_0)|)$ sondern nur die mit $(\mathfrak{Bpo}(N', m_0') \supsetneq \mathfrak{Bpo}(N, m_0))$ berücksichtigt werden. Dies bedeutet, dass nach einem unteren Approximations-Netz mit einer bzgl. Mengeninklusion maximalen Ablaufsemantik gesucht wird. Diese Herangehensweise ist in vielen Fällen sinnvoller als die Anzahl der Elemente zu betrachten. Dies gilt insbesondere für unendliche partielle Sprachen.
Generell ist eine Lösung des obig formulierten Syntheseproblems auch eine Lösung dieser Version, so dass die skizzierten Lösungsverfahren übernommen werden können. Es sind unter der schwächeren Forderung aber auch noch einige Verbesserungen möglich. Eine interessante Verbesserung ergibt sich aus der Überlegung heraus, dass ein unteres Approximations-Netz mit einer maximalen Anzahl an Schrittfolgen

ein Lösungsnetz für die betrachtete Problemversion darstellt. Dies ist darin begründet, dass zwei Netze mit derselben Schrittsemantik auch dieselbe Ablaufsemantik besitzen. Somit besitzt ein Netz mit einer echt größeren Ablaufsemantik auch eine echt größere Schrittsemantik. Mit dieser Vorüberlegung lässt sich die betrachtete Fragestellung im Falle einer schrittabgeschlossenen partiellen Sprache unter Verwendung von Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ als ganzzahliges Optimierungsproblem formulieren. Hierbei schreiben wir für zwei Schrittfolgen $\widetilde{\sigma} < \sigma$, falls $\sigma = \tau_1 \ldots \tau_n$ und $\widetilde{\sigma} = \tau_1 \ldots \tau_j$, $j < n$.

- $k \cdot z_\sigma + \mathbf{a}_\sigma^n \cdot \mathbf{x}^{\sigma'} \geqslant \mathbf{o}$, $\sigma = \tau_1 \ldots \tau_n \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}$, $\sigma' \in \mathcal{L}_{ws}$;
- $-k \cdot (1 - z_\sigma + \sum_{\widetilde{\sigma}<\sigma} z_{\widetilde{\sigma}}) + \mathbf{a}_\sigma^n \cdot \mathbf{x}^{\sigma'} < \mathbf{o}$, $\sigma = \tau_1 \ldots \tau_n \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}$, $\sigma' \in \mathcal{L}_{ws}$, $\sigma < \sigma'$;
- $-k \cdot (\sum_{\widetilde{\sigma}<\sigma'} z_{\widetilde{\sigma}}) + \mathbf{b}_{\sigma'}^{SSS} \cdot \mathbf{x}^{\sigma'} < \mathbf{o}$, $\sigma' \in \mathcal{L}_{ws}$;
- $z_{\widetilde{\sigma}} \leqslant z_\sigma$, $\widetilde{\sigma} < \sigma \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}$;
- $\mathbf{x}^{\sigma'} \in \mathbb{N}^{2m+1}$, $\sigma' \in \mathcal{L}_{ws}$;
- $k \in \mathbb{N}$;
- $z_\sigma \in \{0, 1\}$, $\sigma \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}$;
- $\mathit{min}! \sum_{\sigma \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}} z_\sigma$.

Bei diesem Problem wird für jede falsche Schrittfortsetzung $\sigma' \in \mathcal{L}_{ws}$ ein Vektor $\mathbf{x}^{\sigma'}$, welcher ein Stellentripel definiert, betrachtet. Diese sollen alle Schrittfolgen $\sigma' \in \mathcal{L}_{ws}$ verhindern. Hierzu wird eine falsche Schrittfortsetzung entweder wie üblich separiert oder eine kleinere (Präfix-) Schrittfolge wird separiert. Die Ungleichungen $-k \cdot (\sum_{\widetilde{\sigma}<\sigma'} z_{\widetilde{\sigma}}) + \mathbf{b}_{\sigma'}^{SSS} \cdot \mathbf{x}^{\sigma'} < \mathbf{o}$ erfordern entsprechend, dass entweder σ' durch $\mathbf{x}^{\sigma'}$ separiert wird, oder dass $z_{\widetilde{\sigma}} = 1$ für eine kleinere Schrittfolge $\widetilde{\sigma}$. In letzterem Fall ist die Einschränkung redundant, da k beliebig groß gewählt werden kann. Falls $z_{\widetilde{\sigma}} = 1$, dann gilt entsprechend den Bedingungen $z_{\widetilde{\sigma}} \leqslant z_\sigma$ für alle $\sigma > \widetilde{\sigma}$ auch $z_\sigma = 1$. Falls $z_\sigma = 1$, so stellen die Ungleichungen $-k \cdot (1 - z_\sigma + \sum_{\widetilde{\sigma}<\sigma} z_{\widetilde{\sigma}}) + \mathbf{a}_\sigma^n \cdot \mathbf{x}^{\sigma'} < \mathbf{o}$ sicher, dass σ von $\mathbf{x}^{\sigma'}$ mit $\sigma < \sigma' \in \mathcal{L}_{ws}$ separiert wird, oder dass $z_{\widetilde{\sigma}} = 1$ für ein $\widetilde{\sigma} < \sigma$. In letzterem Fall und im Fall $z_\sigma = 0$ ist die Einschränkung wiederum redundant, da k beliebig groß gewählt werden kann. Es ist hier möglich, σ mithilfe von $\mathbf{x}^{\sigma'}$ zu separieren, da $\mathbf{x}^{\sigma'}$ nicht mehr benötigt wird, um σ' zu separieren. Insgesamt separiert $\mathbf{x}^{\sigma'}$ also die minimale Schrittfolge σ mit $\sigma < \sigma'$ und $z_\sigma = 1$. Da also alle Schrittfolgen σ mit $z_\sigma = 1$ verhindert werden, fordern wir mit den Ungleichungen $k \cdot z_\sigma + \mathbf{a}_\sigma^n \cdot \mathbf{x}^{\sigma'} \geqslant \mathbf{o}$ nur für die Schrittfolgen $\sigma \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}$ mit $z_\sigma = 0$, dass sie in dem zu synthetisierenden Netz aktiviert sind. Insgesamt sind durch die zu den Vektoren $\mathbf{x}^{\sigma'}$, $\sigma' \in \mathcal{L}_{ws}$, gehörigen Stellentripel alle falschen Schrittfortsetzungen verhindert, d.h. das resultierende Netz ist ein unteres Approximations-Netz. Weiter sind in einem entsprechenden Netz alle Schrittfolgen $\sigma \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}$ mit $z_\sigma = 1$ nicht aktiviert, während alle Schrittfolgen mit $z_\sigma = 0$ aktiviert sind. Die Zielfunktion $\mathit{min}! \sum_{\sigma \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}} z_\sigma$ minimiert die Anzahl der nicht aktivierten Schrittfolgen $\sigma \in \{\sigma(\text{bpo}) \mid \text{bpo} \in \text{Slin}(\text{Pref}(\mathcal{L}))\}$, so dass das resultierende Netz die gewünschten Anforderungen erfüllt.

5.4.6 *Variante Optimierung*

Wie in Unterabschnitt 5.4.2 schon diskutiert, stellt die Erzeugung von kompakten und einfach lesbaren Netzen eine zentrale Herausforderung im Rahmen der Synthese von Petrinetzen dar. Auch wenn Lesbarkeit ein eher empirischer Begriff, so ist spielt, wie in Kapitel 2 besprochen, die Anzahl der Komponenten eines Netzes hierbei eine wichtige Rolle. Daneben ist aber auch die Komplexität der einzelnen Komponenten sehr wichtig. Falls die Funktionalität jeder einzelnen Komponente intuitiv verstanden werden kann, so gilt dies häufig auch für das Gesamtnetz oder zumindest für Ausschnitte des Netzes. Da in unseren Syntheseansätzen jeweils Stellen eines zu synthetisierenden Netzes berechnet werden, ist es in diesem Kontext sinnvoll zu fordern, dass Stellen mit möglichst geringer Komplexität synthetisiert werden. Ein naheliegendes Vorgehen stellt hier die Minimierung der Kantengewichte und der Anfangsmarkierungen der Stellen dar. Zum einen sind geringere Anzahlen an Marken und niedrigere Kantengewichten einfacher zu verstehen, da dann weniger komplexe Verhältnisse von vorhandenen Marken zu erforderlichen Marken bei der Betrachtung des Schaltverhaltens von Netzen entstehen. Zum anderen wird durch Kantengewichte von Null die Verzweigungsstruktur des Netzes einfacher, wodurch sich die Abhängigkeitsbeziehungen der Komponenten vereinfachen. Ein vielversprechender Ansatz, um solche Netz zu erzeugen, ist die Betrachtung entsprechender Optimierungsprobleme und zugehöriger Optimierungsverfahren. Mit Optimierungsverfahren kann die Konstruktion einer Stelle eines Netzes derart durch eine Zielfunktion geleitet werden, dass die Kantengewichte und die Anfangsmarkierung der hinzuzufügenden Stelle minimiert werden. Ein entsprechendes Problem formulieren wir in folgender Definition. Allerdings ist auch die Betrachtung anderer Zielfunktionen denkbar.

Problemspezifikation 5.4.6 (Syntheseproblem mit Optimierung)

PROBLEMSPEZIFIKATION 5.4.6 (SYNTHESEPROBLEM MIT OPTIMIERUNG)
Eingabe: *Eine endliche partielle Sprachen* $\mathcal{L}$.
Ausgabe: *Ein markiertes S/T-Netz* (N, m_0), $N = (P, T, W)$, *mit* $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$, *welches für jedes S/T-Netz* (N', m_0'), $N' = (P', T', W')$, *mit* $\mathfrak{Bpo}(N', m_0') = PS(\mathcal{L})$ *die Beziehung* $\max\{m_0(p) + \sum_{t \in T}(W(p,t) + W(t,p)) \mid p \in P\} \leqslant \max\{m_0'(p') + \sum_{t' \in T'}(W'(p',t') + W'(t',p')) \mid p' \in P'\}$ *erfüllt, falls ein solches Lösungsnetz existiert, und eine „*notexists*"-Meldung sonst.*

Eine ähnliche Zielfunktionen im Rahmen der Konstruktion von Petrinetzen wurde bisher in der Arbeit [48] betrachtet. Auch in [69] spielen Zielfunktionen, welche die Konstruktion eines Petrinetzes leiten sollen, eine Rolle.
Das Syntheseproblem mit Optimierung kann mit dem Verfahren zur Berechnung einer Schrittseparations-Repräsentation gelöst werden, indem bei den zu lösenden Zulässigkeitsproblemen eine geeignete Zielfunktion berücksichtigt wird. Im Falle von Schritt-Transitions-Regionen betrachten wir für eine falsche Schrittfortsetzung σ anstelle des üblichen Zulässigkeitsproblems das ganzzahlige lineare Optimierungsproblem $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}, \mathbf{b}_{\sigma}^{SSS} \cdot \mathbf{x} < 0$, $\min! \sum_{i=0}^{2m} x_i$. Es wird also nicht mehr ein beliebiges schrittseparierendes Stellentripel, sondern ein schrittseparierendes Stellentripel st, welches bzgl. der Zielfunktion optimal ist, gesucht. Dadurch wird sichergestellt, dass die zugehörige Stelle p_{st} einen minimalen Wert $m_0(p_{st}) + \sum_{t \in T}(W(p_{st}, t) + W(t, p_{st}))$ hat.

Somit sind die Anforderungen aus Problemspezifikation 5.4.6 für ein auf diese Weise synthetisiertes Netz erfüllt. Tatsächlich erfüllt ein derart synthetisiertes Netz sogar strengere Anforderungen, da lokal für jede falsche Schrittfortsetzung eine entsprechend minimale Stelle berechnet wird.
Die bei diesem Vorgehen auftretenden ganzzahligen linearen Optimierungsprobleme können mit den in Abschnitt 3.4 aufgeführten Standardverfahren gelöst werden. Es ist hierbei zu beachten, dass im Gegensatz zum klassischen Syntheseproblem keine Lösungsverfahren für rationale Optimierungsprobleme verwendet werden können. Dies liegt daran, dass die Multiplikation eines Lösungsvektors mit dem Hauptnenner der Einträge des Lösungsvektors zu einem bzgl. der Zielfunktion nichtoptimalem zulässigen Vektor führen kann.
Wie im Falle von Unterabschnitt 5.4.2 eignet sich die Berechnung einer Erzeugendensystem-Repräsentation zu Optimierungszwecken nur für eine Vorverarbeitungsphase.
Version Schranke: Im Kontext des Syntheseproblems mit Optimierung ist es natürlich ebenfalls möglich, eine obere Schranke für die betrachtete Zielfunktion zu betrachten. In diesem Fall wird dann ein S/T-Netz (N, m_0), $N = (P, T, W)$, mit $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$ gesucht, welches $max\{m_0(p) + \sum_{t\in T}(W(p,t) + W(t,p)) \mid p \in P\} \leqslant b$ für eine gegebene Schranke b erfüllt. Zur Lösung dieses Problems lässt sich wieder obiges Verfahren verwenden. Wenn hierbei allerdings einer der Lösungsvektoren $\mathbf{x}$ der zu betrachtenden Optimierungsprobleme nicht $\sum_{i=0}^{2m} x_i \leqslant b$ erfüllt, so kann direkt eine negative Antwort auf die Problemversion gegeben werden.
Version Global: Bei obigen Syntheseverfahren wird für jede einzelne Stelle eine Optimierung durchgeführt. Es ist darüberhinaus auch möglich, eine globale Optimierung für alle Stellen eines Netzes zu betrachten. Eine entsprechende Problemstellung ist die Suche nach einem S/T-Netz (N, m_0), $N = (P, T, W)$, mit $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$, das für jedes S/T-Netz (N', m_0'), $N' = (P', T', W')$, mit $\mathfrak{Bpo}(N', m_0') = PS(\mathcal{L})$ die Beziehung $\sum_{p\in P}(m_0(p) + \sum_{t\in T}(W(p,t) + W(t,p))) \leqslant \sum_{p\in P'}(m_0(p) + \sum_{t\in T'}(W(p,t) + W(t,p)))$ erfüllt. Es soll also ein Netz mit einer minimalen Gesamtsumme von Kantengewichten und Anfangsmarkierungen synthetisiert werden, d.h. es wird eine globale Zielfunktion für das zu konstruierende Netz betrachtet.
Dieses Problem lässt sich mit dem Verfahren zur Berechnung einer Schrittseparations-Repräsentation lösen, indem alle Partitionen $\{\mathcal{L}_{ws}^1 \ldots \mathcal{L}_{ws}^a\}$ der Menge der falschen Schrittfortsetzungen $\mathcal{L}_{ws}$ betrachtet werden. Für eine Partitionen wird dann versucht, für jede der a Mengen der Partition ein optimales zulässiges Stellentripel zu finden, welches alle falschen Schrittfortsetzungen der Menge separiert. Das bedeutet im Falle von Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$, dass für alle $i \in \{1, \ldots, a\}$ ein Optimierungsproblem gelöst wird, welches sich aus der üblichen linear algebraischen Charakterisierung von Schritt-Transitions-Regionen $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$, einer $\mathbf{b}_\sigma$-Ungleichung für jede falsche Schrittfortsetzung $\sigma \in \mathcal{L}_{ws}^i$ sowie der Zielfunktion $min! \sum_{i=0}^{2m} x_i$ zusammensetzt. Sind alle a Optimierungsprobleme lösbar, so definieren die a Lösungsvektoren eine bzgl. der betrachteten Partition von $\mathcal{L}_{ws}$ optimale Schrittseparations-Repräsentation. Durch einen Vergleich der optimalen Schrittseparations-Repräsentationen aller Partitionen von $\mathcal{L}_{ws}$ ergibt sich eine global optimale Schrittseparations-Repräsentation (N, m_0). Mit einem Übereinstimmungstest kann geprüft

werden, ob $\mathfrak{Bpo}(N, m_0) = PS(\mathcal{L})$. Im positiven Fall stellt (N, m_0) eine positive Lösung für die Problemversion dar. Im negativen Fall hat das Problem eine negative Antwort.
Das hier betrachtete Syntheseproblem mit globaler Optimierung lässt sich auch wieder in ein einzelnes ganzzahliges lineares Optimierungsproblem übersetzen. Wir betrachten wiederum das Prinzip der Berechnung einer Schrittseparations-Repräsentation und Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$. Dann kann ähnlich wie im Falle des ganzzahligen Zulässigkeitsproblems aus Unterabschnitt 5.4.2 vorgegangen werden. Es gibt nur zwei Unterschiede. Zuerst einmal betrachten wir so viele Stellen wie es falsche Schrittfortsetzungen gibt. Mehr Stellen sind sicherlich nicht notwendig und geringere Stellenanzahlen sind dadurch berücksichtigt, dass auch Nullstellen zugelassen sind. Zweitens ergänzen wir das Zulässigkeitsproblem um eine entsprechende Zielfunktion, so dass ein ganzzahliges lineares Optimierungsproblem entsteht. Dieses stellt sich wie folgt dar.

- $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x}^i \geqslant \mathbf{o}$, $i \in \{1, \ldots, |\mathcal{L}_{ws}|\}$;
- $-k \cdot s_{\sigma,i} + \mathbf{b}_{\sigma}^{SSS} \cdot \mathbf{x}^i < \mathbf{o}$, $i \in \{1, \ldots, |\mathcal{L}_{ws}|\}$, $\sigma \in \mathcal{L}_{ws}$;
- $\sum_{i=1}^{|\mathcal{L}_{ws}|} s_{\sigma,i} \leqslant |\mathcal{L}_{ws}| - 1$, $\sigma \in \mathcal{L}_{ws}$;
- $\mathbf{x}^i \in \mathbb{N}^{2m+1}$, $i \in \{1, \ldots, |\mathcal{L}_{ws}|\}$;
- $k \in \mathbb{N}$;
- $s_{\sigma,i} \in \{0, 1\}$, $i \in \{1, \ldots, |\mathcal{L}_{ws}|\}$, $\sigma \in \mathcal{L}_{ws}$;
- $\min! \sum_{i=1}^{|\mathcal{L}_{ws}|} \sum_{j=0}^{2m} x_j^i$.

Version Trade-Off: Die Entwicklung guter Lösungsverfahren von Syntheseproblemen, welche Optimierungsaspekte berücksichtigen, ist eine der wichtigsten Herausforderungen im Hinblick auf die praktische Anwendbarkeit von Petrinetzsynthese [67]. Insbesondere lassen sich durch Optimierungsverfahren die synthetisierten Netze weiter vereinfachen als bisher diskutiert, wenn nicht mehr vorausgesetzt wird, dass die gegebene partielle Sprache exakt reproduziert werden muss. Natürlich soll die Sprache aber auch dann noch möglichst gut widergespiegelt werden. Hier gibt es typischerweise einen Trade-Off zwischen einem einfacheren, weniger präzisen Netz und einem komplexeren, dafür aber präziseren Netz. Um dies zu berücksichtigen, werden auf der einen Seite Kantengewichte und Anfangsmarkierungen von Stellen (oder ähnliche Parameter, welche die Komplexität eines Netzes widerspiegeln) als strukturelle Kosten betrachtet, welche, wie zuvor diskutiert, minimiert werden sollen. Auf der anderen Seite werden Abläufe eines synthetisierten Netzes, welche nicht durch die gegebene partielle Sprache spezifiziert sind (oder eine andere Art von unerwünschtem Verhalten), als Verhaltenskosten betrachtet, welche ebenfalls minimiert werden sollen. Die Anforderung, dass ein zu synthetisierendes Netz die gegeben Sprache exakt reproduziert wird also dadurch ersetzt, dass das Ablaufverhalten des Netzes die Sprache immer noch enthalten soll und zugleich Kosten für zusätzliches unerwünschtes Verhalten des Netzes betrachtet werden. Die zentrale Forderung an ein zu synthetisierendes Netz ergibt sich dann dadurch, dass die Gesamtkosten des Netzes, welche sich aus den strukturellen Kosten und den Verhaltenskosten zusammensetzen, minimiert werden sollen. In diesem Rahmen sind

somit Stellen mit kleinen Kantengewichten und Anfangsmarkierungen, welche viele falsche Schrittfortsetzungen ausschließen, wünschenswert. Solche Stellen führen zu geringen strukturellen Kosten und reduzieren die Verhaltenskosten stark. Auf eine exakte Problemspezifikation und eine Diskussion von möglichen Lösungsverfahren für dieses Syntheseproblem, welches einen Trade-Off zwischen Netzkomplexität und unerwünschtem Netzverhalten sucht, wird hier nicht eingegangen. Je nachdem wie eine entsprechende Problemspezifikation formuliert ist, ergeben sich komplexe, auch nicht-lineare Optimierungsprobleme und Gleichgewichtsprobleme aus der Spieltheorie. Zur Lösung dieser Probleme sind insbesondere auch heuristische Verfahren und Approximations-Verfahren sinnvoll. Es sei zuletzt noch erwähnt, dass bei dem beschriebenen Ansatz natürlich auch weitere Arten von Kosten betrachtet werden können, beispielsweise Kosten für die Kommunikation zwischen verteilten Teilnetzen eines Netzes oder Kosten, welche durch ein Übereinstimmungsmaß bestimmt sind. Beispiele für die Betrachtung von Kosten im Rahmen der Konstruktion von Petrinetzen finden sich in [21, 199, 193].

5.4.7 *Weitere Varianten*

Neben den bisher diskutierten Varianten der klassischen Synthesefragestellung sollen nun noch einige weitere Synthesefragestellungen kurz angesprochen werden. Wir beginnen mit fünf Problemen, welche nur leichte Abwandlungen der klassischen Synthesefragestellung darstellen. Diese können entsprechend auch durch geringe Abwandlungen der Syntheseverfahren aus Kapitel 4 gelöst werden. Für zwei weitere Varianten der klassischen Synthesefragestellung, welche Modularität und Vereinfachungsmöglichkeiten im Rahmen von Petrinetzsynthese betreffen, gibt es bisher weder aus theoretischer noch aus praktischer Sicht im Allgemeinen zufriedenstellende Lösungsverfahren. Daher skizzieren wir für diese beiden Probleme nur interessante Lösungsansätze.

Variante mit vorgegebenen Stellen oder Netzteilen: Im Vorfeld der Modellierung eines Systems stehen in einigen Fällen schon Modelle für Teile des Systems zur Verfügung, beispielsweise existierende Modelle von Teilen eines Systems, unveränderte Teilprozesse bei der Aktualisierung eines Geschäftsprozessmodells oder ein Modell einer Anlage, welche noch durch ein Kontrollmodell ergänzt werden soll. In anderen Fällen, insbesondere im Rahmen von Produktionsprozessen, liegen im Vorfeld der Modellierung wichtige Informationen über Ressourcen vor. Solche Informationen können häufig direkt in Informationen über Stellen eines zu konstruierenden Petrinetzmodells übersetzt werden. Werden die durch die Aktivitäten gegebenen Transitionen des Petrinetzes um entsprechende Stellen ergänzt, so liegt auch in diesem Fall ein Modell für ein Teil des zu konstruierenden Netzes vor. Insgesamt ergibt sich also das Problem, dass neben der Szenario-Spezifikation eines Systems durch eine partielle Sprache und den damit gegebenen Transitionen des zu erstellenden Netzes auch einige Stellen des Netzes mit zugehörigen Kantengewichten und Anfangsmarkierungen vorgegeben sind. Um solche Stellen im Rahmen der Syntheseverfahren aus Kapitel 4 zu berücksichtigen, müssen diese nur um einen vorausgehenden Test ergänzt werden, welcher prüft, ob die Stellen zulässig sind. Im negativen Fall hat das entsprechende Syntheseproblem eine negative Antwort. Im positiven Fall sind dann die Standard-Verfahren verwendbar, wobei

das synthetisierte Netz zusätzlich noch um die vorgegebenen Stellen ergänzt wird. Hierbei können die vordefinierten Stellen bei der Berechnung einer Schrittseparations-Repräsentation auch zum Verhindern falscher Schrittfortsetzungen genutzt werden. Dies lässt sich dadurch erreichen, dass diese Stellen wie schon zu dem zu synthetisierenden Netz hinzugefügte Stellen behandelt werden.

Variante mit einer Menge von Sprachen: Um Unsicherheiten und Unklarheiten bei gewissen Szenario-Informationen zu berücksichtigen, kann es sinnvoll sein, eine Menge von möglichen Ablauf-Spezifikationen für ein Ziel-Systems zu erstellen. Anstelle einer einzelnen partiellen Sprache ergibt sich in einem solchen Fall eine endliche Menge von partiellen Sprachen als Eingabe für einen Synthesealgorithmus. Gesucht ist dann ein Netz, dessen Ablaufverhalten einer der gegebenen Sprachen entspricht. Diese Problemvariante ist eine Verfeinerung des Syntheseproblems mit Sprach-Schranken und kann analog motiviert werden. Um das Problem zu lösen, können einfach die üblichen Syntheseverfahren aus Kapitel 4 auf jede der gegebenen partiellen Sprachen angewendet werden. Dies ist natürlich nicht mehr möglich, wenn eine endlich spezifizierte unendliche Menge von Sprachen betrachtet werden soll. Auf diese interessante Verallgemeinerung wird aber hier nicht näher eingegangen, sondern wir verweisen nur noch darauf, dass ein ähnliches Problem für Transitionssysteme in [19] betrachtet wird.

Variante mit Zusatzanforderungen: Wir interessieren uns nun für die Situation, dass eine durch eine partielle Sprache gegebene Ablauf-Spezifikationen um gewisse Zusatzanforderungen an das zu synthetisierende Systemmodell ergänzt ist. Es gibt natürlich sehr viele denkbare interessante Zusatzforderungen. Wir wollen hier an einigen Beispielen darstellen, dass sich viele solcher Forderungen im Rahmen der Syntheseverfahren aus Kapitel 4 berücksichtigen lassen, indem die linear algebraischen Charakterisierungen der Regionendefinitionen um geeignete Ungleichungen erweitert werden. Wir betrachten wieder Schritt-Transitions-Regionen von $\mathcal{L}$ bzgl. $T = \{t_1, \ldots, t_m\}$ und die entsprechende linear algebraische Charakterisierung $\mathbf{A}_{\mathcal{L}}^{ST} \cdot \mathbf{x} \geqslant \mathbf{0}, \mathbf{x} \geqslant \mathbf{0}$. Eine Beschränkung von Kantengewichten des zu synthetisierenden Netzes durch bestimmte Werte kann ganz einfach durch Ungleichungen, welche die entsprechenden Variablen von Regionen beschränken, realisiert werden. So beschränkt eine Ungleichung der Form $x_i \leqslant k$ das Gewicht der zu x_i gehörigen Kante durch den Wert k. Eine Beschränkung von Markierungen in bestimmten Systemzuständen, wobei ein Systemzustand wie in Unterabschnitt 5.4.3 durch ein Präfix eines Elementes der Sprache gegeben ist, kann durch Ungleichungen, welche die Folgemarkierungen entsprechender Präfixe beschränken, sichergestellt werden. Solche Ungleichungen ergeben sich ähnlich wie in Unterabschnitt 5.4.3. Für ein Präfix $bpo \in Pref(\mathcal{L})$ stellt die Ungleichung $\mathbf{p}_{bpo} \cdot \mathbf{x} \leqslant k$ mit $\mathbf{p}_{bpo,0} = 1$, $\mathbf{p}_{bpo,i} = |V|_l(t_i)$ für $i = 1, \ldots, m$ und $\mathbf{p}_{bpo,i} = -|V|_l(t_{i-m})$ für $i = m+1, \ldots, 2m$ sicher, dass die durch eine entsprechende Region definierte Stelle in der Folgemarkierung des Präfixes höchstens k Marken enthält. Des Weiteren lässt sich eine Transitionsinvariante für ein Netz garantieren, indem eine Gleichung, welche eine entsprechende Linearkombination der Variablen von Regionen gleich Null setzt, betrachtet wird. So kann eine Transitionsinvariante $\tau \in \mathbb{N}^T$ durch die Gleichung $\mathbf{inv}_\tau \cdot \mathbf{x} = 0$ mit $\mathbf{inv}_{\tau,0} = 0$, $\mathbf{inv}_{\tau,i} = \tau(t_i)$ für $i = 1, \ldots, m$ und $\mathbf{inv}_{\tau,i} = -\tau(t_{i-m})$ für $i = m+1, \ldots, 2m$ kodiert werden.

Mit einem ähnlichen Vorgehen wie bei den bisher genannten Beispielen können verschiedenste strukturelle Beschränkungen für das zu synthetisierende Netz berücksichtigt werden. Beispiele hierfür sind die Synthese von Free-Choice-Netzen, markierten Graphen oder Zustandsmaschinen. Des Weiteren können auch fortschrittliche Korrektheitseigenschaften für Petrinetze in die verschiedenen Syntheseverfahren integriert werden. Ein interessantes Beispiel hierzu ist die Soundness-Eigenschaft von Workflownetzen [6, 69]. Es bleibt hier noch anzumerken, dass durch etliche der Beispiel-Zusatzanforderungen inhomogene Ungleichungssysteme entstehen. In diesem Fall ist die Berechnung einer Erzeugendensystem-Repräsentation nicht mehr möglich und bei der Berechnung einer Schrittseparations-Repräsentation müssen dann ganzzahlige Optimierungsverfahren verwendet werden. Der Ansatz der Übersetzung von Anforderungen an ein Petrinetz in entsprechende Ungleichungen wurde in der Literatur inbesondere schon in den Arbeiten [48, 69] verfolgt. Dort finden sich einige zu den Ausführungen dieses Absatzes sehr ähnliche Vorgehensweisen.

Variante Beschränktheit: In der Petrinetztheorie stellt Beschränktheit eine wichtige Eigenschaft von Netzen dar. Beschränktheit ist für die Theorie und die Anwendung von Petrinetzen relevant. Beispielsweise spielt sie eine Rolle bei der Vermeidung des Überlaufs von Stellen, bei bestimmten Analyseverfahren, welche nur für beschränkte Netze anwendbar sind, und manchmal auch für das System-Design. In einigen Fällen, beispielsweise bei der Modellierung von Ressourcen, ist auch k-Beschränktheit, also die Beschränktheit von Stellen durch einen festen Wert k, interessant. In unserer Situation von endlichen partiellen Sprachen sind die synthetisierten Netze immer beschränkt, wenn das Syntheseproblem eine positive Antwort hat. Generell lässt sich Beschränktheit im Falle von endlichen partiellen Sprachen durch zulässige Stellen st^{t}, wie in Unterabschnitt 4.5.1 betrachtet, sicherstellen. Um Beschränktheit durch einen festen Wert k zu garantieren, beispielsweise Einssicherheit, muss jeder spezifizierte Systemzustand, d.h. die Folgemarkierung jedes Präfixes der gegebenen Sprache, wie im letzten Absatz erklärt, durch k beschränkt werden. Dann ist das synthetisierte Netz k-beschränkt, wenn es nur das spezifizierte Verhalten aufweist. Um k-Beschränktheit auch dann zu garantieren, wenn das Syntheseproblem eine negative Antwort besitzt, können die Stellen des synthetisierten Netzes ganz einfach durch Komplementstellen bzgl. der Schranke k ergänzt werden. Dadurch wird das Verhalten des Netzes nicht zu sehr eingeschränkt, da für die ursprünglich synthetisierten Stellen des Netzes bzgl. des spezifizierten Verhaltens ohnehin k-Beschränktheit sichergestellt ist und somit alle derartigen Komplementstellen bzgl. der Spezifikation zulässig sind. In der Literatur der Petrinetzsynthese werden häufig beschränkte Netze betrachtet [13]. In [66] wird auch eine abgeschwächte Form von Beschränktheit berücksichtigt.

Variante verteilte Netze: Petrinetze ermöglichen die Modellierung wahrer Nebenläufigkeit. Dementsprechend stellt die Modellierung von hochgradig nebenläufigen oder verteilten Systemen ein wichtiges Anwendungsgebiet von Petrinetzen dar. Für die Modellierung solcher Systeme bieten sich verteilte Architekturen von Petrinetzen an. Einen entsprechenden Ansatz stellen sog. verteilte Petrinetze dar [15, 66, 64]. Diese bestehen aus verteilten Komponenten, welche als Standorte bezeichnet werden. Transitionen eines Standortes dürfen nur Marken von Stellen desselben Standortes konsumieren, aber sie dürfen zu Kommu-

nikationszwecken Marken in Stellen anderer Standorte produzieren. Somit bestehen verteilte Netze aus lokalen Komponenten, welche durch ein asynchrones Versenden von Nachrichten kommunizieren. Die Ausgangslage für eine Synthese solcher Netze muss so gestaltet sein, dass zusätzlich zu einer partiellen Sprache noch eine Aufteilung der Transitionen der Sprache zu Standorten des zu synthetisierenden Netzes gegeben ist. Dieses Problem kann dadurch gelöst werden, dass die Standorte einzeln betrachtet werden. Die Menge der zulässigen Stellentripel eines Standortes ist dadurch gegeben, dass nur die bzgl. des betrachteten Standortes erlaubten Kanten berücksichtigt werden. In einer linear algebraischen Charakterisierung einer Regionendefinition werden also die Variablen weggelassen, welche Kanten von der zu definierenden Stelle zu einer Transition eines anderen Standortes entsprechen. Dann können die Stellen jedes Standortes mit den üblichen Syntheseverfahren aus Kapitel 4, also den Verfahren zur Konstruktion entsprechender Repräsentations-Netze, berechnet werden. Werden diese zusammengefügt, so ergibt sich ein Netz, welches entweder die Problemvariante positiv löst, oder aber das Problem hat eine negative Antwort. Um dies zu beantworten, kann einer der bekannten Übereinstimmungstests aus Abschnitt 4.5 verwendet werden. Für Transitionssysteme ist die Synthese verteilter Netze in den Arbeiten [15, 66, 64, 67] angesprochen.

Variante mit Modularität: In der Praxis gibt es Beispiele für sehr große Spezifikationen. Dies gilt insbesondere für automatisch generierte Spezifikationen, aber in großen Projekten können auch händisch entwickelte Spezifikationen sehr groß werden. In solchen Fällen kann es passieren, dass bei den bisher vorgestellten Syntheseverfahren Performance-Probleme auftreten oder die Komplexität der erzeugten Netze zu groß wird. Um in solchen Fällen dennoch Syntheseverfahren sinnvoll anwenden zu können, bietet sich eine modulare Vorgehensweise an. Die Idee hierbei ist es, kleinere Probleminstanzen zu behandeln, indem die gegebene partielle Sprache sinnvoll in modulare Teile zerlegt wird. Die Teile sollen so klein sein, dass sich für diese dann die üblichen Syntheseverfahren wieder nutzen lassen.

Grundsätzlich sehen wir drei Möglichkeiten für eine geeignete Aufteilung einer partiellen Sprache. Die erste Möglichkeit besteht darin, ähnlich wie im letzten Absatz, die Menge der Transitionen derart zu partitionieren, dass eine Aufteilung in Transitionsmengen möglichst unabhängiger Teile des Systems entsteht. Jede der Transitionsmengen der Partition definiert dann eine kleinere partielle Sprache, indem die Ausgangssprache auf die Transitionen der jeweiligen Menge projeziert wird, d.h. in jeder BPO werden alle Ereignisse anderer Transitionen weggelassen. Dann wird für jede der projezierten Sprachen ein Netz synthetisiert und die entsprechenden Netze unabhängig zueinander arrangiert. Eine besondere Herausforderung ergibt sich dadurch, dass die unabhängigen Netzteile noch durch geeignete Kommunikationsstellen synchronisiert werden sollten. Hierzu müssen geeignete Informationen aus der ursprünglichen partiellen Sprache gewonnen werden. Um so besser die ursprüngliche Aufteilung der Transitionen zu unabhängigen Netzteilen gelingt, um so weniger Stellen müssen hier berücksichtigt werden.

Die zweite Möglichkeit ist, die partielle Sprache direkt in kleinere partielle Sprachen zu partitionieren. Jede der entstehenden Sprachen repräsentiert dann einen alternativen Teil des Systems. Es kann ein Netz für jede Sprache synthetisiert werden, allerdings besitzen die verschie-

denen Netze dann normalerweise übereinstimmende Transitionen. Die Grundidee ist nun, zu Beginn des Systemmodells eine nichtdeterministische Auswahl einzuführen, welche zu einer der Anfangsmarkierungen der verschiedenen Netze führt. Die entscheidende Herausforderung bei diesem Ansatz besteht darin, das durch die übereinstimmenden Transitionen entstehende Beschriftungs-Splitting soweit wie möglich zu vermeiden. Hierbei ist zum einen eine anfangs geschickt gewählte Aufteilung der partiellen Sprache wichtig. Zum anderen müssen geeignete Verfahren zum Verschmelzen der gleichbeschrifteten Transitionen entwickelt werden.
Eine dritte Möglichkeit besteht darin, die einzelnen BPOs der partiellen Sprache nach bestimmten Kriterien zu zerlegen. Hierbei kann eine BPO entweder in möglichst unabhängige oder in möglichst sequentiell abhängige Teile zerlegt werden. Die so entstehenden Teile aller BPOs der Sprache werden dann geeignet zu kleineren partiellen Sprachen gruppiert, beispielsweise zu einer Gruppe von BPO-Teilen, die den Anfang des Systemverhaltens modellieren. Für jede Gruppe wird dann ein Netz synthetisiert. Die einzelnen Netze müssen anschließend verbunden werden. Für unabhängige Netze bedeutet dies wiederum die Berücksichtigung geeigneter Synchronisations-Stellen. Für sequentiell anzuordnende Netze müssen geeignete nichtdeterministische Übergänge von Endmarkierungen eines Netzes zu Anfangsmarkierungen folgender Netze eingeführt werden. Dies erfordert normalerweise auch die Einführung zusätzlicher Verbindungsstellen. In jedem Fall ergibt sich auch hier wieder die Problematik des Beschriftungs-Splitting. Die wichtigste Herausforderung bei dieser Herangehensweise stellt allerdings das Finden einer geeigneten Zerlegung der BPOs der ursprünglichen Sprache dar.
Natürlich lassen sich insgesamt die drei diskutierten Modularisierungsmöglichkeiten auch kombinieren. Zuletzt bleibt allerdings zu erwähnen, dass die hier präsentierten Ideen für modulare Syntheseansätze noch viele offene Fragen beinhalten. Generell wurde das wichtige Problem der Modularität bisher nur im Rahmen der Synthese von Kontrollmodellen genauer betrachtet [67].
Variante mit Vereinfachungen: Im Hinblick auf praktische Anwendbarkeit von Petrinetzsynthese spielen, wie schon erwähnt, zwei Punkte eine entscheidende Rolle. Zum einen ist die Performance der Syntheseverfahren wichtig und zum anderen ist es wichtig, dass die Verfahren klare, intuitive Systemmodelle, welche kein „Spaghetti"-Code-Aussehen aufweisen, generieren. Eine mögliche Herangehensweise, um diese zwei Aspekte zu berücksichtigen, ist eine Vereinfachung der Synthesefragestellung, so dass sich vereinfachte Verfahren verwenden lassen. Es wird also keine exakte Reproduktion des spezifizierten Verhaltens mehr gefordert, sondern die Synthesefrage orientiert sich spezifisch an möglichen vereinfachten Synthesevorgehen. Wir skizzieren hier einige Möglichkeiten für solche eher heuristisch anmutende Ansätze. Vereinfachungsmöglichkeiten bestehen zum einen darin, die gegebene partielle Sprache zu vereinfachen. Es ist beispielsweise möglich, einige weniger relevante Abläufe der Spezifikation wegzulassen bzw. gar einige weniger relevante Transitionen aus der Spezifikation zu entfernen oder sich wiederholendes Verhalten der gegebenen Sprache vereinfacht zu strukturieren. Zum anderen lassen sich natürlich direkt die Syntheseverfahren vereinfachen. Es bietet sich beispielsweise an, nur die in einem gewissen Sinne wichtigsten Stellentripel zu dem zu

synthetisierenden Netz hinzuzufügen oder schon von Anfang an nur eine geeignete Auswahl aller falscher Schrittfortsetzungen zu betrachten bzw. nicht alle Erzeugenden-Regionen zu konstruieren. Es können auch einfach zu komplexe Stellen mit hohen Kantengewichten und Anfangsmarkierungen ignoriert werden. Alternativ können bei komplexen Stellen auch die Kantengewichte künstlich reduziert werden oder bestimmte Kanten zwischen relativ unabhängigen Netzkomponenten weggelassen werden. Des Weiteren vereinfacht ein hoher Grad an Nebenläufigkeit typischerweise die zu synthetisierenden Netze. Ein solcher kann durch entsprechende Modifikationen der Ungleichungssysteme erreicht werden. Schließlich lässt sich ein synthetisiertes Netz auch im nachhinein noch durch verschiedenste Techniken vereinfachen. Das in Kapitel 2 diskutierte Process-Mining stellt, wie dort auch ausführlich diskutiert, ein wohlbekanntes Beispiel dar, bei dem vereinfachte Synthese- und Konstruktionsverfahren für Netze eine wichtige Rolle spielen [3, 1, 69, 7, 84, 211, 51, 53]. Insgesamt lässt sich zusammenfassen, dass heuristische Vereinfachungen von Syntheseverfahren und generell heuristische Netz-Konstruktionsverfahren ein im Hinblick auf viele Anwendungsbereiche wichtiges Thema darstellen, welches wir aber im Rahmen dieser Arbeit nicht näher diskutieren werden.
Zum Abschluss dieses Abschnittes sei noch erwähnt, dass es der Natur der Sache entsprechend auch noch weitere, hier nicht genannte mögliche Varianten der klassischen Synthesefragestellung gibt. Allerdings sind in diesem Abschnitt viele interessante Probleme angesprochen, welche nach unserem Wissen insbesondere alle bisher in der Literatur betrachteten modifizierten Synthesefragestellungen für Petrinetze in unserem Kontext behandeln.

6

ABSCHLUSSBETRACHTUNGEN

Abschließend fassen wir die Inhalte der Arbeit noch einmal kurz zusammen und geben einen Ausblick auf interessante weiterführende Forschungsthemen.

6.1 ZUSAMMENFASSUNG

Die vorliegende Arbeit hat ausführlich das Thema der Synthese von Petrinetzen aus halbgeordneten Abläufen behandelt. Zunächst haben wir in Kapitel 2 die Bedeutung dieser Problemstellung im Rahmen der Modellierung von Systemen verschiedenster Art dargestellt. Wir haben hierzu den sog. streng ablauforientierten Modellierungsansatz eingeführt und umfassend diskutiert. Die Idee dieses Ansatzes besteht darin, die Abläufe eines Systems in den Vordergrund der Modellierung zu stellen. In unserem Kontext ergibt sich für einen Modellierer die Aufgabe, die Abläufe des Systems zu erheben und anschließend in der Form von BPOs zu modellieren. Der entscheidende Schritt des Ansatzes ist dann eine automatische Übersetzung der Modelle der einzelnen Abläufe des Systems in ein zu den gegebenen Abläufen konsistentes Petrinetzmodell des Systems. Hierfür sind nun entsprechende Syntheseverfahren notwendig. Wir haben die Nützlichkeit eines solchen Vorgehens anhand von drei konkreten Anwendungsbereichen für konzeptuelle Modellierung dargelegt. Zuerst haben wir den in der Praxis sehr bedeutsamen Bereich des Softwareengineering, also der Entwicklung von Modellen für Softwaresysteme bzw. Programme, betrachtet. Anschließend haben wir das Feld des Geschäftsprozess-Managements bzw. der Geschäftsprozessmodellierung diskutiert, welches sich in den letzten Jahren in der Industrie auf breiter Front zu einem viel beachteten und wichtigen Thema entwickelt hat. Schließlich haben wir den Modellierungsansatz im Kontext des noch sehr jungen, aber mehr und mehr an Bedeutung erlangenden Bereiches der Lernprozessmodellierung dargestellt. Für alle drei Bereiche haben wir mit konkreten Beispielen illustriert, dass generell der streng ablauforientierte Modellierungsansatz und speziell die Synthese von Petrinetzen aus partiellen Sprachen zur Modellierung entsprechender Systeme sehr sinnvoll und hilfreich sein kann.
Nach diesen einführenden und motivierenden praxisnahen Ausführungen haben wir das Problem der Synthese von Petrinetzen aus halbgeordneten Abläufen auf einer theoretischen Ebene umfassend untersucht. Hierzu haben wir ein Baukasten-Modell verwendet, welches einerseits verschiedene Bausteine eines entsprechenden Syntheseverfahrens und andererseits auch verschiedene Bausteine eines entsprechenden Syntheseproblems unterscheidet. Als Bausteine für ein Lösungsverfahren – diese bezeichnen wir als Lösungsdimension unseres Baukastens – haben

wir eine Regionendefinition, ein Berechnungsverfahren für Regionen und einen Übereinstimmungstest für das berechnete Netz identifiziert. Die Bausteine für eine Syntheseproblemstellung – diese subsumieren wir unter dem Begriff der Problemdimension unseres Baukastens – sind eine Sprachklasse und ein Sprachtyp sowie eine Netzklasse und eine Variante der Synthesefragestellung.
Zunächst haben wir uns in Kapitel 4 dann auf die konkrete Standard-Syntheseproblemstellung der klassischen (exakten) Synthese eines S/T-Netzes aus einer endlichen partiellen Sprache festgelegt. Für diese haben wir die verschiedenen Bausteine für ein Lösungsverfahren ausführlich diskutiert. Zuerst haben wir die Regionendefinitionen der Schritt-Transitions-Region, der BPO-Transitions-Region, der Markenfluss-Region und der Schritt-Transitionssystem-Region entwickelt. Anschließend haben wir, ausgehend von einer beliebigen der betrachteten Regionendefinitionen, Verfahren zur Berechnung einer Schrittseparations-Repräsentation, einer BPO-Separations-Repräsentation und einer Erzeugendensystem-Repräsentation erläutert. Für einen abschließenden Übereinstimmungstest eines der berechneten Repräsentations-Netze mit der betrachteten partiellen Sprache haben wir den optimistischen und den pessimistischen Übereinstimmungstest vorgestellt. Die verschiedenen Ansätze für die verschiedenen Bausteine der Lösungsdimension lassen sich beliebig zu einem Syntheseverfahren kombinieren. In diesem Kontext haben wir weiter ein Werkzeug präsentiert, welches die aus entsprechenden Kombinationen der Ansätze resultierenden Syntheseverfahren zur Verfügung stellt. Schließlich haben wir die verschiedenen resultierenden Verfahren auch ausführlich verglichen. Hierbei haben sich interessante Resultate ergeben, insbesondere sind abhängig von der Struktur der betrachteten partiellen Sprache in unterschiedlichen Situationen verschiedene der Ansätze vorteilhaft.
Wie die Anwendungsbeispiele aus Kapitel 2 gezeigt haben, kommen in der Realität auch verschiedenste Arten von partiellen Sprachen vor, so dass eine Auswahl von Syntheseverfahren mit unterschiedlichen Stärken und Schwächen, wie sie in Kapitel 4 präsentiert wurde, nicht nur aus theoretischer, sondern auch aus praktischer Sicht interessant ist. Insbesondere hat sich im Rahmen des Vergleichs der Verfahren auch gezeigt, dass sie prinzipiell für Anwendungen, wie sie in Kapitel 2 geschildert wurden, tauglich sind.
In Kapitel 5 haben wir dann dargestellt, wie die zuvor entwickelten Syntheseverfahren abgeändert bzw. ergänzt werden müssen, wenn wir die Syntheseproblemstellung verändern. Wir haben für jeden der vier Bausteine der Problemdimension des Synthesebaukastens diskutiert, wie sich Veränderungen auf die Syntheseverfahren auswirken. Dazu haben wir ausgewählte, interessante Instanziierungen der einzelnen Bausteine betrachtet. Als Sprachklassen haben wir zwei Klassen unendlicher partieller Sprachen, welche sich jeweils durch eine bestimmte Art von Termen repräsentieren lassen, untersucht. Als Sprachtypen haben wir geschichtete Sprachen sowie verschiedene Semantiken von partiellen Sprachen diskutiert. Unterschiedliche Netzklassen haben wir mithilfe der parametrischen Definition der Netztypen auf einer allgemeinen Ebene betrachtet. Schließlich haben wir eine ganze Reihe von Varianten der klassischen Synthesefragestellung, welche entweder die Forderung der exakten Synthese abschwächen oder aber sogar zusätzliche Anforderungen an das zu synthetisierende Netz stellen, gelöst.

Betrachten wir wiederum die Anwendungsszenarien aus Kapitel 2, so wird deutlich, dass die vier in Kapitel 5 untersuchten Problem-Bausteine auch für praktische Syntheseproblemstellungen relevant sind. Beispielsweise haben wir in Kapitel 2 einen verteilten Algorithmus mit unendlich vielen Abläufen, welche sich durch einen Term repräsentieren lassen, betrachtet (Baustein Sprachklasse). Wir haben auch dargestellt, dass die Prozessablaufsemantik für partielle Sprachen im Rahmen der Modellierung von Ressourcen in Geschäftsprozessen wichtig sein kann (Baustein Sprachtyp). Weiter haben wir dargelegt, dass einfache S/T-Netze zur Modellierung kollaborativer Lernprozessmodelle nicht geeignet sind, sondern hier Erweiterungen notwendig sind (Baustein Netzklasse). Schließlich haben sich gerade im Bereich des Process-Mining einige über die klassische Synthesefragestellung hinausgehende Problemstellungen wie die Synthese möglichst kompakter Netze ergeben (Baustein Variante).
Insgesamt haben wir in der vorliegenden Arbeit das Problem der Synthese von Petrinetzen aus halbgeordneten Abläufen durch die Betrachtung des Synthesebaukastens systematisch und sehr ausführlich diskutiert. Insbesondere hat sich eine sehr schön abgerundete Darstellung des Themas ergeben. Dabei haben wir heuristische Verfahren und einfache Faltungsverfahren zur Konstruktion von Netzen sowie die Synthese von Netzen mit Transitions-Beschriftungen nicht betrachtet. Im Rahmen unseres Baukastens haben wir ansonsten, soweit wir das beurteilen können, alle Ideen zur Synthese von Petrinetzen aus Sprachen, welche sich in der Literatur finden lassen, aufgegriffen und in unserem Kontext von halbgeordneten Abläufen verwendet und ergänzt. Zu dem Thema der Synthese von Petrinetzen aus Sprachen gibt es in der Literatur Arbeiten zu halbgeordneten Abläufen, Schrittsprachen und sequentiellen Sprachen. Die Ausführungen dieser Arbeit behandeln die Thematik der Synthese von Petrinetzen aus halbgeordneten Abläufen, aus heutigem Forschungsstand betrachtet, in einer umfassenden Weise. Die Überlegungen lassen sich aber auch größtenteils auf Schrittsprachen und sequentielle Sprachen übertragen, welche ja eigentlich jeweils nur einen Spezialfall partieller Sprachen darstellen. Abgesehen von einigen Zusatzüberlegungen, welche in diesen Fällen teilweise notwendig sind, gibt diese Arbeit somit auch einen umfassenden Überblick generell über die Synthese von Petrinetzen aus Sprachen. In den Ausführungen dieser Arbeit sind bekannte Syntheseverfahren bzw. Syntheseansätze für Schrittsprachen und sequentielle Sprachen aus der Literatur berücksichtigt und es werden in diesem Kontext auch etliche Lücken geschlossen. Insbesondere lassen sich entsprechende Ansätze größtenteils sehr gut in den vorgestellten Synthesebaukasten einordnen. Der Baukasten eignet sich somit auch für eine umfassende Darstellung der Synthese von Petrinetzen ganz allgemein aus Sprachen. Mit geeigneten Anpassungen ist er darüber hinaus sogar für die Synthese von Petrinetzen aus Graphen und folglich für eine generelle Diskussion der Synthese von Petrinetzen geeignet.

6.2 AUSBLICK

Wie erläutert, bietet die Arbeit insgesamt einen abgeschlossenen Überblick über das Thema der Synthese von Petrinetzen aus halbgeordneten Abläufen. Insbesondere Kapitel 4, welches sich mit der Lösungsdimension des Synthesebaukastens beschäftigt, behandelt Verfahren zur Syn-

these eines S/T-Netzes aus einer endlichen partiellen Sprache in einer umfassenden Weise. Für weiterführende Arbeiten bietet sich im Kontext von Kapitel 4 allerdings noch die genauere Ausarbeitung von Optimierungen der vorgestellten Syntheseverfahren an. Wir haben in diesem Kapitel für alle vorgestellten Ansätze etliche Optimierungsvorschläge gemacht, ohne diese jedoch detailliert zu untersuchen. Darüber hinaus stellen auch die Berücksichtigung von Transitions-Beschriftungen bei den Syntheseverfahren sowie die Kopplung der Syntheseverfahren mit Faltungsverfahren interessante weiterführende Forschungsgebiete dar. In Kapitel 5, das mit der Problemdimension des Synthesebaukastens befasst ist, stellt sich die Situation etwas anders dar. In diesem Kapitel wird eine Vielzahl interessanter Fragen für weitere Forschungsarbeiten aufgeworfen. Für jeden der vier diskutierten Problem-Bausteine haben wir bestimmte ausgewählte Problemstellungen untersucht. Allerdings sind die Überlegungen natürlich nicht so ausführlich wie für unser Standard-Syntheseproblem, so dass an einigen Stellen noch offene Fragen auftauchen. Außerdem haben wir jeweils noch auf weitere, über die ausgewählten Problemstellungen hinausgehende, interessante Themen im Rahmen des betrachteten Problem-Bausteins hingewiesen. So ist im Kontext von Sprachklassen beispielsweise die Synthese von Petrinetzen aus unendlichen partiellen Sprachen, welche durch Terme mit einem Rekursionsoperator, durch geeignete Ereignisstrukturen mit Cut-Off-Ereignissen oder durch einssichere Petrinetze mit Transitions-Beschriftungen gegeben sind, interessant. Auch die Betrachtung regulärer partieller Sprache ist vielversprechend. Sprachtypen betreffend, gibt es neben geschichteten Ordnungen auch noch weitere zu untersuchende Verallgemeinerungen von BPOs, wie allgemeine gerichtete azyklische Graphen. Darüber hinaus ist die Synthese bzgl. der Prozessablaufsemantik noch nicht ausreichend untersucht und stellt ein spannendes Forschungsfeld dar. Regionen für verschiedene Netzklassen haben wir zwar auf einer allgemeinen Ebene parametrisch diskutiert, konkrete Berechnungsverfahren erfordern jedoch für jede Netzklasse spezifische Überlegungen. Hier besteht auch noch viel Raum für neue Ergebnisse. Bei etlichen der diskutierten Varianten der klassischen Synthesefragestellung haben wir auf offene Fragen hingewiesen. Vor allem Modularität, Optimierungen und Vereinfachungen von Syntheseverfahren sind hier wichtige Themen. Gerade diese drei Gebiete sind im Hinblick auf die praktische Anwendbarkeit von Petrinetz-Syntheseverfahren entscheidend.

Dies führt uns zu dem abschließenden, vielleicht wichtigsten Bereich für weiterführende Forschungsarbeiten, nämlich der Anwendung von Verfahren zur Synthese von Petrinetzen aus halbgeordneten Abläufen in der Praxis. Wir haben in Kapitel 2 im Rahmen des streng ablauforientierten Modellierungsansatzes einige Vorschläge zur Anwendbarkeit der Verfahren für verschiedene Einsatzgebiete dargestellt. Wir haben erläutert, wie entsprechende Syntheseverfahren zur Modellierung von Software, von Geschäftsprozessen und von Lernprozessen genutzt werden können. Wie gezeigt, haben wir in diesem Kontext auch schon einige praktische Erfahrungen sowohl im industriellen als auch im akademischen Bereich gewonnen. Allerdings stellen diese nur einen ersten Schritt in die richtige Richtung dar. Es sind hier noch viele weitere empirische Erfahrungswerte und insbesondere auch umfassende Evaluationen erforderlich. Die tatsächliche praktische Verwendung von Syntheseverfahren für reale Problemstellungen stellt die wichtigste

Aufgabe im Hinblick auf eine potentielle industrielle Nutzbarkeit dar. Sicherlich sind in diesem Zuge dann aber auch auf der theoretischen Ebene noch Anpassungen der Syntheseverfahren an die Bedürfnisse der verschiedenen Anwendungsbereiche sowie weiterführende konzeptuelle Überlegungen notwendig. Auf der Implementierungsebene muss dies natürlich Hand in Hand mit einer entsprechenden Verbesserung der Werkzeugunterstützung gehen.

LITERATURVERZEICHNIS

[1] Aalst, W. M. P. d. ; Günther, C. W.: Finding Structure in Unstructured Processes: The Case for Process Mining. In: Basten, T. (Hrsg.) ; Juhás, G. (Hrsg.) ; Shukla, S. K. (Hrsg.): *Proc. of ACSD 2007*, IEEE Computer Society, 2007, S. 3–12 (Zitiert auf Seiten 13, 48, 66, 67, 68, 73, 74, und 339.)

[2] Aalst, W. M. P. d. ; Dongen, B. F. ; Günther, C. W. ; Mans, R. S. ; Medeiros, A. K. A. ; Rozinat, A. ; Rubin, V. ; Song, M. ; Verbeek, H. M. W. ; Weijters, A. J. M. M.: ProM 4.0: Comprehensive Support for real Process Analysis. In: Kleijn, J. (Hrsg.) ; Yakovlev, A. (Hrsg.): *Proc. of ICATPN 2007, LNCS 4546*, Springer, 2007, S. 484–494 (Zitiert auf Seiten 14, 67, 222, und 249.)

[3] Aalst, W. M. P. d. ; Dongen, B. F. ; Herbst, J. ; Maruster, L. ; Schimm, G. ; Weijters, A. J. M. M.: Workflow Mining: A Survey of Issues and Approaches. In: *Data Knowl. Eng.* 47 (2003), Nr. 2, S. 237–267 (Zitiert auf Seiten 13, 14, 48, 66, 67, und 339.)

[4] Aalst, W. M. P. d. ; Weijters, T. ; Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. In: *IEEE Trans. Knowl. Data Eng.* 16 (2004), Nr. 9, S. 1128–1142 (Zitiert auf Seite 14.)

[5] Aalst, W.M.P.d. (Hrsg.) ; Desel, J. (Hrsg.) ; Oberweis, A. (Hrsg.): *Business Process Management, Models, Techniques, and Empirical Studies, LNCS 1806*. Springer, 2000 (Zitiert auf Seite 46.)

[6] Aalst, W.M.P.d. ; Hee, K.: *Workflow Management: Models, Methods, and Systems.* MIT Press, 2002 (Zitiert auf Seiten 13, 46, 50, 75, 76, 77, und 336.)

[7] Aalst, W.M.P.d. ; Rubin, V. ; Dongen, B.F. ; Kindler, E. ; Guenther, C.W.: Process Mining: A Two-Step Approach using Transition Systems and Regions. / Department of Technology Management, Eindhoven University of Technology. 2006 (BPM Center Report BPM-06-30). – Forschungsbericht (Zitiert auf Seiten 13, 14, 68, 176, und 339.)

[8] Aleven, V. ; McLaren, B.M. ; Sewell, J. ; Koedinger, K.R.: The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains. In: Ikeda, M. (Hrsg.) ; Ashley, K.D. (Hrsg.) ; Chan, T.-W. (Hrsg.): *Proc. of ITS 2006, LNCS 4053*, Springer, 2006 (Zitiert auf Seite 80.)

[9] Amyot, D. ; Eberlein, A.: An Evaluation of Scenario Notations and Construction Approaches for Telecommunication Systems Development. In: *Telecommunication Systems* 24 (2003), Nr. 1, S. 61–94 (Zitiert auf Seiten 13, 37, 38, und 39.)

[10] Antonellis, V. D. ; Demo, G. B.: Requirements Collection and Analysis. In: Ceri, S. (Hrsg.): *Methodology and Tools for Data Base Design*. North-Holland, 1983, S. 9–24 (Zitiert auf Seite 37.)

[11] Avis, D. ; Bremner, D. ; Seidel, R.: How Good Are Convex Hull Algorithms? In: *Comput. Geom.* 7 (1997), S. 265–301 (Zitiert auf Seiten 113, 114, 115, und 116.)

[12] Badouel, E. ; Bednarczyk, M. A. ; Darondeau, P.: Generalized Automata and Their Net Representations. In: Ehrig, H. (Hrsg.) ; Juhás, G. (Hrsg.) ; Padberg, J. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Unifying Petri Nets, LNCS 2128*, Springer, 2001, S. 304–345 (Zitiert auf Seite 95.)

[13] Badouel, E. ; Bernardinello, L. ; Darondeau, P.: Polynomial Algorithms for the Synthesis of Bounded Nets. In: Mosses, P. D. (Hrsg.) ; Nielsen, M. (Hrsg.) ; Schwartzbach, M. I. (Hrsg.): *Proc. of TAPSOFT'95, LNCS 915*, Springer, 1995, S. 364–378 (Zitiert auf Seiten 11, 12, 14, 69, 74, 136, 144, 172, 253, und 336.)

[14] Badouel, E. ; Bernardinello, L. ; Darondeau, P.: The Synthesis Problem for Elementary Net Systems is NP-Complete. In: *Theor. Comput. Science* 186 (1997), Nr. 1-2, S. 107–134 (Zitiert auf Seite 11.)

[15] Badouel, E. ; Caillaud, B. ; Darondeau, P.: Distributing Finite Automata Through Petri Net Synthesis. In: *Formal Asp. Comput.* 13 (2002), Nr. 6, S. 447–470 (Zitiert auf Seiten 13, 14, 319, 336, und 337.)

[16] Badouel, E. ; Darondeau, P.: Dualities Between Nets and Automata Induced by Schizophrenic Objects. In: Pitt, D. H. (Hrsg.) ; Rydeheard, D. E. (Hrsg.) ; Johnstone, P. (Hrsg.): *Proc. of CTCS'95, LNCS 953*, Springer, 1995, S. 24–43 (Zitiert auf Seiten 11, 95, 290, und 291.)

[17] Badouel, E. ; Darondeau, P.: On the Synthesis of General Petri Nets. / Inria, Université de Rennes I. 1996 (RR-3025). – Forschungsbericht (Zitiert auf Seiten 11, 12, 14, 95, 96, 136, 172, 173, 177, 193, 290, 291, 298, und 325.)

[18] Badouel, E. ; Darondeau, P.: Theory of Regions. In: Reisig, W. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Lectures on Petri Nets I: Basic Models, LNCS 1491*, Springer, 1998, S. 529–586 (Zitiert auf Seiten 11, 12, 69, 74, 95, 126, 128, 136, 142, 144, 172, 173, 176, 177, 189, 193, 202, 253, 290, 291, 298, und 325.)

[19] Badouel, E. ; Darondeau, P.: The Synthesis of Petri Nets from Path-Automatic Specifications. In: *Inf. Comput.* 193 (2004), Nr. 2, S. 117–135 (Zitiert auf Seiten 13 und 335.)

[20] Barron, B.: When Smart Groups Fail. In: *Journal of the Learning Sciences* 12 (2003), Nr. 3, S. 307–359 (Zitiert auf Seite 74.)

[21] Basile, F. ; Chiacchio, P. ; Seatzu, C.: Optimal Petri Net Monitor Design. In: Caillaud, B. (Hrsg.) ; Darondeau, P. (Hrsg.) ; Lavagno, L. (Hrsg.) ; Xie, X. (Hrsg.): *Synthesis and Control of Discrete Event Systems*, Kluwer, 2002, S. 141–152 (Zitiert auf Seiten 13, 319, und 334.)

[22] Baumgarten, B.: *Petri-Netze*. Spektrum Akademischer Verlag, 1996 (Zitiert auf Seiten 89 und 249.)

[23] Bergenthum, R. ; Desel, J. ; Harrer, A. ; Mauser, S.: Learnflow Mining. In: Seehusen, S. (Hrsg.) ; Lucke, U. (Hrsg.) ; Fischer, S. (Hrsg.): *DeLFI 2008, LNI 132*, GI, 2008, S. 269–280 (Zitiert auf Seite 21.)

[24] Bergenthum, R. ; Desel, J. ; Juhás, G. ; Lorenz, R.: Can I Execute My Scenario in Your Net? VipTool Tells You! In: Donatelli, S. (Hrsg.) ; Thiagarajan, P. S. (Hrsg.): *Proc. of ICATPN 2006, LNCS 4024*, Springer, 2006, S. 381–390 (Zitiert auf Seiten 223, 225, und 229.)

[25] Bergenthum, R. ; Desel, J. ; Kölbl, C. ; Mauser, S.: Experimental Results on Process Mining Based on Regions of Languages. In: *Proc. of Workshop CHINA, Petri Nets 2008, X'ian, China*, 2008 (Zitiert auf Seiten 69, 73, und 74.)

[26] Bergenthum, R. ; Desel, J. ; Lorenz, R. ; Mauser, S.: Process Mining Based on Regions of Languages. In: Alonso, G. (Hrsg.) ; Dadam, P. (Hrsg.) ; Rosemann, M. (Hrsg.): *Proc. of BPM 2007, LNCS 4714*, Springer, 2007, S. 375–383 (Zitiert auf Seite 21.)

[27] Bergenthum, R. ; Desel, J. ; Lorenz, R. ; Mauser, S.: Synthesis of Petri Nets from Finite Partial Languages. In: *Fundam. Inform.* 88 (2008), Nr. 4, S. 437–468 (Zitiert auf Seite 21.)

[28] Bergenthum, R. ; Desel, J. ; Lorenz, R. ; Mauser, S.: Synthesis of Petri Nets from Infinite Partial Languages. In: Billington, J. (Hrsg.) ; Duan, Z. (Hrsg.) ; Koutny, M. (Hrsg.): *Proc. of ACSD 2008*, IEEE Computer Society, 2008, S. 170–179 (Zitiert auf Seite 21.)

[29] Bergenthum, R. ; Desel, J. ; Lorenz, R. ; Mauser, S.: Synthesis of Petri Nets from Scenarios with VipTool. In: Hee, K. M. (Hrsg.) ; Valk, R. (Hrsg.): *Proc. of Petri Nets 2008, LNCS 5062*, Springer, 2008, S. 388–398 (Zitiert auf Seite 21.)

[30] Bergenthum, R. ; Desel, J. ; Mauser, S.: Comparison of Different Algorithms to Synthesize a Petri Net from a Partial Language. In: *Transactions on Petri Nets and Other Models of Concurrency* 3, LNCS 5800 (2009), S. 216–243 (Zitiert auf Seite 21.)

[31] Bergenthum, R. ; Desel, J. ; Mauser, S. ; Lorenz, R.: Construction of Process Models from Example Runs. In: *Transactions on Petri Nets and Other Models of Concurrency* 2, LNCS 5460 (2009), S. 243–259 (Zitiert auf Seiten 20 und 21.)

[32] Bergenthum, R. ; Desel, J. ; Mauser, S. ; Lorenz, R.: Synthesis of Petri Nets from Term Based Representations of Infinite Partial Languages. In: *Fundam. Inform.* 95 (2009), Nr. 1, S. 187–217 (Zitiert auf Seiten 21, 264, und 265.)

[33] Bergenthum, R. ; Mauser, S.: Synthesis of Petri Nets from Infinite Partial Languages with VipTool. In: *Proc. of Workshop AWPN 2008, Rostock*, 2008 (Zitiert auf Seiten 230, 232, 264, und 266.)

[34] Bergenthum, R. ; Mauser, S. ; Lorenz, R. ; Juhás, G.: Unfolding Semantics of Petri Nets Based on Token Flows. In: *Fundam. Inform.* 94 (2009), Nr. 3-4, S. 331–360 (Zitiert auf Seiten 155, 214, 216, 229, und 276.)

[35] BERKELAAR, M.: *Lp solve reference guide*. – http://lpsolve.sourceforge.net/5.5/ (Zitiert auf Seite 230.)

[36] BERNARDINELLO, L.: Synthesis of Net Systems. In: MARSAN, M. A. (Hrsg.): *Proc. of Application and Theory of Petri Nets 1993, LNCS 691*, Springer, 1993, S. 89–105 (Zitiert auf Seite 11.)

[37] BERNARDINELLO, L. ; MICHELIS, G.D. ; PETRUNI, K. ; VIGNA, S.: On The Synchronic Structure of Transition Systems. In: DESEL, J. (Hrsg.): *Structures in Concurrency Theory*, Springer, 1996, S. 11–31 (Zitiert auf Seite 11.)

[38] BERNHARD, J. ; JODIN, D. ; HÖMBERG, K. ; KUHNT, S. ; SCHÜRMANN, C. ; WENZEL, S.: Vorgehensmodell zur Informationsgewinnung – Prozessschritte und Methodennutzung / Sonderforschungsbereich 559, Modellierung großer Netze in der Logistik, Universität Dortmund. 2007 (Technical Report 06008). – Forschungsbericht (Zitiert auf Seite 51.)

[39] BERTHELOT, G.: Transformations and Decompositions of Nets. In: BRAUER, W. (Hrsg.) ; W.REISIG (Hrsg.) ; ROZENBERG, G. (Hrsg.): *Proc. of an Advanced Course on Petri Nets: Central Models and Their Properties, LNCS 254*, Springer, 1987, S. 359–376 (Zitiert auf Seiten 93 und 192.)

[40] BEST, E. ; DEVILLERS, R.R.: Sequential and Concurrent Behaviour in Petri Net Theory. In: *Theoretical Computer Science* 55 (1987), Nr. 1, S. 87–136 (Zitiert auf Seiten 98 und 104.)

[41] BILLINGTON, J.: Protocol Specification Using P-Graphs, a Technique Based on Coloured Petri Nets. In: REISIG, W. (Hrsg.) ; ROZENBERG, G. (Hrsg.): *Lectures on Petri Nets II: Applications, LNCS 1492*, Springer, 1998, S. 293–330 (Zitiert auf Seite 93.)

[42] BORGWARDT, K.-H.: *The Simplex Algorithm: A Probabilistic Analysis, Algorithms and Combinatorics, Volume 1*. Springer, 1987 (Zitiert auf Seite 118.)

[43] BORNER, N. ; B.PAECH ; J.RÜCKERT: Vom Modellverstehen zum Modellerstellen. In: *Workshop Modellierung in Lehre und Weiterbildung, Modellierung 2006, Technischer Bericht ifi-2006-03*, Institut für Informatik, Universität Zürich, 2006, S. 7–15 (Zitiert auf Seite 31.)

[44] BOTE LORENZO, M.L. ; HERNÁNDEZ LEO, D. ; DIMITRIADIS, Y. ; ASENSIO PÉREZ, J.I. ; GÓMEZ SÁNCHEZ, E. ; VEGA GORGOJO, G. ; VAQUERO GONZÁLEZ, L.M.: Towards Reusability and Tailorability in Collaborative Learning Systems Using IMS-LD and Grid Services. In: *Advanced Technology for Learning* 1 (2004), Nr. 3, S. 129–138 (Zitiert auf Seite 76.)

[45] BRINGHURST, R.: *The Elements of Typographic Style*. Hartley & Marks, 2002 (Zitiert auf Seite 365.)

[46] BRONDSTED, A.: *Introduction to Convex Polytopes*. Springer, 1981 (Zitiert auf Seiten 113 und 114.)

[47] BUSI, N. ; PINNA, G.M.: Synthesis of Nets with Inhibitor Arcs. In: MAZURKIEWICZ, A.W. (Hrsg.) ; WINKOWSKI, J. (Hrsg.): *Proc. of CONCUR '97, LNCS 1243*, Springer, 1997, S. 151–165 (Zitiert auf Seiten 12, 279, und 290.)

[48] Cabasino, M. P. ; Giua, A. ; Seatzu, C.: Identification of Petri Nets from Knowledge of Their Language. In: *Discrete Event Dynamic Systems* 17 (2007), Nr. 4, S. 447–474 (Zitiert auf Seiten 13, 321, 322, 331, und 336.)

[49] Caillaud, B.: *Synet*. – http://www.irisa.fr/s4/tools/synet/ (Zitiert auf Seiten 13, 14, und 222.)

[50] Caillaud, B. ; Darondeau, P. ; Hélouët, L. ; Lesventes, G.: HMSCs as Partial Specifications ... with PNs as Completions. In: Cassez, F. (Hrsg.) ; Jard, C. (Hrsg.) ; Rozoy, B. (Hrsg.) ; Ryan, M. D. (Hrsg.): *Proc. of MOVEP 2000, LNCS 2067*, Springer, 2001, S. 125–152 (Zitiert auf Seite 12.)

[51] Carmona, J. ; Cortadella, J. ; Kishinevsky, M.: A Region-Based Algorithm for Discovering Petri Nets from Event Logs. In: Dumas, M. (Hrsg.) ; Reichert, M. (Hrsg.) ; Shan, M.-C. (Hrsg.): *Proc. of BPM 2008, LNCS 5240*, Springer, 2008, S. 358–373 (Zitiert auf Seiten 13, 69, 74, und 339.)

[52] Carmona, J. ; Cortadella, J. ; Kishinevsky, M.: Genet: A Tool for the Synthesis and Mining of Petri Nets. In: *Proc. of ACSD 2009*, IEEE Computer Society, 2009, S. 181–185 (Zitiert auf Seiten 14 und 222.)

[53] Carmona, J. ; Cortadella, J. ; Kishinevsky, M.: New Region-Based Algorithms for Deriving Bounded Petri Nets. In: *IEEE Trans. Computers* 59 (2010), Nr. 3, S. 371–384 (Zitiert auf Seiten 12, 13, 14, 69, 74, 175, und 339.)

[54] Carmona, J. ; Cortadella, J. ; Kishinevsky, M. ; Kondratyev, A. ; Lavagno, L. ; Yakovlev, A.: A Symbolic Algorithm for the Synthesis of Bounded Petri Nets. In: Hee, K. M. (Hrsg.) ; Valk, R. (Hrsg.): *Proc. of Petri Nets 2008, LNCS 5062*, Springer, 2008, S. 92–111 (Zitiert auf Seiten 12 und 14.)

[55] Castela, N. ; Tribolet, J. M. ; Guerra, A. ; Lopes, E. R.: Survey, Analysis and Validation of Information for Business Process Modeling. In: *Proc. of ICEIS 2002, Ciudad Real, Spanien*, 2002, S. 803–806 (Zitiert auf Seite 50.)

[56] Chazelle, B.: An Optimal Convex Hull Algorithm in Any Fixed Dimension. In: *Discrete & Computational Geometry* 10 (1993), S. 377–401 (Zitiert auf Seite 114.)

[57] Colom, J. M. ; Silva, M.: Improving the Linearly Based Characterization of P/T Nets. In: Rozenberg, G. (Hrsg.): *Advances in Petri Nets 1990, LNCS 483*, Springer, 1990, S. 113–145 (Zitiert auf Seiten 93 und 192.)

[58] Cortadella, J. ; Kishinevsky, M. ; Kondratyev, A. ; Lavagno, L. ; Yakovlev, A.: Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers. In: *IEICE Trans. of Informations and Systems* E80-D (1997), Nr. 3, S. 315–325 (Zitiert auf Seiten 11, 14, und 221.)

[59] Cortadella, J. ; Kishinevsky, M. ; Kondratyev, A. ; Lavagno, L. ; Yakovlev, A.: Hardware and Petri Nets: Application to Asynchronous Circuit Design. In: Nielsen, M. (Hrsg.) ; Simpson,

D. (Hrsg.): *Proc. of ICATPN 2000, LNCS 1825*, Springer, 2000, S. 1–15 (Zitiert auf Seiten 11 und 13.)

[60] CORTADELLA, J. ; KISHINEVSKY, M. ; LAVAGNO, L. ; YAKOVLEV, A.: Deriving Petri Nets from Finite Transition Systems. In: *IEEE Trans. Computers* 47 (1998), Nr. 8, S. 859–882 (Zitiert auf Seiten 11, 12, 14, und 175.)

[61] DANTZIG, G. B. ; THAPA, M. N.: *Linear programming 1: Introduction*. Springer, 1997 (Zitiert auf Seite 117.)

[62] DANTZIG, G. B. ; THAPA, M. N.: *Linear programming 2: Theory and Extensions*. Springer, 2003 (Zitiert auf Seiten 117, 118, 119, 120, und 121.)

[63] DARONDEAU, P.: Deriving Unbounded Petri Nets from Formal Languages. In: SANGIORGI, D. (Hrsg.) ; SIMONE, R. (Hrsg.): *Proc. of CONCUR 1998, LNCS 1466*, Springer, 1998, S. 533–548 (Zitiert auf Seiten 12, 13, 69, 74, 136, 142, 144, 177, 189, 193, 253, und 318.)

[64] DARONDEAU, P.: Region Based Synthesis of P/T-Nets and Its Potential Applications. In: NIELSEN, M. (Hrsg.) ; SIMPSON, D. (Hrsg.): *Proc. of ICATPN 2000, LNCS 1825*, Springer, 2000, S. 16–23 (Zitiert auf Seiten 13, 318, 319, 336, und 337.)

[65] DARONDEAU, P.: Unbounded Petri Net Synthesis. In: DESEL, J. (Hrsg.) ; REISIG, W. (Hrsg.) ; ROZENBERG, G. (Hrsg.): *Lectures on Concurrency and Petri Nets, LNCS 3098*, Springer, 2003, S. 413–438 (Zitiert auf Seiten 12, 13, 136, 142, 144, 173, 176, 177, 189, 193, 202, 253, 323, und 327.)

[66] DARONDEAU, P.: Distributed Implementations of Ramadge-Wonham Supervisory Control with Petri Nets. In: *Proc. of CDC-ECC 2005*, IEEE Computer Society, 2005, S. 2107–2112 (Zitiert auf Seiten 13, 319, 336, und 337.)

[67] DARONDEAU, P.: Synthesis and Control of Asynchronous and Distributed Systems. In: *Proc. of ACSD 2007*, IEEE Computer Society, 2007, S. 13–22 (Zitiert auf Seiten 13, 319, 333, 337, und 338.)

[68] DARONDEAU, P. ; KOUTNY, M. ; PIETKIEWICZ-KOUTNY, M. ; YAKOVLEV, A.: Synthesis of Nets with Step Firing Policies. In: *Fundam. Inform.* 94 (2009), Nr. 3-4, S. 275–303 (Zitiert auf Seiten 11, 290, 298, und 300.)

[69] DERWERF, J. M. E. M. ; DONGEN, B. F. ; HURKENS, C. A. J. ; SEREBRENIK, A.: Process Discovery using Integer Linear Programming. In: *Fundam. Inform.* 94 (2009), Nr. 3-4, S. 387–412 (Zitiert auf Seiten 13, 14, 69, 74, 331, 336, und 339.)

[70] DESEL, J.: Formaler Umgang mit semi-formalen Ablaufbeschreibungen. In: LAUSEN, G. (Hrsg.) ; OBERWEIS, A. (Hrsg.) ; SCHLAGETER, G. (Hrsg.): *Angewandte Informatik und Formale Beschreibungsverfahren, Teubner-Texte zur Informatik 29*, Teubner, 1999, S. 45–90 (Zitiert auf Seite 37.)

[71] DESEL, J.: Model Validation - A Theoretical Issue? In: ESPARZA, J. (Hrsg.) ; LAKOS, C. (Hrsg.): *Proc. of ICATPN 2002, LNCS 2360*, Springer, 2002, S. 23–43 (Zitiert auf Seite 223.)

[72] Desel, J.: From Human Knowledge to Process Models. In: Kaschek, R. (Hrsg.) ; Kop, C. (Hrsg.) ; Steinberger, C. (Hrsg.) ; Fliedl, G. (Hrsg.): *Proc. of UNISCON 2008, LNBIP 5*, Springer, 2008, S. 84–95 (Zitiert auf Seiten xi, 13, 23, und 50.)

[73] Desel, J.: Modellieren lernen über Formalisieren von Ablaufbeschreibungen. In: Desel, J. (Hrsg.) ; Glinz, M. (Hrsg.): *Technischer Bericht ifi-2008.04*, Institut für Informatik, Universität Zürich, 2008, S. 37–46 (Zitiert auf Seiten 13, 30, und 36.)

[74] Desel, J. ; Juhás, G.: What Is a Petri Net?. In: Ehrig, H. (Hrsg.) ; Juhás, G. (Hrsg.) ; Padberg, J. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Unifying Petri Nets, LNCS 2128*, Springer, 2001, S. 1–25 (Zitiert auf Seite 89.)

[75] Desel, J. ; Juhás, G. ; Lorenz, R.: Petri Nets over Partial Algebra. In: Ehrig, H. (Hrsg.) ; Juhás, G. (Hrsg.) ; Padberg, J. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Unifying Petri Nets, LNCS 2128*, Springer, 2001, S. 126–172 (Zitiert auf Seite 291.)

[76] Desel, J. ; Juhás, G. ; Lorenz, R. ; Neumair, C.: Modelling and Validation with VipTool. In: Aalst, W.M.P.d. (Hrsg.) ; Hofstede, A.H.M.t. (Hrsg.) ; Weske, M. (Hrsg.): *Proc. of BPM 2003, LNCS 2678*, Springer, 2003, S. 380–389 (Zitiert auf Seiten 50, 222, 223, 225, 228, und 229.)

[77] Desel, J. ; Neumair, C.: Entwicklung und Bewertung einer Unterrichtssequenz zur ablauforientierten Sichtweise von Algorithmen. In: Brinda, T. (Hrsg.) ; Fothe, M. (Hrsg.) ; Hubwieser, P. (Hrsg.) ; Schlüter, K. (Hrsg.): *Tagungsband Didaktik der Informatik: Aktuelle Forschungsergebnisse, LNI 135*, GI, 2008, S. 151–152 (Zitiert auf Seiten 30 und 36.)

[78] Desel, J. ; Reisig, W.: The Synthesis Problem of Petri Nets. In: *Acta Informatica* 33 (1996), Nr. 4, S. 297–315 (Zitiert auf Seite 11.)

[79] Desel, J. ; W.Reisig: Place/Transition Petri Nets. In: Reisig, W. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, LNCS 1491*, Springer, 1998, S. 123–174 (Zitiert auf Seite 89.)

[80] Dieckert, V. ; Metivier, Y: Partial Communication and Traces. In: Rozenberg, G. (Hrsg.) ; Salomaa, A. (Hrsg.): *Handbook of Formal Languages, Vol. 3*, Springer, 1997 (Zitiert auf Seite 98.)

[81] Donatelli, S. ; Franceschinis, G.: Modelling and Analysis of Distributed Software Using GSPNs. In: Reisig, W. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Lectures on Petri Nets II: Applications, LNCS 1492*, Springer, 1998, S. 438–476 (Zitiert auf Seite 93.)

[82] Dongen, B. F. ; Aalst, W. M. P. d.: Multi-Phase Process Mining: Building Instance Graphs. In: Atzeni, P. (Hrsg.) ; Chu, W. W. (Hrsg.) ; Lu, H. (Hrsg.) ; Zhou, S. (Hrsg.) ; Ling, T. W. (Hrsg.): *Proc. of Conceptual Modeling - ER 2004, LNCS 3288*, Springer, 2004, S. 362–376 (Zitiert auf Seite 71.)

[83] Dongen, B.F. ; Aalst, W.M.P.d.: Multi-Phase Process Mining: Aggregating Instance Graphs into EPC's and Petri Nets. In: *Proc. of 2nd Workshop on Applications of Petri Nets to Coordination,*

Workflow and Business Process Management, Petri Nets 2005, Miami, 2005, S. 35–58 (Zitiert auf Seiten 14, 50, 56, und 70.)

[84] Dongen, B.F. ; Busi, N. ; Pinna, G.M. ; Aalst, W.M.P.d.: An Iterative Algorithm for Applying the Theory of Regions in Process Mining. / Department of Technology Management, Eindhoven University of Technology. 2007 (Beta rapport 195). – Forschungsbericht (Zitiert auf Seiten 13, 14, 68, 176, und 339.)

[85] Droste, M. ; Shortt, R. M.: Petri nets and automata with concurrency relations - an adjunction. In: Droste, M. (Hrsg.) ; Gurevich, Y. (Hrsg.): *Semantics of programming languages and model theory*, Gordon and Breach Science Publishers, Inc., 1993, S. 69 – 87 (Zitiert auf Seite 11.)

[86] Droste, M. ; Shortt, R. M.: From Petri Nets to Automata with Concurrency. In: *Applied Categorical Structures* 10 (2002), Nr. 2, S. 173–191 (Zitiert auf Seite 291.)

[87] Dumas, M. ; Aalst, W.M.P.d. ; Hofstede, A.H.t.: *Process-Aware Information Systems: Bridging People and Software Through Process Technology.* Wiley, 2005 (Zitiert auf Seiten 46, 50, und 75.)

[88] Eckhardt, U.: Redundante Ungleichungen bei linearen Ungleichungssystemen. In: *Mathematical Methods of Operations Research* 15 (1971), Nr. 1, S. 279–286 (Zitiert auf Seite 121.)

[89] Ehrenfeucht, A. ; Rozenberg, G.: Partial (Set) 2-Structures. Part I: Basic Notions and the Representation Problem. In: *Acta Informatica* 27 (1989), Nr. 4, S. 315–342 (Zitiert auf Seiten 11, 14, und 128.)

[90] Ehrenfeucht, A. ; Rozenberg, G.: Partial (Set) 2-Structures. Part II: State Spaces of Concurrent Systems. In: *Acta Informatica* 27 (1989), Nr. 4, S. 343–368 (Zitiert auf Seiten 11, 14, 128, und 172.)

[91] Engelfriet, J.: Branching Processes of Petri Nets. In: *Acta Informatica* 28 (1991), Nr. 6, S. 575–591 (Zitiert auf Seiten 213 und 215.)

[92] Esparza, J. ; Römer, S. ; Vogler, W.: An Improvement of McMillan's Unfolding Algorithm. In: *Formal Methods in System Design* 20 (2002), Nr. 3, S. 285–310 (Zitiert auf Seiten 213 und 215.)

[93] Esparza, J. ; Schröter, C.: Net Reductions for LTL Model-Checking. In: Margaria, T. (Hrsg.) ; Melham, T. F. (Hrsg.): *Proc. of CHARME 2001, LNCS 2144*, Springer, 2001, S. 310–324 (Zitiert auf Seite 192.)

[94] Fanchon, J. ; Morin, R.: Pomset Languages of Finite Step Transition Systems. In: Franceschinis, G. (Hrsg.) ; Wolf, K. (Hrsg.): *Proc. of Petri Nets 2009, LNCS 5606*, Springer, 2009, S. 83–102 (Zitiert auf Seiten 98, 100, und 277.)

[95] Faulk, S.R.: Software Requirements: A Tutorial. In: Thayer, R. (Hrsg.) ; Dorfman, M. (Hrsg.): *Software Engineering*, IEEE Computer Society, 1995, S. 82–101 (Zitiert auf Seite 37.)

[96] Fliedl, G. ; Kop, C. ; Mayr, H. C.: From Textual Scenarios to a Conceptual Schema. In: *Data Knowl. Eng.* 55 (2005), Nr. 1, S. 20–37 (Zitiert auf Seiten 38 und 45.)

[97] Ford, L.R. ; Fulkerson, D.R.: Maximal Flow Through a Network. In: *Canadian Journal of Mathematics* 8 (1956), S. 399–404 (Zitiert auf Seiten 219, 221, und 229.)

[98] Frederiks, P. J. M. ; Weide, T. P. d.: Information modeling: The process and the required competencies of its participants. In: *Data Knowl. Eng.* 58 (2006), Nr. 1, S. 4–20 (Zitiert auf Seite 45.)

[99] Freytag, T.: *Softwarevalidierung durch Auswertung von Petrinetz-Abläufen*, Universität Karlsruhe, Diss., 2001 (Zitiert auf Seiten 222 und 225.)

[100] Fukuda, K. ; Prodon, A.: Double Description Method Revisited. In: Deza, M. (Hrsg.) ; Euler, R. (Hrsg.) ; Manoussakis, Y. (Hrsg.): *Combinatorics and Computer Science 1995, LNCS 1120*, Springer, 1996, S. 91–111 (Zitiert auf Seiten 113, 114, 115, und 116.)

[101] Gaifman, H. ; Pratt, V. R.: Partial Order Models of Concurrency and the Computation of Functions. In: *Proc. of LICS 1987*, IEEE Computer Society, 1987, S. 72–85 (Zitiert auf Seite 109.)

[102] Ghaffari, A. ; Rezg, N. ; Xie, X.: Maximally Permissive and Non-Blocking Control of Petri Nets using Theory of Regions. In: *Proc. of ICRA 2002*, IEEE Computer Society, 2002, S. 1895–1900 (Zitiert auf Seite 13.)

[103] Glabbeek, R. J. ; Plotkin, G. D.: Event Structures for Resolvable Conflict. In: Fiala, J. (Hrsg.) ; Koubek, V. (Hrsg.) ; Kratochvíl, J. (Hrsg.): *Proc. of MFCS 2004, LNCS 3153*, Springer, 2004, S. 550–561 (Zitiert auf Seite 12.)

[104] Glinz, M.: An Integrated Formal Model of Scenarios Based on Statecharts. In: Schäfer, W. (Hrsg.) ; Botella, P. (Hrsg.): *Proc. of ESEC 1995, LNCS 989*, Springer, 1995, S. 254–271 (Zitiert auf Seiten 37, 38, und 265.)

[105] Glinz, M.: Improving the Quality of Requirements with Scenarios. In: *Proc. of the Second World Congress on Software Quality, Yokohama, Japan*, 2000, S. 55–60 (Zitiert auf Seiten 13, 37, und 38.)

[106] Goltz, U. ; Reisig, W.: The Non-Sequential Behaviour of Petri Nets. In: *Information and Control* 57 (1983), Nr. 2/3, S. 125–147 (Zitiert auf Seiten 104 und 105.)

[107] Goltz, U. ; Reisig, W.: Processes of Place/Transition-Nets. In: Díaz, J. (Hrsg.): *Proc. of ICALP 1983, LNCS 154*, Springer, 1983, S. 264–277 (Zitiert auf Seiten 104 und 105.)

[108] Grabowski, J.: On Partial Languages. In: *Fundamenta Informaticae* 4 (1981), Nr. 2, S. 428–498 (Zitiert auf Seiten 12 und 99.)

[109] Hack, M-H-T.: *Decidability Questions for Petri Nets*, MIT, Diss., 1976 (Zitiert auf Seite 93.)

[110] Harel, D. ; Marelly, R.: *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine.* Springer, 2003 (Zitiert auf Seiten 13, 36, 37, 38, und 98.)

[111] HARRER, A. ; HOPPE, U.: Visual Modelling of Collaborative Learning Processes – Uses, Desired Properties, and Approaches. In: BOTTURI, L. (Hrsg.) ; STUBBS, S.T. (Hrsg.): *Handbook of Visual Languages for Instructional Design: Theory and Practices*. Information Science Reference, 2008, S. 281–298 (Zitiert auf Seite 75.)

[112] HARRER, A. ; MALZAHN, N.: Bridging the Gap - Towards a Graphical Modelling Language for Learning Designs and Collaboration Scripts of Various Granularities. In: *Proc. of ICALT 2006*, IEEE Computer Society, 2006, S. 296–300 (Zitiert auf Seite 76.)

[113] HARRER, A. ; MALZAHN, N. ; WICHMANN, A.: The Remote Control Approach - An Architecture for Adaptive Scripting across Collaborative Learning Environments. In: *Journal of Universal Computer Science* 14 (2008), Nr. 1, S. 148–173 (Zitiert auf Seiten 75 und 77.)

[114] HEE, K. M. ; SIDOROVA, N. ; SOMERS, L. J. ; VOORHOEVE, M.: Consistency in Model Integration. In: *Data Knowl. Eng.* 56 (2006), Nr. 1, S. 4–22 (Zitiert auf Seite 50.)

[115] HEINER, M. ; KOCH, I. ; WILL, J.: Model Validation of Biological Pathways Using Petri Nets - Demonstrated for Apoptosis. In: *Journal BioSystems* 75 (2004), Nr. 1-3, S. 15–28 (Zitiert auf Seite 249.)

[116] HERNANDEZ-LEO, D.: *A pattern-based design process for the creation of CSCL macro-scripts computationally represented with IMS LD*, University of Valladolid, Diss., 2007 (Zitiert auf Seite 75.)

[117] HIRAISHI, K.: Some Complexity Results on Transition Systems and Elementary Net Systems. In: *Theor. Comput. Science* 135 (1994), Nr. 2, S. 361–376 (Zitiert auf Seite 11.)

[118] HOLTEN, R. ; STRIEMER, R. ; WESKE, M.: Vergleich von Ansätzen zur Entwicklung von Workflow-Anwendungen. In: OBERWEIS, A. (Hrsg.) ; SNEED, H. (Hrsg.): *Proc. of Software Management '97*, Teubner, 1997, S. 258–274 (Zitiert auf Seite 50.)

[119] HÖMBERG, K. ; JODIN, D. ; LEPPIN, M.: Methoden der Informations– und Datenerhebung / Sonderforschungsbereich 559, Modellierung großer Netze in der Logistik, Universität Dortmund. 2004 (Technical Report 04002). – Forschungsbericht (Zitiert auf Seiten 51, 53, und 54.)

[120] HOOGERS, P.W. ; KLEIJN, H.C.M. ; THIAGARAJAN, P.S.: A Trace Semantics for Petri Nets. In: *Information and Computation* 117 (1995), Nr. 1, S. 98–114 (Zitiert auf Seiten 12, 136, und 144.)

[121] HOOGERS, P.W. ; KLEIJN, H.C.M. ; THIAGARAJAN, P.S.: An Event Structure Semantics for General Petri Nets. In: *Theoretical Computer Science* 153 (1996), Nr. 1&2, S. 129–170 (Zitiert auf Seite 12.)

[122] IMS GLOBAL CONSORTIUM: *IMS Learning Design XML Binding*. 2003. – 1.0 Specification (Zitiert auf Seite 75.)

[123] JACOBSON, I.: *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992 (Zitiert auf Seiten 13, 37, und 98.)

[124] Janicki, R. ; Koutny, M.: Invariants and Paradigms of Concurrency Theory. In: Aarts, E. H. L. (Hrsg.) ; Leeuwen, J. (Hrsg.) ; Rem, M. (Hrsg.): *Proc. of PARLE '91: Parallel Architectures and Languages Europe, Volume II: Parallel Languages, LNCS 506*, Springer, 1991, S. 59–74 (Zitiert auf Seite 109.)

[125] Janicki, R. ; Koutny, M.: Order Structures and Generalisations of Szpilrajn's Theorem. In: Shyamasundar, R. K. (Hrsg.): *Proc. of FSTTCS 1993, LNCS 761*, Springer, 1993, S. 348–357 (Zitiert auf Seite 109.)

[126] Janicki, R. ; Koutny, M.: Structure of Concurrency. In: *Theor. Comput. Sci.* 112 (1993), Nr. 1, S. 5–52 (Zitiert auf Seiten 98 und 108.)

[127] Janicki, R. ; Koutny, M.: Semantics of Inhibitor Nets. In: *Inf. Comput.* 123 (1995), Nr. 1, S. 1–16 (Zitiert auf Seiten 93, 108, 109, und 111.)

[128] J.Desel ; L.Petrucci: Aggregating views for Petri net model construction. In: *Proc. of Workshop PNDS08, Petri Nets 2008, Xi'an, China*, 2008, S. 17–31 (Zitiert auf Seiten 14 und 50.)

[129] Jensen, K.: *Monographs in Theoretical Computer Science*. Bd. 1-3: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Springer, 1992, 1994, 1997 (Zitiert auf Seiten 84 und 249.)

[130] Josephs, M. B. ; Furey, D. P.: A Programming Approach to the Design of Asynchronous Logic Blocks. In: Cortadella, J. (Hrsg.) ; Yakovlev, A. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Concurrency and Hardware Design 2002, LNCS 2549*, Springer, 2002, S. 34–60 (Zitiert auf Seite 13.)

[131] Juhás, G.: *Are these Events Independend? It Depends!* Habilitation, 2005 (Zitiert auf Seiten 1, 16, 89, 98, 104, 105, 154, 289, 291, 305, und 306.)

[132] Juhás, G. ; Lorenz, R.: Towards Synthesis of Petri Nets from Scenarios. In: Donatelli, S. (Hrsg.) ; Thiagarajan, P. S. (Hrsg.): *Proc. of ICATPN 2006, LNCS 4024*, Springer, 2006, S. 302–321 (Zitiert auf Seiten 12, 15, 16, und 154.)

[133] Juhás, G. ; Lorenz, R. ; Desel, J.: Can I Execute My Scenario in Your Net? In: Ciardo, G. (Hrsg.) ; Darondeau, P. (Hrsg.): *Proc. of ICATPN 2005, LNCS 3536*, Springer, 2005, S. 289–308 (Zitiert auf Seiten 16, 154, 223, und 229.)

[134] Juhás, G. ; Lorenz, R. ; Desel, J.: Unifying Petri Net Semantics with Token Flows. In: Franceschinis, G. (Hrsg.) ; Wolf, K. (Hrsg.): *Proc. of Petri Nets 2009, LNCS 5606*, Springer, 2009, S. 2–21 (Zitiert auf Seiten 154, 155, und 287.)

[135] Juhás, G. ; Lorenz, R. ; Mauser, S.: Complete Process Semantics for Inhibitor Nets. In: Kleijn, J. (Hrsg.) ; Yakovlev, A. (Hrsg.): *Proc. of ICATPN 2007, LNCS 4546*, Springer, 2007, S. 184–203 (Zitiert auf Seite 111.)

[136] Juhás, G ; Lorenz, R ; Mauser, S: Complete Process Semantics of Petri Nets. In: *Fundam. Inform.* 87 (2008), Nr. 3-4, S. 331–365 (Zitiert auf Seite 111.)

[137] KARMARKAR, N.: A new polynomial-time algorithm for linear programming. In: *Combinatorica* 4 (1984), S. 373–395 (Zitiert auf Seite 118.)

[138] KAUFMANN, M. (Hrsg.) ; WAGNER, D. (Hrsg.): *Drawing Graphs - Methods and Models, LNCS 2025*. Springer, 2001 (Zitiert auf Seite 228.)

[139] KHOMENKO, V. ; KOUTNY, M.: Towards an Efficient Algorithm for Unfolding Petri Nets. In: LARSEN, K.G. (Hrsg.) ; NIELSEN, M. (Hrsg.): *Proc. of CONCUR 2001, LNCS 2154*, Springer, 2001, S. 366–380 (Zitiert auf Seiten 213 und 215.)

[140] KHOMENKO, V. ; KOUTNY, M.: Branching Processes of High-Level Petri Nets. In: GARAVEL, H. (Hrsg.) ; HATCLIFF, J. (Hrsg.): *Proc. of TACAS 2003, LNCS 2619*, Springer, 2003, S. 458–472 (Zitiert auf Seiten 213 und 215.)

[141] KHOMENKO, V. ; KOUTNY, M. ; VOGLER, W.: Canonical prefixes of Petri net unfoldings. In: *Acta Inf.* 40 (2003), Nr. 2, S. 95–118 (Zitiert auf Seiten 213 und 215.)

[142] KIEHN, A.: On the Interrelation Between Synchronized and Non-Synchronized Behaviour of Petri Nets. In: *Elektronische Informationsverarbeitung und Kybernetik* 24 (1988), Nr. 1/2, S. 3–18 (Zitiert auf Seiten 104, 107, und 305.)

[143] KLEIJN, H. C. M. ; KOUTNY, M.: Process Semantics of P/T-Nets with Inhibitor Arcs. In: NIELSEN, M. (Hrsg.) ; SIMPSON, D. (Hrsg.): *Proc. of ICATPN 2000, LNCS 1825*, Springer, 2000, S. 261–281 (Zitiert auf Seite 111.)

[144] KLEIJN, H. C. M. ; KOUTNY, M.: Process Semantics of General Inhibitor Nets. In: *Inf. Comput.* 190 (2004), Nr. 1, S. 18–69 (Zitiert auf Seiten 93, 108, und 111.)

[145] KOPER, R. (Hrsg.) ; TATTERSALL, C. (Hrsg.): *Learning Design - Modelling and implementing network-based education & training*. Springer, 2005 (Zitiert auf Seite 75.)

[146] KOUTNY, M. ; PIETKIEWICZ-KOUTNY, M.: Synthesis of Elementary Net Systems with Context Arcs and Localities. In: *Fundam. Inform.* 88 (2008), Nr. 3, S. 307–328 (Zitiert auf Seiten 12, 173, 290, 298, und 300.)

[147] KUSKE, D. ; MORIN, R.: Pomsets for Local Trace Languages - Recognizability, Logic & Petri Nets. In: PALAMIDESSI, C. (Hrsg.): *Proc. of CONCUR 2000, LNCS 1877*, Springer, 2000, S. 426–441 (Zitiert auf Seite 12.)

[148] L. WOLSEY, G. N.: *Integer and Combinatorial Optimization*. Wiley, 1988 (Zitiert auf Seiten 116, 117, 119, 120, und 121.)

[149] LAND, A. H. ; DOIG, A. G.: An automatic method of solving discrete programming problems. In: *Econometrica* 28 (1960), S. 497–520 (Zitiert auf Seite 120.)

[150] Liang, H. ; Dingel, J. ; Diskin, Z.: A Comparative Survey of Scenario-Based to State-Based Model Synthesis Approaches. In: Whittle, J. (Hrsg.) ; Geiger, L. (Hrsg.) ; Meisinger, M. (Hrsg.): *Proc. of SCESM '06*, ACM, 2006, S. 5–12 (Zitiert auf Seiten 13, 38, und 39.)

[151] Lodaya, K. ; Weil, P.: Series-Parallel Posets: Algebra, Automata and Languages. In: Morvan, M. (Hrsg.) ; Meinel, C. (Hrsg.) ; Krob, D. (Hrsg.): *Proc. of STACS 98, LNCS 1373*, Springer, 1998, S. 555–565 (Zitiert auf Seiten 12 und 253.)

[152] Lodaya, K. ; Weil, P.: Series-Parallel Languages and the Bounded-Width Property. In: *Theor. Comput. Sci.* 237 (2000), Nr. 1-2, S. 347–380 (Zitiert auf Seiten 12 und 253.)

[153] Lorenz, R.: *Szenario-basierte Verifikation und Synthese von Petrinetzen: Theorie und Anwendungen*. Habilitation, 2006 (Zitiert auf Seiten xiii, 12, 15, 16, 19, 20, 154, 155, 202, 203, 217, und 253.)

[154] Lorenz, R.: Towards Synthesis of Petri Nets from General Partial Languages. In: *Workshop AWPN 2008, Rostock*, 2008 (Zitiert auf Seite 276.)

[155] Lorenz, R. ; Bergenthum, R. ; Desel, J. ; Mauser, S.: Synthesis of Petri Nets from Finite Partial Languages. In: Basten, T. (Hrsg.) ; Juhás, G. (Hrsg.) ; Shukla, S. K. (Hrsg.): *Proc. of ACSD 2007*, IEEE Computer Society, 2007, S. 157–166 (Zitiert auf Seite 21.)

[156] Lorenz, R. ; Juhás, G. ; Bergenthum, R. ; Desel, J. ; Mauser, S.: Executability of scenarios in Petri nets. In: *Theor. Comput. Sci.* 410 (2009), Nr. 12-13, S. 1190–1216 (Zitiert auf Seiten 219, 220, 221, 223, 229, 309, und 311.)

[157] Lorenz, R. ; Juhás, G. ; Mauser, S.: Partial Order Semantics of Types of Nets. In: Nielsen, M. (Hrsg.) ; Kucera, A. (Hrsg.) ; Miltersen, P. B. (Hrsg.) ; Palamidessi, C. (Hrsg.) ; Tuma, P. (Hrsg.) ; Valencia, F. D. (Hrsg.): *Proc. of SOFSEM 2009, LNCS 5404*, Springer, 2009, S. 388–400 (Zitiert auf Seite 21.)

[158] Lorenz, R. ; Mauser, S. ; Bergenthum, R.: Theory of Regions for the Synthesis of Inhibitor Nets from Scenarios. In: Kleijn, J. (Hrsg.) ; Yakovlev, A. (Hrsg.): *Proc. of ICATPN 2007, LNCS 4546*, Springer, 2007, S. 342–361 (Zitiert auf Seite 21.)

[159] Lorenz, R. ; Mauser, S. ; Juhás, G.: How to synthesize nets from languages: a survey. In: Henderson, S. G. (Hrsg.) ; Biller, B. (Hrsg.) ; Hsieh, M.-H. (Hrsg.) ; Shortle, J. (Hrsg.) ; Tew, J. D. (Hrsg.) ; Barton, R. R. (Hrsg.): *Proc. of WSC 2007*, 2007, S. 637–647 (Zitiert auf Seiten 20 und 21.)

[160] Malkin, P. N.: *Computing Markov bases, Gröbner bases, and extreme rays*, Universite catholique de Louvain, Diss., 2007 (Zitiert auf Seiten 113, 114, 115, und 116.)

[161] Mans, R.S. ; Schoenenberg, M.H. ; Song, M. ; Aalst, W.M.P.v.d. ; Bakker, P.J.M.: Application of Process Mining in Healthcare – A Case Study in a Dutch Hospital. In: Fred, A.L.N. (Hrsg.) ; Filipe, J. (Hrsg.) ; Gamboa, H. (Hrsg.): *Selected Papers BIOSTEC 2008, CCIS 25*, Springer, 2008, S. 425–438 (Zitiert auf Seite 67.)

[162] MARTEL, C. ; VIGNOLLET, L. ; FERRARIS, C. ; DAVID, J.-P. ; LEJEUNE, A.: Modelling collaborative learning activities in e-learning platforms. In: *Proc. of ICALT 2006*, IEEE Computer Society, 2006, S. 707–709 (Zitiert auf Seite 75.)

[163] MATHEISS, T. H. ; RUBIN, D.S.: A survey and comparison of methods for finding all vertices of convex polyhedral sets. In: *Mathematics of operations research* 5 (1980), S. 167–185 (Zitiert auf Seite 114.)

[164] MAUSER, S. ; LORENZ, R.: Variants of the Language Based Synthesis Problem for Petri Nets. In: *Proc. of ACSD 2009*, IEEE Computer Society, 2009, S. 89–98 (Zitiert auf Seite 22.)

[165] MAYR, H. C. ; DITTRICH, K. R. ; LOCKEMANN, P. C.: Datenbankentwurf. In: LOCKEMANN, P. C. (Hrsg.) ; SCHMIDT, J. W. (Hrsg.): *Datenbankhandbuch*. Springer, 1987, S. 481–557 (Zitiert auf Seite 37.)

[166] MAYR, H. C. ; KOP, C.: A User Centered Approach to Requirements Modeling. In: *Tagungsband Modellierung 2002, LNI 12*, GI, 2002, S. 75–86 (Zitiert auf Seiten 45, 50, und 54.)

[167] MAYR, H. C. ; KOP, C. ; ESBERGER, D.: Business Process Modeling and Requirements Modeling. In: *Proc. of ICDS 2007*, IEEE Computer Society, 2007, S. 8–14 (Zitiert auf Seiten 45 und 50.)

[168] MAZURKIEWICZ, A.: Petri Nets Without Tokens. In: KLEIJN, J. (Hrsg.) ; YAKOVLEV, A. (Hrsg.): *Proc. of ICATPN 2007, LNCS 4546*, Springer, 2007, S. 20–23 (Zitiert auf Seite 291.)

[169] MCMILLAN, K. L.: Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits. In: BOCHMANN, G. (Hrsg.) ; PROBST, D. K. (Hrsg.): *Proc. of CAV '92, LNCS 663*, Springer, 1993, S. 164–177 (Zitiert auf Seiten 213 und 215.)

[170] MCMULLEN, P.: The maximum numbers of faces of a convex polytope. In: *Mathematika* 17 (1970), S. 179–184 (Zitiert auf Seite 114.)

[171] MENDLING, J. ; SIMON, C.: Business Process Design by View Integration. In: EDER, J. (Hrsg.) ; DUSTDAR, S. (Hrsg.): *Business Process Management Workshops, BPM 2006, LNCS 4103*, Springer, 2006, S. 55–64 (Zitiert auf Seite 50.)

[172] MOODY, D. L.: Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. In: GOTTLOB, G. (Hrsg.) ; BENCZÚR, A. A. (Hrsg.) ; DEMETROVICS, J. (Hrsg.): *Proc. of ADBIS 2004, LNCS 3255*, Springer, 2004, S. 129–143 (Zitiert auf Seite 45.)

[173] MUKUND, M.: Petri Nets and Step Transition Systems. In: *Int. J. Found. Comput. Sci.* 3 (1992), Nr. 4, S. 443–478 (Zitiert auf Seiten 11, 12, 136, 144, 172, 173, und 298.)

[174] NATHAN, M. ; KOEDINGER, K. R. ; ALIBALI, M.: Expert blind spot: When content knowledge eclipses pedagogical content knowledge. In: *AERA 2001, Seattle, USA*, 2001 (Zitiert auf Seite 80.)

[175] Nielsen, M. ; Plotkin, G.D. ; Winskel, G.: Petri Nets, Event Structures and Domains, Part I. In: *Theoretical Computer Science* 13 (1981), S. 85–108 (Zitiert auf Seiten 12, 104, 213, und 215.)

[176] Nielsen, M. ; Rozenberg, G. ; Thiagarajan, P. S.: Elementary Transition Systems. In: *Theor. Comput. Science* 96 (1992), Nr. 1, S. 3–33 (Zitiert auf Seite 11.)

[177] Nielsen, M. ; Thiagarajan, P. S.: Regular Event Structures and Finite Petri Nets: The Conflict-Free Case. In: Esparza, J. (Hrsg.) ; Lakos, C. (Hrsg.): *Proc. of ICATPN 2002, LNCS 2360*, Springer, 2002, S. 335–351 (Zitiert auf Seite 12.)

[178] O'Donnell, A.M. ; Dansereau, D.F.: Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. In: Hertz-Lazarowitz, R. (Hrsg.) ; Miller, N. (Hrsg.): *Interaction in cooperative groups: The theoretical anatomy ofgroup learning*, Cambridge University Press, 1992 (Zitiert auf Seite 74.)

[179] Oestereich, B.: Objektorientierte Geschäftsprozess-Modellierung und modellgetriebene Softwareentwicklung. In: *HMD - Praxis Wirtschaftsinformatik* 241 (2005) (Zitiert auf Seiten 45 und 50.)

[180] Oliveira Oliveira, M.: Hasse Diagram Generators and Petri Nets. In: Franceschinis, G. (Hrsg.) ; Wolf, K. (Hrsg.): *Proc. of Petri Nets 2009, LNCS 5606*, Springer, 2009, S. 183–203 (Zitiert auf Seiten 12, 15, 171, und 289.)

[181] Ott, D. ; Jaeschke, P. ; Endl, R.: *Requirements Engineering Barometer.* Studie der FHS St. Gallen, 2008 (Zitiert auf Seite 37.)

[182] Peterson, J.L.: *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981 (Zitiert auf Seiten 89 und 93.)

[183] Petri, C.A.: *Kommunikation mit Automaten*, Universität Bonn, Diss., 1962 (Zitiert auf Seiten 4 und 89.)

[184] Petri, C.A.: Rechnender Netzraum. In: *Spektrum der Wissenschaft* Spezial 3/07: Ist der Universum ein Computer? (2007), S. 16–19 (Zitiert auf Seite vii.)

[185] Pietkiewicz-Koutny, M.: The Synthesis Problem for Elementary Net Systems with Inhibitor Arcs. In: *Fundam. Inform.* 40 (1999), Nr. 2-3, S. 251–283 (Zitiert auf Seiten 12, 279, und 290.)

[186] Pietkiewicz-Koutny, M.: Synthesising Elementary Net Systems with Inhibitor Arcs from Step Transition Systems. In: *Fundam. Inform.* 50 (2002), Nr. 2, S. 175–203 (Zitiert auf Seiten 12, 279, und 290.)

[187] Pohl, K.: *Requirements Engineering*. dpunkt, 2008 (Zitiert auf Seiten 13, 37, 38, 45, 53, 58, 60, 61, und 98.)

[188] Pratt, V.: Modelling Concurrency with Partial Orders. In: *Int. Journal of Parallel Programming* 15 (1986), S. 33–71 (Zitiert auf Seiten 12, 98, und 99.)

[189] Ramadge, P.J. ; Wonham, W.M.: Supervisory Control of a Class of Discrete Event Processes. In: *SIAM Journal of Control and Optimization* 25 (1987), Nr. 1, S. 206–230 (Zitiert auf Seiten 13 und 319.)

[190] Rawlings, A. ; Rosmalen, P. ; Koper, R. ; Rodriguez Artacho, M. ; Lefrere, P.: Survey of Educational Modelling Languages (EMLs). In: *CEN/ISSS Learning Technology Workshop, Kopenhagen, Dänemark*, 2002 (Zitiert auf Seite 75.)

[191] Recker, J.: Process Modeling in the 21st Century. In: *BPTrends* 3 (2006), Nr. 5, S. 1–6 (Zitiert auf Seite 51.)

[192] Reisig, W.: *Petrinetze - Eine Einführung*. Springer, 1986 (Zitiert auf Seite 89.)

[193] Reveliotis, S. A. ; Choi, J.-Y.: Designing Reversibility-Enforcing Supervisors of Polynomial Complexity for Bounded Petri Nets Through the Theory of Regions. In: Donatelli, S. (Hrsg.) ; Thiagarajan, P. S. (Hrsg.): *Proc. of ICATPN 2006, LNCS 4024*, Springer, 2006, S. 322–341 (Zitiert auf Seiten 13, 319, und 334.)

[194] Riesz, M. ; Seckár, M. ; Juhás, G.: PetriFlow: A Petri Net Based Framework for Modelling and Control of Workflow Processes. In: *Workshop ART, ACSD 2010, Braga, Portugal*, 2010 (Zitiert auf Seiten 127 und 192.)

[195] Rodriguez Artacho, M.: PALO Language Overview / Universidad Nacional de Educacion a Distancia. 2002. – Forschungsbericht (Zitiert auf Seite 75.)

[196] Rozenberg, G. ; Engelfriet, J.: Elementary Net Systems. In: Reisig, W. (Hrsg.) ; Rozenberg, G. (Hrsg.): *Lectures on Petri Nets I: Basic Models, LNCS 1491*, Springer, 1998, S. 12–121 (Zitiert auf Seite 97.)

[197] Rozinat, A. ; Aalst, W. M. P. d.: Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In: Bussler, C. (Hrsg.) ; Haller, A. (Hrsg.): *Business Process Management Workshops, BPM 2005, LNCS 3812*, Springer, 2005, S. 163–176 (Zitiert auf Seite 320.)

[198] Rozinat, A. ; Jong, I.S.M.d. ; Günther, C.W. ; Aalst, W.M.P.d.: Process Mining Applied to the Test Process of Wafer Steppers in ASML. In: *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews* 39 (2009), Nr. 4, S. 474–479 (Zitiert auf Seite 67.)

[199] Rudie, K. ; Lafortune, S.: Minimal Communication in a Distributed Discrete-Event System. In: *IEEE Trans. on Automatic Control* 48 (2003), Nr. 6, S. 957–975 (Zitiert auf Seiten 13 und 334.)

[200] Salbrechter, A. ; Mayr, H. C. ; Kop, C.: Mapping Pre-Designed Business Process Models to UML. In: Hamza, M. H. (Hrsg.): *Proc. of IASTED Conf. on Software Engineering and Applications 2004*, IASTED/ACTA Press, 2004, S. 400–405 (Zitiert auf Seite 50.)

[201] Sandhu, R.S. ; Coyne, E.J. ; Feinstein, H.L. ; Youman, C.E.: Role-Based Access Control Models. In: *IEEE Computer* 29 (1996), Nr. 2, S. 38–47 (Zitiert auf Seite 76.)

[202] Schank, R.C. ; Abelson, R.P.: *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, 1977 (Zitiert auf Seite 74.)

[203] SCHEER: *IDS Scheer: ARIS Process Performance Manager*. – http://www.ids-scheer.com (Zitiert auf Seiten 50, 56, 67, und 70.)

[204] SCHEER, A.-W.: *Architecture of Integrated Information Systems: Foundations of Enterprise-Modeling*. Springer, 1992 (Zitiert auf Seite 50.)

[205] SCHEER, A.-W.: *ARIS – Vom Geschäftsprozess zum Anwendungssystem*. Springer, 2002 (Zitiert auf Seiten 45 und 46.)

[206] SCHEER, A.-W. ; NÜTTGENS, M.: ARIS Architecture and Reference Models for Business Process Management. In: AALST, W.M.P.d. van d. (Hrsg.) ; DESEL, J. (Hrsg.) ; OBERWEIS, A. (Hrsg.): *Business Process Management, Models, Techniques, and Empirical Studies, LNCS 1806*, Springer, 2000, S. 376–389 (Zitiert auf Seiten 50 und 54.)

[207] SCHRIJVER, A.: *Theory of Linear and Integer Programming*. Wiley, 1986 (Zitiert auf Seiten 12, 113, 114, 116, 117, 118, 119, 120, und 121.)

[208] SCHULTE, C. ; KNOBELSDORF, M.: Das informatische Weltbild von Studierenden. In: *INFOS 2007, LNI P-112*, GI, 2007, S. 69–80 (Zitiert auf Seite 30.)

[209] SEYBOLD, C. ; MEIER, S. ; GLINZ, M.: Scenario-Driven Modeling and Validation of Requirements Models. In: WHITTLE, J. (Hrsg.) ; GEIGER, L. (Hrsg.) ; MEISINGER, M. (Hrsg.): *Proc. of SCESM '06*, ACM, 2006, S. 83–89 (Zitiert auf Seiten 37 und 38.)

[210] SMITH, E.: *Zur Bedeutung der Concurrency-Theorie für den Aufbau hochverteilter Systeme*, Universität Hamburg, Diss., 1989 (Zitiert auf Seite 14.)

[211] SOLÉ, M. ; CARMONA, J.: Process Mining from a Basis of State Regions. In: LILIUS, J. (Hrsg.) ; PENCZEK, W. (Hrsg.): *Proc. of Petri Nets 2010, LNCS 6128*, Springer, 2010, S. 226–245 (Zitiert auf Seiten 13, 69, und 339.)

[212] TERZER, M.: *Large Scale Methods to Enumerate Extreme Rays and Elementary Modes*, Swiss Federal Institute of Technology, Zurich, Diss., 2009 (Zitiert auf Seiten 113, 114, 115, 116, und 231.)

[213] THIAGARAJAN, P. S.: Regular Event Structures and Finite Petri Nets: A Conjecture. In: BRAUER, W. (Hrsg.) ; EHRIG, H. (Hrsg.) ; KARHUMÄKI, J. (Hrsg.) ; SALOMAA, A. (Hrsg.): *Formal and Natural Computing, LNCS 2300*, Springer, 2002, S. 244–256 (Zitiert auf Seite 12.)

[214] VANDERBEI, R. J.: *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 1996 (Zitiert auf Seiten 116, 117, 118, und 121.)

[215] VERBEEK, E. ; TOORN, R. A.: Transit Case Study. In: CORTADELLA, J. (Hrsg.) ; REISIG, W. (Hrsg.): *Proc. of ICATPN 2004, LNCS 3099*, Springer, 2004, S. 391–410 (Zitiert auf Seite 249.)

[216] VOGLER, W.: *Modular Construction and Partial Order Semantics of Petri Nets. LNCS 625*. Springer, 1992 (Zitiert auf Seiten 104, 105, 107, und 305.)

[217] VOGLER, W.: Partial Words Versus Processes: a Short Comparison. In: ROZENBERG, G. (Hrsg.): *Advances in Petri Nets: The DEMON Project, LNCS 609*, Springer, 1992, S. 292–303 (Zitiert auf Seiten 104, 105, 107, und 305.)

[218] VOGTEN, H. ; MARTENS, H. ; NADOLSKI, R. ; TATTERSALL, C. ; ROSMALEN, P. ; KOPER, R: CopperCore Service Integration - Integrating IMS Learning Design and IMS Question and Test Interoperability. In: *Proc. of ICALT 2006*, IEEE Computer Society, 2006, S. 378–382 (Zitiert auf Seite 76.)

[219] WALTER, R.: *Petrinetzmodelle verteilter Algorithmen: Beweistechnik und Intuition*. Bertz, 1995 (Zitiert auf Seiten 32 und 33.)

[220] WEBER, M. ; KINDLER, E.: The Petri Net Markup Language. In: EHRIG, H. (Hrsg.) ; REISIG, W. (Hrsg.) ; ROZENBERG, G. (Hrsg.) ; WEBER, H. (Hrsg.): *Petri Net Technology for Communication-Based Systems, LNCS 2472*, Springer, 2003, S. 124–144 (Zitiert auf Seite 227.)

[221] WEINBERGER, A. ; ERTL, B. ; FISCHER, F. ; MANDL, H.: Epistemic and social scripts in computer-supported collaborative learning. In: *Instructional Science* 33 (2005), Nr. 1, S. 1–30 (Zitiert auf Seiten 74 und 77.)

[222] WERF, J. M. E. M. ; DONGEN, B. F. ; HURKENS, C. A. J. ; SEREBRENIK, A.: Process Discovery Using Integer Linear Programming. In: HEE, K. M. (Hrsg.) ; VALK, R. (Hrsg.): *Proc. of Petri Nets 2008, LNCS 5062*, Springer, 2008, S. 368–387 (Zitiert auf Seiten 69 und 74.)

[223] WESKE, M.: *Business Process Management – Concepts, Languages and Architectures*. Springer, 2007 (Zitiert auf Seiten 46, 49, 50, 75, und 77.)

[224] WESKE, M. ; GOESMANN, T. ; HOLTEN, R. ; STRIEMER, R.: A Reference Model for Workflow Application Development Processes. In: *Proc. of WACC 1999*, ACM, 1999, S. 1–10 (Zitiert auf Seite 50.)

[225] WIESNER, B. ; BRINDA, T.: Erfahrungen bei der Vermittlung algorithmischer Grundstrukturen im Infomatikunterricht der Realschule mit einem Robotersystem. In: *INFOS 2007, LNI P-112*, GI, 2007, S. 124–133 (Zitiert auf Seite 36.)

[226] WINSKEL, G.: Event Structures. In: BRAUER, W. (Hrsg.) ; REISIG, W. (Hrsg.) ; ROZENBERG, G. (Hrsg.): *Petri Nets: Central Models and Their Properties, Part II, LNCS 255*, Springer, 1986, S. 325–392 (Zitiert auf Seiten 12 und 98.)

[227] ZHOU, M.C. ; CESARE, F.D.: *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer, 1993 (Zitiert auf Seite 13.)

KOLOPHON

Diese Dissertation wurde mittels LaTeX 2ε unter der Verwendung der Schriftbilder Palatino von Hermann Zapf und Euler gesetzt (als Type 1 PostScript Fonts wurden URW Palladio L und FPL verwendet). Aufzählungen wurden dagegen mit Bera Mono gesetzt. Bera Mono wurde ursprünglich von Bitstream Inc. als Bitstream Vera entwickelt (Type 1 PostScript Fonts wurden von Malte Rosenau und Ulrich Dirr zugänglich gemacht).
Der typografische Stil wurde von Bringhursts Arbeit *The Elements of Typographic Style* [45] inspiriert. Er ist für LaTeX über CTAN unter der Bezeichnung „`classicthesis`“ verfügbar.
Die spezifische Größe des Textblocks wurde nach Bringhursts Vorgaben berechnet. 10 pt Palatino benötigt 133.21 pt für die Zeichenfolge „abcdefghijklmnopqrstuvwxyz“. Dies ergibt eine gute Zeilenlänge zwischen 24–26 pc (288–312 pt). Die Wahl eines „double square textblock“ mit einem 1:2 Verhältnis resultiert in einem Textblock der Maße 312:624 pt (welcher die Kopfzeile in diesem Design einschließt).

CURRICULUM VITAE

PERSÖNLICHE ANGABEN

Name	Sebastian Mauser
Geburtsdatum	31. August 1981
Geburtsort	Dachau
Nationalität	deutsch
Konfession	römisch-katholisch
Anschrift	Heinrich-Zille-Str. 9 58093 Hagen
Telefon	0160 / 94925066
E-Mail	sebastian.mauser@gmx.de

AUSBILDUNGSGANG

09.1988 – 09.1992	Besuch der Montessori-Schule Breitbrunn am Ammersee
09.1992 – 06.2001	Besuch des Carl-Spitzweg-Gymnasiums Unterpfaffenhofen, neusprachlicher Zweig Leistungskurse: Mathematik, Wirtschafts- und Rechtslehre
06.2001	Abitur mit Abschlussnote 1,0
10.2001 – 01.2006	Studium der Wirtschaftsmathematik an der Katholischen Universität Eichstätt-Ingolstadt Studienschwerpunkte: Wahrscheinlichkeitstheorie, Finanzmathematik, Finance & Banking, Petrinetztheorie, Softwareentwicklung
09.2003	Vordiplom in Wirtschaftsmathematik mit Note 1,09
02.2005 – 07.2005	Studiensemester an der Växjö University in Schweden im Rahmen des ERASMUS-Programms
01.2006	Diplom in Wirtschaftsmathematik mit Abschlussnote 1,00 Titel der Diplomarbeit: Semantiken von Petrinetzen - ein algebraischer Ansatz, der zwischen nebenläufigem und synchronem Verhalten unterscheidet Auszeichnung mit dem Maximilian-Bickhoff-Preis der Katholischen Universität Eichstätt-Ingolstadt

BERUFLICHE TÄTIGKEITEN

08.2002 – 01.2006	Wissenschaftliche Hilfskraft an dem Lehrstuhl für Angewandte Informatik der Katholischen Universität Eichstätt-Ingolstadt von Prof. Dr. Jörg Desel Aufgabenfelder: Korrekturarbeiten, Tutorien und Softwarentwicklung
10.2004 – 02.2005	Wissenschaftliche Hilfskraft an der Professur für Informatik der Katholischen Universität Eichstätt-Ingolstadt von Prof. Dr. Stephan Diehl Aufgabenfeld: Tutorium Programmiertechnik
08.2005 – 10.2005	Praktikum an der Bayerischen Hypo- und Vereinsbank AG in München im Bereich Global Derivatives / Quantitative Research & Structuring Aufgabenfeld: Entwicklung einer finanzmathematischen Software
01.2006 – 08.2010	Wissenschaftlicher Angestellter am Lehrstuhl für Angewandte Informatik der Katholischen Universität Eichstätt-Ingolstadt von Prof. Dr. Jörg Desel Umfangreiche Lehr- und Forschungstätigkeit Forschungsgebiete: Synthese, Verifikation und Entfaltung von Petrinetzen, Petrinetzsemantiken, Prozessterme, konzeptuelle Modellierung von Systemen und Szenarien, Geschäftsprozessmodellierung, Process Mining, Lernprozessmodellierung Erasmus-Betreungsdozent für die Växjö University und die Linnaeus University in Schweden
09.2010 – heute	Wissenschaftlicher Angestellter am Lehrgebiet Software Engineering der FernUniversität in Hagen von Prof. Dr. Jörg Desel