Halbordnungs- und Reduktionstechniken für die automatische Verifikation von verteilten Systemen

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigte Abhandlung

vorgelegt von

Claus Schröter

aus Hildesheim

Vorsitzender:	Univ.-Prof. Dr. Volker Diekert
Hauptberichter:	Univ.-Prof. Dr. Javier Esparza
Mitberichter:	Univ.-Prof. Dr. Jörg Desel, Kath. Univ. Eichstätt
Tag der mündlichen Prüfung:	21. Juli 2006

Institut für Formale Methoden der Informatik
der Universität Stuttgart

2006

Kurzfassung

Das Hauptproblem bei der automatischen Verifikation von verteilten Systemen unter Verwendung von Model-Checking-Verfahren besteht in der Zustandsraumexplosion der Ausgangssysteme. Dieses bedeutet, daß sogar die Spezifikation eines kleinen Systems oftmals einen großen Zustandsraum aufspannt, da dieser exponentiell in der Größe des Ausgangssystems wachsen kann. In der Vergangenheit wurden verschiedene Techniken vorgeschlagen, um diese Problematik in den Griff zu bekommen. Diese können klassifiziert werden in Techniken, die auf eine implizite, kompakte Repräsentation des gesamten Zustandsraums abzielen, zum Beispiel codiert als BDD, oder die unter Verwendung von Abstraktionen oder Halbordnungsreduktionen eine reduzierte Zustandsraumrepräsentation verwenden. Eine dritte Klasse von Techniken stützt sich auf die Halbordnungssicht auf nebenläufige Berechnungen, und Systemzustände werden hier implizit durch die Verwendung azyklischer Petrinetze repräsentiert. Sehr populär darunter ist die Technik von McMillan zur Erzeugung endlicher, vollständiger Präfixe von Petrinetzentfaltungen.
Das Ziel dieser Arbeit besteht darin, effiziente Halbordnungs- und Reduktionstechniken für die automatische Verifikation von verschiedenen Petrinetzklassen zu analysieren, zu erweitern, zu implementieren und experimentell zu vergleichen.
Eine der Hauptfragestellungen auf dem Gebiet der automatischen Verifikation untersucht die Erreichbarkeit von Systemzuständen, da viele Sicherheitseigenschaften eines Systems auf einfache Erreichbarkeitsfragen zurückgeführt werden können. Als typisches Beispiel sei hier die Eigenschaft zum wechselseitigen Ausschluß von nebenläufigen Prozessen erwähnt. In dieser Arbeit werden vier Verfahren betrachtet, die das Erreichbarkeitsproblem für 1-sichere Petrinetze unter Verwendung von Entfaltungspräfixen exakt lösen. Es werden zwei Halbentscheidungsverfahren betrachtet, die das Erreichbarkeitsproblem für 1-sichere Petrinetze unter Verwendung von Techniken der linearen Programmierung zu lösen versuchen. Die Verfahren werden anhand von experimentellen Ergebnissen miteinander verglichen, und es wird eine Empfehlung hergeleitet, welche Verfahren für praktische Belange geeignet zu sein scheinen. Es wird gezeigt, daß das PSPACE-vollständige Erreichbarkeitsproblem für 1-sichere Petrinetze in ein (möglicherweise exponentiell größeres) NP-vollständiges Problem überführt werden kann, sofern Entfaltungstechniken eingesetzt werden.
Eine weitere wichtige Fragestellung auf dem Gebiet der automatischen Verifikation beschäftigt sich mit dem Nachweis von Sicherheits- und Lebendigkeitseigenschaften, die in Form von temporalen Logiken wie beispielsweise CTL oder LTL formuliert werden. Es werden Reduktionsregeln für den Nachweis von LTL-X-Eigenschaften betrachtet, die in zwei Kategorien eingeteilt werden können: Regeln, die lineare Programmierung unter Verwendung von Invarianten oder impliziten Stellen einsetzen, und Regeln, die lokale Netzreduktionen betrachten. Es wird gezeigt, daß die Bedingungen für die Anwendbarkeit einiger lokaler Netzreduktionen abgeschwächt werden können, sofern man LTL-X-Eigenschaften mit einem Verfahren von Esparza und Heljanko nachweist. Anhand von experimentellen Ergebnissen wird gezeigt, daß die Verifikationszeit für eine LTL-X-Eigenschaft eines Systems signifikant verringert werden kann, sofern dieses mit

den vorgeschlagenen Reduktionsregeln vorverarbeitet wird.
Schließlich wird der von Esparza und Heljanko vorgestellte entfaltungsbasierte Ansatz zum Nachweis von LTL-X-Eigenschaften 1-sicherer Petrinetze auf eine höhere Petrinetzklasse übertragen. Anhand von experimentellen Ergebnissen wird der in dieser Arbeit vorgestellte Ansatz mit dem Ansatz von Esparza und Heljanko sowie dem Model-Checker Spin verglichen. Die Ergebnisse zeigen, daß der in dieser Arbeit verwendete Model-Checker dem Model-Checker von Esparza und Heljanko in allen Beispielen, und Spin gegenüber in einigen Beispielen überlegen ist.

Abstract

The main problem in automatic verification of distributed systems using model checking techniques is that it suffers from state space explosion. This means that even a relatively small system specification often yields a very large state space that may grow exponentially in the size of the original system. Many different techniques have been proposed to alleviate this problem. They can be classified into techniques that aim at an implicit compact representation of the full state space, for instance coded as a BDD, or at an explicit generation of a reduced state space presentation using abstraction or partial-order reduction techniques. A third class of techniques relies on the partial-order view of concurrent computation, and represents system states implicitly using acyclic Petri nets. Among them, McMillan's finite complete prefixes of Petri net unfoldings are a very popular technique.

The goal of this work is to analyse, to improve, to implement, and to compare efficient partial-order and Petri net reduction techniques for model checking tasks regarding different classes of Petri nets.

One of the key questions in the area of automatic verification is the reachability of states because many safety properties of systems can be reduced to simple reachability properties. A typical example is the mutual-exclusion property of mutual-exclusion algorithms. This work studies four solutions to the reachability problem for 1-safe Petri nets that are based on the unfolding technique. Two semi-decision algorithms are presented that try to solve the reachability problem for 1-safe Petri nets without unfoldings but with linear programming techniques. All algorithms are tested on a set of examples and a recommendation is extracted on which algorithms should be used and which ones not. It is shown that the PSPACE-complete reachability problem for 1-safe Petri nets can be translated into a (probably exponentially larger) NP-complete problem using Petri net unfoldings.

Another important task in the area of automatic verification is the verification of safety and liveness properties which are expressed in terms of temporal logics like CTL and LTL. A set of reduction rules for LTL-X model checking of 1-safe Petri nets is presented. The reduction techniques are of two kinds: Linear programming techniques based on Petri net techniques like invariants and implicit places, and local net reductions. It is shown that the conditions for the application of some local net reductions can be weakened using an LTL-X model checking approach which has been proposed by Esparza and Heljanko. A number of experimental results is presented, and it is shown that the model checking time of a net system can be significantly decreased if it has been preprocessed with the proposed reduction techniques.

Finally, the unfolding-based approach to LTL-X model checking of 1-safe Petri nets proposed by Esparza and Heljanko is extended to high-level Petri nets. A number of experimental results is presented comparing the approach of this thesis with the one proposed by Esparza and Heljanko and the well-known model checker Spin, and it is shown that the model checker presented in this thesis beats the one of Esparza and Heljanko on all examples, and has the edge over Spin in some examples.

Danksagung

Die vorliegende Arbeit wurde am Institut für Künstliche Intelligenz und Theoretische Informatik der Technischen Universität München begonnen, am Laboratory for Foundations of Computer Science der University of Edinburgh fortgesetzt und schließlich am Institut für Formale Methoden der Informatik der Universität Stuttgart beendet.
Mein ganz besonderer Dank gilt Herrn Prof. Dr. Javier Esparza für die vielen fruchtbaren Anregungen zu dieser Thematik und für sein lebhaftes Interesse während der Durchführung dieser Arbeit. Seine allzeit vorhandene Diskussionsbereitschaft und seine Geduld haben entscheidend zum Gelingen dieser Arbeit beigetragen.
Mein weiterer Dank gilt Herrn Prof. Dr. Jörg Desel für seine Tätigkeit als Mitberichter.
Allen meinen Kollegen an den Universitäten München, Edinburgh und Stuttgart sei herzlich gedankt für ihre ständige Unterstützung und Hilfsbereitschaft sowie viele fruchtbare Diskussionen, wobei ich in diesem Zusammenhang insbesondere meine langjährigen Weggefährten Dr. Stefan Römer, Dr. Stefan Schwoon und Dr. Alin Stefanescu erwähnen möchte, die jederzeit ein offenes Ohr für meine Forschungsanliegen hatten.
Mein weiterer Dank gilt Dr. Keijo Heljanko und Dr. Victor Khomenko für die zahlreichen Diskussionen und Anregungen bezüglich des entfaltungsbasierten LTL-X Model-Checkers.
Meiner Familie und allen meinen Freunden danke ich ganz herzlich für ihre fortwährende Unterstützung. Sie haben es jederzeit verstanden, für den notwendigen Ausgleich zu sorgen, der während der Anfertigung einer solchen Arbeit vorhanden sein muß.

München, im Oktober 2005 Claus Schröter

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Bei der Entwicklung von Softwaresystemen ist es unabdingbar, den Entwicklungsprozeß durch qualitätssichernde Maßnahmen zu unterstützen [Wal90, Tra93]. Neben Qualitätsmerkmalen wie beispielsweise Verfügbarkeit, Leistung, Benutzerfreundlichkeit, Wartungsfreundlichkeit und Portabilität ergeben sich darüberhinaus noch drei weitere, äußerst wichtige Qualitätsmerkmale [Wal90]:

- Die *Sicherheit* eines Softwaresystems beschreibt dessen Eigenschaft, daß unter vorgegebenen Rahmenbedingungen keine Ereignisse auftreten können, die unerwünscht sind und Schaden verursachen.
- Die *Zuverlässigkeit* eines Softwaresystems beschreibt dessen Fähigkeit, eine bestimmte Funktion unter vorgegebenen Rahmenbedingungen für einen vorgegebenen Zeitraum zu erfüllen.
- Die *Funktionserfüllung* eines Softwaresystems beschreibt dessen Eignung, Funktionen in Übereinstimmung mit den vorgegebenen Rahmenbedingungen gemäß ihrer Spezifikation auszuführen.

Um Softwaresysteme zu entwickeln, die einen möglichst hohen Standard bezüglich dieser Qualitätsmerkmale aufweisen, haben sich die Methoden des *Testens* und der *formalen Verifikation* als qualitätssichernde Maßnahmen etabliert. Durch Testen soll überprüft werden, ob sich das Softwaresystem gemäß seiner Spezifikation verhält, indem es mit einer stichprobenartig ausgewählten Menge von Eingabewerten versehen und abgearbeitet wird. Das Ziel des Testens besteht also darin, möglichst alle im Softwaresystem vorhandenen Fehler aufzudecken. Dieses Vorhaben ist jedoch in der Praxis nicht realisierbar, da durch das Testen einerseits immer nur ein Teil des gesamten Systemverhaltens abgedeckt wird und andererseits lediglich die Anwesenheit von Fehlern, aber nicht deren

Abwesenheit festgestellt werden kann. Somit bleibt immer eine Unsicherheit bezüglich der Restfehlerzahl eines Softwaresystems bestehen.
Dahingegen wird mit den Methoden der *formalen Verifikation* eine ausgiebige Untersuchung des gesamten Systemverhaltens vorgenommen. In diesem Zusammenhang hat sich im Bereich der formalen Verifikation von Systemen mit endlichen Zustandsräumen die Methode des *Model-Checkings* [CGP99] bewährt, welche für ein System durch (explizite oder implizite) Aufzählung aller Systemzustände die Frage beantwortet, ob das System eine bestimmte Eigenschaft erfüllt oder nicht. Model-Checking-Techniken weisen gegenüber anderen formalen Methoden wie beispielsweise Theorembeweisern den Vorteil auf, daß sie

(i) *voll automatisiert* ablaufen, d.h., bei Eingabe eines Systems und einer Eigenschaft terminieren die Algorithmen immer mit einer „Ja/Nein"-Antwort, und

(ii) im Falle einer „Nein"-Antwort ein die Eigenschaft verletzendes *Gegenbeispiel* liefern, was die anschließende Fehlersuche im System erheblich erleichtert [CGP99].

Es gibt eine Fülle von Eigenschaften, auf die ein System untersucht werden kann, aber aus praxisorientierter Sichtweise haben sich vor allem die folgenden drei Eigenschaftstypen als interessant erwiesen:

- *Nachweis von Verklemmungsfreiheit*
  Diese Eigenschaft betrifft die Fragestellung, ob ein System möglicherweise in einen Zustand gelangt, in dem keine Aktion mehr ausgeführt werden kann, d.h., das System „tot" ist. Eine solche Situation kann beispielsweise eintreten, wenn in einem System zwei Prozesse miteinander kommunizieren und jeder Prozeß auf ein Berechnungsergebnis des anderen Prozesses wartet, das benötigt wird, um selbst weiterrechnen zu können.

- *Nachweis der Erreichbarkeit eines Systemzustandes*
  Bei dieser Fragestellung wird untersucht, ob es einen Systemablauf gibt, der das System in einen vorher festgelegten Zustand überführt. Beispielsweise gilt bei den Algorithmen zum wechselseitigen Ausschluß kritischer Bereiche („*mutual-exclusion*") [Ray86] die Eigenschaft, daß sich zwei oder mehrere Prozesse niemals gleichzeitig innerhalb eines kritischen Bereichs befinden können. Diese Eigenschaft kann nachgewiesen werden, indem gezeigt wird, daß kein Systemzustand erreichbar ist, in dem sich zwei oder mehrere Prozesse gleichzeitig innerhalb eines kritischen Bereichs befinden.

- *Nachweis temporallogischer Eigenschaften*
  Durch den Einsatz temporaler Logiken besteht die Möglichkeit, Eigenschaften mit zeitlichen Abhängigkeiten zu versehen. Auf diesem Wege können die meisten, für Praxisbelange relevanten Sicherheits- und Zuverlässigkeitseigenschaften formuliert und anschließend verifiziert werden.

Die Hauptproblematik beim Einsatz von Model-Checking-Techniken besteht in der Zustandsraumexplosion, die bei komplexen Systemen, welche aus vielen nebenläufigen Komponenten bestehen, akut zum Vorschein kommt. Es ist also nicht weiter verwunderlich, daß man bei der Anwendung von Verfahren, die explizit den gesamten Zustandsraum erzeugen und durchsuchen, relativ schnell an die Grenzen der Rechnerkapazitäten stößt. Deshalb wurde in der Vergangenheit das Augenmerk verstärkt auf die Entwicklung von Methoden gelegt, die den Zustandsraum nicht explizit erzeugen, sondern stattdessen kompakte Kodierungen des Zustandsraums verwenden, auf deren Informationen effizient zugegriffen werden kann.
In diesem Zusammenhang haben sich *symbolische Techniken* [McM93] etabliert, bei denen eine symbolische Repräsentation des Zustandsraums in Form von beispielsweise OBDDs [Bry86] vorliegt. Andere Methoden versuchen, der Zustandsraumexplosion mittels halbordnungsbasierter Reduktionstechniken entgegenzuwirken. Als Beispiele hierfür seien die Konzepte der *sturen* [Val90], *weiten* [Pel94] und *schlafenden Mengen* [God90] erwähnt. Darüberhinaus haben sich im Bereich der Petrinetztheorie *Netzentfaltungen* [McM92, ERV02, KK03, DJN04] und *Prozeßnetze* [Des00] als Halbordnungssemantik für verteilte Systeme bewährt und sind zu Analysezwecken und zur Lösung zahlreicher Verifikationsprobleme eingesetzt worden [McM92, Esp93, Esp94, MR97, Mel98, ER99, Hel99, Röm00, DE00, KK00, ES01a, ES01b, KK01, Hel02, KKV02, DJLN03, Kho03, SK04, DMN04]. Zusätzlich besteht bei Systemen, die ein hohes Datenvolumen aufweisen, die Möglichkeit, den Zustandsraum dadurch einzugrenzen, daß von den Daten abstrahiert wird und lediglich eine *Abstraktion* des ursprünglichen Systemmodells für die Verifikationszwecke verwendet wird. Solche Abstraktionen zur Reduzierung der Komplexität von Model-Checking-Problemen sind beispielsweise in [CGL94, DGG97] betrachtet worden.
Dennoch beschränkt weiterhin vor allem die Problematik der Zustandsraumexplosion den Einsatz von Model-Checking-Techniken in der Praxis, sodaß in dieser Arbeit die Lücke zwischen Theorie und Praxis durch die Erweiterung der Anwendbarkeit ausgewählter Techniken und die experiementelle Messung und Analyse derer Leistungen geschlossen werden soll.
Unter diesem Hintergrund bestand das

> *Ziel dieser Arbeit darin, effiziente Model-Checking-Algorithmen für die Verifikation von verteilten Systemen unter Verwendung von Halbordnungs- und Reduktionstechniken zu analysieren, zu erweitern, zu implementieren und experimentell zu vergleichen.*

Als grundlegender Formalismus werden in dieser Arbeit Petrinetze [DE95, Rei85] verwendet, da sie wie kein anderer Formalismus eine getrennte Sichtweise einerseits auf die Struktur und andererseits auf das Verhalten eines Systems erlauben. Dieses wirkt sich wiederum vorteilhaft auf die Analysemethoden aus, da aufgrund struktureller Systemeigenschaften Rückschlüsse auf das dynamische Systemverhalten gezogen werden können. Im Verlauf dieser Arbeit werden Verfahren betrachtet, die das Erreichbarkeitsproblem für sichere S/T-Netzsysteme mittels Netzentfaltungen exakt lösen, Halbentscheidungs-

verfahren, die das Erreichbarkeitsproblem für sichere S/T-Netzsysteme mit Techniken der linearen Programmierung zu lösen versuchen, Verfahren, welche die Gültigkeit einer temporallogischen Eigenschaft für sichere S/T-Netzsysteme exakt entscheiden und deren Effizienz durch vorgeschaltete Netzreduktionen gesteigert werden kann, sowie exakte Verfahren für den Nachweis temporallogischer Eigenschaften von M-Netzsystemen.

Im einzelnen ist die Arbeit wie folgt gegliedert:
In Kapitel 2 werden die zum Verständnis der Arbeit notwendigen mathematischen Grundlagen sowie zwei verschiedene Petrinetzklassen eingeführt, die als grundlegender Formalismus zur Beschreibung der verteilten Systeme dienen.
In Kapitel 3 werden vier verschiedene Algorithmen betrachtet, welche die Fragestellung nach der Erreichbarkeit von Teilmarkierungen in sicheren S/T-Netzsystemen exakt beantworten. Dabei weisen die vier Algorithmen die Gemeinsamkeit auf, daß die Erreichbarkeitsanalyse unter Verwendung von Halbordnungssemantiken der Netzsysteme durchgeführt wird, die in Form von McMillans Netzentfaltungen vorliegen. Die Algorithmen werden anhand von Testreihen miteinander verglichen und analysiert.
Demgegenüber werden in Kapitel 4 zwei Halbentscheidungsverfahren zur Erreichbarkeitsanalyse in sicheren S/T-Netzsystemen vorgestellt, die keine Halbordnungstechniken einsetzen, sondern das Erreichbarkeitsproblem mittels Methoden der linearen Programmierung zu lösen versuchen. Dabei wird der Zustandsraum sukzessiv durch die Festlegung von Schnittebenen eingeschränkt, die aufgrund struktureller und verhaltensorientierter Netzeigenschaften bestimmt werden. Auch diese beiden Algorithmen werden anhand von Testreihen bezüglich ihrer Performance analysiert.
In Kapitel 5 werden Reduktionstechniken beschrieben, die ein sicheres S/T-Netzsystem in dem Sinne „verhaltensäquivalent“ verkleinern, sodaß die Verifikation einer temporallogischen Eigenschaft mittels des in [EH01, Hel02] beschriebenen Verfahrens sowohl für das ursprüngliche Netzsystem als auch für das reduzierte Netzsystem dasselbe Ergebnis liefert. Anhand von Testreihen wird analysiert, inwiefern sich die Reduktionen auf die Performance des in [EH01, Hel02] beschriebenen Model-Checkers auswirken.
In Kapitel 6 wird der in [EH01, Hel02] vorgeschlagene Algorithmus für die Verifikation von temporallogischen Eigenschaften für sichere S/T-Netzsysteme auf M-Netzsysteme übertragen, die eine natürlichere Modellierung und kompaktere Darstellung von Systemen erlauben als sichere S/T-Netzsysteme. Das Kapitel wird abgerundet durch eine Analyse des Verfahrens unter Einbezug experimenteller Ergebnisse.
Die Arbeit schließt in Kapitel 7 mit Schlußbemerkungen. In den Anhängen befinden sich Beschreibungen aller im Rahmen dieser Arbeit verwendeten Fallbeispiele und implementierten Algorithmen.

Kapitel 2

Allgemeine Grundlagen

In diesem Kapitel werden die zum Verständnis der vorliegenden Arbeit notwendigen grundlegenden Begriffe und Schreibweisen eingeführt.
Dabei wird in Kapitel 2.1 zunächst auf die für diese Arbeit wichtigsten mathematischen Grundlagen und Konzepte eingegangen, wobei kein Anspruch auf Vollständigkeit besteht, sondern lediglich auszugsweise die Notationen und Konzepte vorgestellt werden, die auch in dieser Arbeit Verwendung finden. Für eine detaillierte Einführung sei auf [BS89] verwiesen.
In Kapitel 2.2 auf Seite 28 wird ein Einblick in die Welt der Petrinetztheorie gegeben, da Petrinetzsysteme in dieser Arbeit den für die Modellierung nebenläufiger Systeme zugrundeliegenden Formalismus bilden. Hierbei finden die beiden Petrinetzklassen der sogenannten S/T- und M-Netzsysteme besondere Beachtung. Eine umfassende Einführung in die Petrinetztheorie kann in [DE95, Rei85, Mur89, RR98a, RR98b] nachgelesen werden.

2.1 Mathematische Grundlagen

Definition 2.1.1 (Grundbegriffe der mathematischen Logik)
Die Konstanten **wahr** und **falsch**, der einstellige Junktor $\neg$ (*Negation*), die zweistelligen Junktoren $\wedge$ (*Konjunktion*), $\vee$ (*Disjunktion*), $\Rightarrow$ (*Implikation*) und $\Leftrightarrow$ (*Äquivalenz*) sowie die Quantoren $\exists$ (*Existenzquantor*) und $\forall$ (*Allquantor*) werden in ihren herkömmlichen Bedeutungen verwendet.

Definition 2.1.2 (Mengen)
Die Schreibweise $\in$ (*ist Element von*) sowie die mengentheoretischen Operationen $\cup$ (*Vereinigung*), $\cap$ (*Durchschnitt*), $\setminus$ (*Differenz*), $\subseteq$ (*Inklusion*), $\subset$ (*echte Inklusion*) und

$\times$ (*kartesisches Produkt*) werden in ihren herkömmlichen Bedeutungen verwendet. Die *Kardinalität* einer Menge X wird als $|X|$ geschrieben. Desweiteren bezeichnen

$$\begin{aligned}
\emptyset &= \{x \mid x \neq x\} && \text{die } \textit{leere} \text{ Menge} \\
\mathbb{N} &= \{0, 1, 2, 3, \dots\} && \text{die Menge der } \textit{natürlichen} \text{ Zahlen} \\
\mathbb{Z} &= \{0, \pm 1, \pm 2, \pm 3, \dots\} && \text{die Menge der } \textit{ganzen} \text{ Zahlen} \\
\mathbb{Q} &= \left\{ \frac{n}{m} \mid n \in \mathbb{Z}, m \in \mathbb{N} \setminus \{0\} \right\} && \text{die Menge der } \textit{rationalen} \text{ Zahlen}
\end{aligned}$$

Die Menge der *reellen* Zahlen wird als $\mathbb{R}$ geschrieben. Zwei Mengen X und Y heißen *disjunkt*, falls $X \cap Y = \emptyset$.
Eine Menge $\mathcal{P}_X = \{X_1, X_2, \dots, X_n\}$ von disjunkten, nichtleeren Teilmengen einer Menge X wird als *Partition* von X bezeichnet, falls

$$X = \bigcup_{1 \leq i \leq n} X_i$$

Definition 2.1.3 (Relationen)
Gegeben seien die beiden Mengen X und Y. Eine Teilmenge $R \subseteq X \times Y$ wird als (*binäre*) *Relation zwischen der Menge X und der Menge Y* bezeichnet. Im Falle $X = Y$ heißt $R \subseteq X \times X$ eine (*binäre*) *Relation in der Menge X*. Anstelle von $(x, y) \in R$ wird auch die Notation xRy verwendet. Eine binäre Relation R in der Menge X heißt

reflexiv,	falls	$\forall x \in X : xRx$
irreflexiv,	falls	$\forall x \in X : \neg(xRx)$
symmetrisch,	falls	$\forall x, y \in X : xRy \Rightarrow yRx$
asymmetrisch,	falls	$\forall x, y \in X : xRy \Rightarrow \neg(yRx)$
antisymmetrisch,	falls	$\forall x, y \in X : xRy \wedge yRx \Rightarrow x = y$
transitiv,	falls	$\forall x, y, z \in X : xRy \wedge yRz \Rightarrow xRz$
linear,	falls	$\forall x, y \in X : xRy \vee yRx$
konnex,	falls	$\forall x, y \in X : xRy \vee (x = y) \vee yRx$

Die *reflexive, transitive Hülle* einer binären Relation R wird mit R^* bezeichnet. Eine binäre Relation R wird als *Äquivalenzrelation* bezeichnet, falls R reflexiv, symmetrisch und transitiv ist.

Definition 2.1.4 (Ordnungen)
Gegeben sei eine binäre Relation R in der Menge X. R heißt

- *reflexive* (*irreflexive*) *Halbordnung in der Menge X*, falls R reflexiv (irreflexiv), antisymmetrisch (asymmetrisch) und transitiv ist. Für eine reflexive (irreflexive) Halbordnung R in der Menge X wird für zwei Elemente $x, y \in X$ anstatt xRy auch die Schreibweise $x \sqsubseteq y$ ($x \sqsubset y$) verwendet und nennt x *kleiner* (*echt kleiner*)

als y bzw. y *größer* (*echt größer*) als x. Für eine Halbordnung R in der Menge X und eine Menge $Y \subseteq X$ wird ein Element $y \in Y$ als *minimales Element* von Y bezüglich R bezeichnet, falls

$$\forall x \in X \colon (x \in Y \Rightarrow x \not\sqsubset y)$$

- *reflexive* (*irreflexive*) *Ordnung in der Menge* X, falls R eine reflexive (irreflexive) Halbordnung in der Menge X ist und zudem die Eigenschaft der Linearität (Konnexität) aufweist.

- *reflexive* (*irreflexive*) *Wohlordnung in der Menge* X, falls R eine reflexive (irreflexive) Ordnung in der Menge X ist und zudem jede nichtleere Teilmenge $Y \subseteq X$ ein bezüglich R minimales Element besitzt.

Für eine Halbordnung R in der Menge X wird eine nichtleere Teilmenge $Y \subseteq X$ als *Kette* (bezüglich R) bezeichnet, falls die Relation $R \cap (Y \times Y)$ eine Ordnung in der Menge Y darstellt. Eine Halbordnung R in der Menge X heißt *fundiert*, falls jede Kette ein bezüglich R minimales Element aufweist.

Definition 2.1.5 (Abbildungen oder Funktionen)
Gegeben seien die beiden Mengen X und Y. Eine binäre Relation f zwischen der Menge X und der Menge Y heißt *Abbildung* (oder *Funktion*) *von* X *in* Y, falls es zu jedem $x \in X$ mindestens ein $y \in Y$ mit $(x, y) \in f$ gibt und f zudem *eindeutig* ist, d.h.,

$$\forall x \forall y_1 \forall y_2 \colon (x, y_1) \in f \wedge (x, y_2) \in f \Rightarrow y_1 = y_2$$

Dabei werden X als *Definitionsbereich*, Y als *Wertebereich* und

$$f[X] = \{y \mid (x, y) \in f\}$$

als *Bildbereich* (oder *Wertemenge*) von f bezeichnet. Eine Abbildung f von X in Y wird auch als $f \colon X \to Y$ mit $x \mapsto f(x)$ notiert. Falls $(x, y) \in f$, wird auch $y = f(x)$ geschrieben. Eine Abbildung $f \colon X \to Y$ heißt

injektiv,	falls	$\forall x_1 \forall x_2 \colon f(x_1) = f(x_2) \Rightarrow x_1 = x_2$
surjektiv,	falls	$f[X] = Y$
bijektiv,	falls	f injektiv und surjektiv ist

Für eine Abbildung $f \colon X \to Y$ wird $f^{-1} \colon f[X] \to X$ als *Umkehrabbildung* (oder *Umkehrfunktion*) von f bezeichnet, falls f^{-1} eine Abbildung ist. Wird der Definitionsbereich einer Abbildung $f \colon X \to Y$ auf eine Teilmenge $Z \subseteq X$ eingeschränkt, so bezeichnet $f|_Z \colon Z \to Y$ die *Restriktion* von f auf Z.

Definition 2.1.6 (Multimengen)
Gegeben sei eine Menge X. Eine Abbildumg $\mu\colon X \to \mathbb{N}$ wird als *Multimenge* über der Menge X bezeichnet. Die Menge

$$\mathcal{M}(X) = \{\mu \mid \mu\colon X \to \mathbb{N}\}$$

beschreibt die Menge aller Multimengen über X. Eine Multimenge μ heißt *endlich*, falls die Menge $\{x \in X \mid \mu(x) > 0\}$ endlich ist. Die Menge aller endlichen Multimengen über X wird mit $\mathcal{M}_f(X)$ bezeichnet. Eine Multimenge μ stellt eine *Teilmenge* von X dar, falls $\mu(x) \leq 1$ für alle $x \in X$ gilt. Die mengentheoretischen Operationen werden wie folgt auf Multimengen übertragen:

$$\begin{array}{ll}
\textit{Vereinigung:} & (\mu_1 \cup \mu_2)(x) = \max(\mu_1(x), \mu_2(x)) \\
\textit{Durchschnitt:} & (\mu_1 \cap \mu_2)(x) = \min(\mu_1(x), \mu_2(x)) \\
\textit{Summe:} & (\mu_1 + \mu_2)(x) = \mu_1(x) + \mu_2(x) \\
\textit{Differenz:} & (\mu_1 \setminus \mu_2)(x) = \begin{cases} \mu_1(x) - \mu_2(x) & \text{falls } \mu_1(x) \geq \mu_2(x) \\ 0 & \text{sonst} \end{cases}
\end{array}$$

Bezüglich der *Multimengeninklusion* gilt:

$$\mu_1 \subseteq \mu_2 \Leftrightarrow \forall x \in X\colon \mu_1(x) \leq \mu_2(x)$$

Definition 2.1.7 (Vektoren und Matrizen)
n-dimensionale Vektoren und $(n \times m)$-*Matrizen* werden in ihren herkömmlichen Schreibweisen

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \text{und} \qquad M = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}$$

verwendet. Das *Skalarprodukt* $X \cdot Y$ zweier Vektoren X und Y, deren komponentenweisen Vergleichsoperatoren $X\ op\ Y$ für $op \in \{=, <, \leq, \geq, >\}$ sowie die *Transponierte* X^t eines Vektors X werden wie üblich definiert. Der *Nullvektor* wird mit $\mathbf{0}$ und der *Einsvektor* wird mit $\mathbf{1}$ bezeichnet.
Für eine endliche Menge $X = \{x_1, x_2, \dots, x_n\}$ kann jede Abbildung $f\colon X \to \mathbb{R}$ auch als n-dimensionaler Vektor

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}$$

aufgefaßt werden. Ebenso kann jede Abbildung $f\colon X \times Y \to \mathbb{R}$ für zwei endliche Mengen X und Y durch eine $(|X| \times |Y|)$-Matrix beschrieben werden.

Definition 2.1.8 (Arithmetische Ausdrücke)
Gegeben seien die disjunkten Mengen V von Variablen und Z von Werten. Die Menge $\mathcal{A}_{exp}(V \cup Z)$ der *arithmetischen Ausdrücke* über $V \cup Z$ ist induktiv wie folgt definiert:

- $v \in V$ bezeichnet einen arithmetischen Ausdruck.
- $z \in Z$ bezeichnet einen arithmetischen Ausdruck.
- Bezeichnen a_1 und a_2 arithmetische Ausdrücke, dann stellen auch

$$a_1 + a_2, \quad a_1 - a_2, \quad a_1 \cdot a_2 \quad \text{und} \quad a_1 \div a_2$$

  arithmetische Ausdrücke dar.

Für einen arithmetischen Ausdruck $a \in \mathcal{A}_{exp}(V \cup Z)$ bezeichnet $Var(a) \subseteq V$ die Menge der Variablen, die in a vorkommen. Dabei ist $Var(a)$ wie folgt induktiv über die Struktur von a definiert:

- $Var(z) = \emptyset$ für $z \in Z$
- $Var(v) = \{v\}$ für $v \in V$
- $Var(a_1 \; op \; a_2) = Var(a_1) \cup Var(a_2)$ für $op \in \{+, -, \cdot, \div\}$

Die Erweiterung von Var auf Mengen von arithmetischen Ausdrücken erfolgt im herkömmlichen Sinn, d.h., für $X \subseteq \mathcal{A}_{exp}(V \cup Z)$ gilt:

$$Var(X) = \bigcup_{x \in X} Var(x)$$

Für einen arithmetischen Ausdruck $a \in \mathcal{A}_{exp}(V \cup Z)$ definiert die Abbildung

$$\varsigma : Var(a) \to Z$$

eine *Variablenbelegung* für a, unter der a ausgewertet werden kann. Die Evaluierung von a unter ς wird als $a[\varsigma]$ notiert und ist wie folgt induktiv über die Struktur von a definiert:

- $z[\varsigma] = z$ für $z \in Z$
- $v[\varsigma] = \varsigma(v)$ für $v \in Var(a)$
- $(a_1 \; op \; a_2)[\varsigma] = a_1[\varsigma] \; op \; a_2[\varsigma]$ für $op \in \{+, -, \cdot, \div\}$

Definition 2.1.9 (Boolesche Ausdrücke)
Gegeben seien die paarweise disjunkten Mengen V von Variablen, V_b von booleschen Variablen und Z von Werten. Die Menge $\mathcal{B}_{exp}(V \cup V_b \cup Z)$ der *booleschen Ausdrücke* über $V \cup V_b \cup Z$ ist wie folgt induktiv definiert:

- **wahr** und **falsch** bezeichnen boolesche Ausdrücke.
- $v \in V_b$ bezeichnet einen booleschen Ausdruck.
- Bezeichnen a_1 und a_2 arithmetische Ausdrücke, d.h., $a_1, a_2 \in \mathcal{A}_{exp}(V \cup Z)$, dann stellen auch

  $$a_1 = a_2, \quad a_1 \neq a_2, \quad a_1 \leq a_2, \quad a_1 < a_2, \quad a_1 \geq a_2, \quad a_1 > a_2$$

  boolesche Ausdrücke dar.
- Bezeichnen b_1 und b_2 boolesche Ausdrücke, dann stellen auch

  $$\neg b_1, \quad b_1 \wedge b_2, \quad b_1 \vee b_2, \quad b_1 \Rightarrow b_2, \quad b_1 \Leftrightarrow b_2$$

  boolesche Ausdrücke dar.

Die Menge der booleschen Ausdrücke, die lediglich über den Operatoren der Menge

$$Op \subset \{=, \neq, \leq, <, \geq, >, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$$

gebildet werden, wird als $\mathcal{B}_{exp}(V \cup V_b \cup Z)|_{Op}$ notiert. Für einen booleschen Ausdruck $b \in \mathcal{B}_{exp}(V \cup V_b \cup Z)$ bezeichnet $Var(b) \subseteq V \cup V_b$ die Menge der Variablen, die in b vorkommen. Dabei ist $Var(b)$ wie folgt induktiv über die Struktur von b definiert:

- $Var(\textbf{wahr}) = Var(\textbf{falsch}) = \emptyset$
- $Var(v) = \{v\}$ für $v \in V_b$
- $Var(a_1\ op\ a_2) = Var(a_1) \cup Var(a_2)$ für $op \in \{=, \neq, \leq, <, \geq, >\}$
- $Var(\neg b) = Var(b)$
- $Var(b_1\ op\ b_2) = Var(b_1) \cup Var(b_2)$ für $op \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$

Die Erweiterung von Var auf Mengen von booleschen Ausdrücken erfolgt im herkömmlichen Sinn, d.h., für $X \subseteq \mathcal{B}_{exp}(V \cup V_b \cup Z)$ gilt:

$$Var(X) = \bigcup_{x \in X} Var(x)$$

Für einen booleschen Ausdruck $b \in \mathcal{B}_{exp}(V \cup V_b \cup Z)$ beschreibt die Abbildung

$$\varsigma \colon Var(b) \to Z \cup \{\textbf{wahr}, \textbf{falsch}\}$$

mit der Eigenschaft

$$\forall v \in Var(b) \colon v \in V \Leftrightarrow \varsigma(v) \in Z$$

eine *Variablenbelegung* von b, unter der b zu **wahr** oder **falsch** evaluiert werden kann. Falls b unter ς zu **wahr** evaluiert wird, dann *erfüllt* die Variablenbelegung ς den booleschen Ausdruck b. Dieser Sachverhalt wird mit $\varsigma \models b$ ausgedrückt. Dabei ist die Erfüllbarkeitsrelation $\models$ wie folgt induktiv über die Struktur von b definiert:

$$
\begin{array}{lcll}
\varsigma \models \textbf{wahr} & & & \\
\varsigma \models v \in V_b & \Leftrightarrow & \varsigma(v) = \textbf{wahr} & \\
\varsigma \models (a_1 \; op \; a_2) & \Leftrightarrow & a_1[\varsigma] \; op \; a_2[\varsigma] & \text{für } op \in \{=, \neq, \leq, <, \geq, >\} \\
\varsigma \models \neg b & \Leftrightarrow & \varsigma \not\models b & \\
\varsigma \models (b_1 \wedge b_2) & \Leftrightarrow & \varsigma \models b_1 \wedge \varsigma \models b_2 & \\
\varsigma \models (b_1 \vee b_2) & \Leftrightarrow & \varsigma \models b_1 \vee \varsigma \models b_2 & \\
\varsigma \models (b_1 \Rightarrow b_2) & \Leftrightarrow & \varsigma \models \neg b_1 \vee \varsigma \models b_2 & \\
\varsigma \models (b_1 \Leftrightarrow b_2) & \Leftrightarrow & \varsigma \models (b_1 \Rightarrow b_2) \wedge \varsigma \models (b_2 \Rightarrow b_1) &
\end{array}
$$

Definition 2.1.10 (Sprachen)
Gegeben sei eine Menge A von Symbolen (ein *Alphabet*). Eine beliebige (endliche oder unendliche) Folge $a_1 a_2 \ldots$ von Elementen aus A wird als *Wort* bezeichnet, wobei das leere Wort durch ε repräsentiert wird. Die Länge eines endlichen Wortes w wird durch $|w|$ ausgedrückt, d.h., für ein Wort $w = a_1 \ldots a_n$ gilt $|w| = n$.
Die Menge aller endlichen Wörter über A wird mit A^* bezeichnet. Die Menge aller endlichen Wörter über A, die nicht das leere Wort ε enthält, wird als A^+ geschrieben. A^ω beschreibt die Menge aller unendlichen Wörter (ω-*Wörter*) über A. Eine beliebige Menge von (endlichen oder unendlichen) Wörtern über A wird als *Sprache* bezeichnet. Ein ω-Wort σ kann als Abbildung $\sigma \colon \mathbb{N} \to A$ angesehen werden, d.h., $\sigma = \sigma(0)\sigma(1)\ldots$. Mit σ^i wird der i-te Suffix eines ω-Wortes σ bezeichnet, d.h., $\sigma^i = \sigma(i)\sigma(i+1)\ldots$.

Definition 2.1.11 (Graphen)
Ein Tupel $G = (V, E)$ wird als *Graph* bezeichnet, wobei V eine endliche Menge von Knoten („*vertices*“) und $E \subseteq V \times V$ eine Menge von *Kanten* („*edges*“) bezeichnen. Dabei wird vorausgesetzt, daß E irreflexiv ist. Ein Graph G ist *ungerichtet*, falls E symmetrisch ist, andernfalls ist G *gerichtet*. G heißt *vollständig*, falls $E = V \times V$.
Ein *Pfad* von einem Knoten x zu einem Knoten y in G wird durch eine Folge $v_0 v_1 \ldots v_n$ von Knoten beschrieben, wobei $x = v_0, y = v_n$ und $(v_{i-1}, v_i) \in E$ für $1 \leq i \leq n$ gilt.
Ein ungerichteter Graph G heißt *zusammenhängend*, falls alle Knoten von G paarweise durch einen Pfad verbunden sind. Ein gerichteter Graph heißt *zusammenhängend*, falls der zugrundeliegende ungerichtete Graph zusammenhängend ist. Ein gerichteter Graph G heißt *stark zusammenhängend*, falls alle Knoten von G paarweise durch einen Pfad verbunden sind.
Für einen ungerichteten (gerichteten) Graphen $G = (V, E)$ bezeichnet der Teilgraph $G' = (V', E')$ mit $V' \subseteq V$ und $E' = E \cap (V' \times V')$ eine (*starke*) *Zusammenhangskomponente* von G, falls G' (stark) zusammenhängend ist.

Ein Graph $G = (V, E)$ heißt *k-partit*, falls die Knotenmenge V in k Klassen partitioniert werden kann, sodaß jede Kante von G jeweils in zwei verschiedenen Klassen endet. Im Falle $k = 2$ wird G auch als *bipartiter* Graph bezeichnet.
Für einen ungerichteten Graphen $G = (V, E)$ wird ein Teilgraph $G' = (V', E')$ mit $V' \subseteq V, |V'| = k$ für $k \in \mathbb{N}$ und $E' = E \cap (V' \times V')$ als k-CLIQUE von G bezeichnet, falls G' vollständig ist.

2.2 Petrinetze

Der Formalismus der Petrinetze basiert auf der Arbeit von Petri [Pet62] aus dem Jahre 1962 und entstand mit der Absicht „möglichst viele Erscheinungen bei der Informationsübertragung und Informationswandlung in einheitlicher und exakter Weise beschreiben“ zu können. Seitdem entwickelte sich die Petrinetztheorie zu einem bedeutenden Formalismus für die Modellierung und Verifikation von verteilten Systemen, da sowohl eine begriffliche Trennung zwischen Struktur und Verhalten der Systeme vorliegt, auf die innerhalb von Analysemethoden zurückgeriffen werden kann, als auch eine sehr natürliche Darstellung von Nebenläufigkeit ermöglicht wird.
Ein Petrinetz bildet strukturell gesehen einen bipartiten Graphen, wobei die beiden disjunkten Knotenmengen als *Stellen* und *Transitionen* bezeichnet und graphisch als Kreise bzw. Vierecke dargestellt werden. Die Stellen werden als *lokale Zustände* und die Transitionen als *Aktionen* des Systems interpretiert. Zur Beschreibung des *dynamischen Verhaltens* eines Netzes werden die Stellen mit sogenannten *Marken* versehen, die nach bestimmten Regeln im Netz „umherwandern“ und somit *Zustandsänderungen* des Systems herbeiführen können.
In dieser Arbeit werden zwei Petrinetzklassen betrachtet, zum einen *Stellen-/Transitionen-Netze* [DE95, Rei85], die sehr einfach struktiert sind und lediglich eine „low-level“-Modellierung von Systemen gestatten, und zum anderen *M-Netze* [BFF+95a, BFF+95b], die programmiersprachenähnliche Modellierungsmöglichkeiten aufweisen und somit komfortabler als die Stellen-/Transitionen-Netze zu handhaben sind.

2.2.1 Stellen-/Transitionen-Netze

Stellen-/Transitionen-Netze [DE95, Rei85] stellen die einfachste Form von Petrinetzen dar. Sie weisen eine sehr einfache Struktur auf, und es gibt kaum Varianten. Von daher können sie als eine Art „Assemblersprache“ für die Modellierung verteilter Systeme angesehen werden. Wie bereits erwähnt wurde, kann eine begriffliche Abgrenzung zwischen der Struktur und dem Verhalten eines Petrinetzes vorgenommen werden. Dieses spiegelt sich auch in den folgenden Definitionen wider. Zunächst wird in Definition 2.2.1 auf der nächsten Seite die strukturelle Beschaffenheit von Stellen-/Transitionen-Netzen vorgestellt, und anschließend wird in den Definitionen 2.2.2 auf der nächsten Seite und 2.2.3 auf Seite 30 auf dynamische Aspekte und Eigenschaften eingegangen.

Definition 2.2.1 (S/T-Netz, Vor-/Nachbereich, Schlinge)
Ein 4-Tupel $N = (S, T, F, W)$ wird als *Stellen-/Transitionen-Netz* (*S/T-Netz*) bezeichnet, falls

- $S = (s_1, s_2, \ldots, s_n)$ und $T = (t_1, t_2, \ldots, t_m)$ disjunkte Mengen bezeichnen,
- $F \subseteq (S \times T) \cup (T \times S)$ eine Relation darstellt und
- $W : F \to \mathbb{N}$ eine Abbildung beschreibt.

Die Elemente von S werden als *Stellen* und die Elemente von T als *Transitionen* bezeichnet. Für Stellen und Transitionen wird im allgemeinen auch die Bezeichnung *Knoten* verwendet. F beschreibt die *Flußrelation* des Netzes und wird auch mit ihrer charakteristischen Funktion über der Menge $(S \times T) \cup (T \times S)$ identifiziert. Die *Gewichtsfunktion* W ordnet jeder Kante des Netzes ein *Gewicht* zu.[1]
Für einen Knoten $x \in S \cup T$ wird die Menge

$$^\bullet x = \{y \in S \cup T \mid (y, x) \in F\}$$

als dessen *Vorbereich* und die Menge

$$x^\bullet = \{y \in S \cup T \mid (x, y) \in F\}$$

als dessen *Nachbereich* definiert. Der Vor- und Nachbereich einer Menge von Knoten ergibt sich aus der Vereinigung der Vor- und Nachbereiche der einzelnen Knoten, d.h., für $X \subseteq S \cup T$ gilt:

$$^\bullet X = \bigcup_{x \in X} {}^\bullet x \quad \text{und} \quad X^\bullet = \bigcup_{x \in X} x^\bullet$$

Ein Knoten x besitzt eine *Schlinge* mit Knoten y (und umgekehrt), falls $y \in {}^\bullet x \cap x^\bullet$.

Definition 2.2.2 (Markierungen, Schaltregel, Schaltfolge)
Eine *Markierung* M eines S/T-Netzes $N = (S, T, F, W)$ wird durch eine Abbildung

$$M : S \to \mathbb{N}$$

beschrieben.
Ein Tupel $\Sigma = (N, M_0)$ wird *S/T-Netzsystem mit der Anfangsmarkierung* M_0 genannt, falls N ein S/T-Netz und M_0 eine Markierung von N bezeichnen.
Eine Markierung M *aktiviert* eine Transition t, falls

$$\forall s \in {}^\bullet t : M(s) \geq W((s, t))$$

Aktiviert eine Markierung M keine Transition, so wird M als *tote Markierung* bezeichnet. Eine unter einer Markierung M aktivierte Transition t kann *schalten*, wodurch die

[1] Bei S/T-Netzen mit nur einfachen Kantengewichten wird die Gewichtsfunktion W vollständig durch die Flußrelation F beschrieben. In diesem Fall wird das Netz als Tripel (S, T, F) angegeben.

Markierung M in eine *Folgemarkierung* M' überführt wird. Ein solcher Schaltvorgang wird durch die Notation $M \xrightarrow{t} M'$ ausgedrückt. Dabei berechnet sich die Folgemarkierung M' nach folgender *Schaltregel:*

$$\forall s \in S\colon M'(s) = \begin{cases} M(s) - W((s,t)) & \text{falls } s \in {}^\bullet t \setminus t^\bullet \\ M(s) + W((t,s)) & \text{falls } s \in t^\bullet \setminus {}^\bullet t \\ M(s) - W((s,t)) + W((t,s)) & \text{falls } s \in {}^\bullet t \cap t^\bullet \\ M(s) & \text{sonst} \end{cases}$$

Eine endliche Folge von Transitionen $\sigma = t_1 t_2 \ldots t_n$ wird *endliche Schaltfolge* genannt, falls Markierungen $M_1, M_2, \ldots, M_n$ existieren, sodaß

$$M_{i-1} \xrightarrow{t_i} M_i \quad \text{für} \quad 1 \leq i \leq n.$$

M_n beschreibt die Markierung, die von der Anfangsmarkierung M_0 durch Ausführen der Schaltfolge σ erreicht wird. Notationell wird dieser Vorgang durch $M_0 \xrightarrow{\sigma} M_n$ ausgedrückt. Eine *unendliche Schaltfolge* σ wird in analoger Weise wie eine endliche Schaltfolge definiert und als $M_0 \xrightarrow{\sigma}$ geschrieben.
Die Sprache $L(\Sigma)$ enthält alle endlichen und unendlichen Schaltfolgen von Σ. Die Sprache $L^\omega(\Sigma)$ enthält alle unendlichen Schaltfolgen von Σ.
Eine Markierung M ist *erreichbar*, falls eine Schaltfolge σ existiert, sodaß $M_0 \xrightarrow{\sigma} M$. Die Menge aller von einer Markierung M aus erreichbaren Markierungen ist durch

$$\mathcal{R}(M) = \{M' \mid \exists \sigma\colon M \xrightarrow{\sigma} M'\}$$

gegeben.

Definition 2.2.3 (Verklemmungsfreiheit, Beschränktheit, Sicherheit)
Gegeben sei ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F, W)$. Σ heißt

- *verklemmungsfrei*, falls keine tote Markierung von M_0 aus erreichbar ist.
- *n-beschränkt* (*beschränkt*), falls alle erreichbaren Markierungen $M \in \mathcal{R}(M_0)$ n-beschränkt (beschränkt) sind. Eine Markierung M ist *n-beschränkt* (*beschränkt*), falls gilt:
$$\exists n \in \mathbb{N}\colon \forall s \in S\colon M(s) \leq n$$
- *sicher*, falls es 1-beschränkt ist.

Bemerkung 2.2.1 Eine Markierung M eines sicheren S/T-Netzsystems wird auch mit der Menge $S' \subseteq S$ identifiziert, sodaß

$$\forall s \in S\colon s \in S' \Leftrightarrow M(s) = 1$$

2.2.2 M-Netze

M-Netze [BFF+95a, BFF+95b] gehören zur Klasse der High-Level-Petrinetze [Gen87, Bau90, Jen91, Rei91, Jen92, Jen94, Jen97, Jen98], jedoch besteht der Hauptunterschied zu den traditionellen High-Level-Netzen darin, daß auf M-Netzen kompositionelle Operationen, im speziellen für Synchronisation, definiert sind, die den Mechanismen von Prozeßalgebren wie beispielsweise CCS [Mil80, Mil89] sehr ähnlich sind.
Im folgenden werden M-Netze aber nur mit den Bestandteilen eingeführt, welche für die in dieser Arbeit durchzuführenden Verifikationsaufgaben von wesentlicher Bedeutung sind. Dabei werden die Definitionen in gleicher Weise wie bereits bei den S/T-Netzen in strukturelle Beschaffenheiten sowie dynamische Aspekte und Eigenschaften der M-Netze unterteilt.

Definition 2.2.4 (M-Netz)
Gegeben seien eine endliche Menge V von *Variablen* und eine endliche Menge Z von *Werten*[2] („*Farben*“), wobei $V \cap Z = \emptyset$. Ein 4-Tupel $N = (S, T, F, \iota)$ wird als *M-Netz* bezeichnet, falls

- (S, T, F) ein S/T-Netz beschreibt und
- ι eine Abbildung mit Definitionsbereich $S \cup T \cup F$ bezeichnet, sodaß gilt:
  - (i) $\forall s \in S\colon \iota(s) \subseteq Z$
    Dabei wird $\iota(s)$ auch als *Typ* der Stelle s bezeichnet.
  - (ii) $\forall t \in T\colon \iota(t) \in \mathcal{B}_{exp}(V \cup Z)$
    Dabei wird $\iota(t)$ auch als *Wächter* der Transition t bezeichnet.
  - (iii) $\forall f \in F\colon \iota(f) \in \mathcal{M}_f(V)$
    Dabei bezeichnet $(\iota(f))(v)$ für eine Variable v die Anzahl, wie oft v in der Multimenge $\iota(f)$ vorkommt.

Definition 2.2.5 (Markierungen, Schaltregel, Schaltfolge)
Eine *Markierung* eines M-Netzes $N = (S, T, F, \iota)$ wird durch eine Abbildung

$$M\colon S \to \mathcal{M}_f(Z)$$

mit der Eigenschaft

$$\forall s \in S\ \forall z \in Z\colon z \notin \iota(s) \Rightarrow (M(s))(z) = 0$$

beschrieben. Die Bedingung besagt, daß $M(s)$ für jede Stelle $s \in S$ eine Multimenge über dem Typ $\iota(s)$ von s definiert. Dabei gibt $(M(s))(z)$ die Anzahl an, wie oft der Wert z auf der Stelle s unter der Markierung M vorkommt. Im Falle $M(s) = \emptyset$ weist die Stelle s keinen Wert unter der Markierung M auf.
Ein Tupel $\Upsilon = (N, M_0)$ wird *M-Netzsystem mit der Anfangsmarkierung* M_0 genannt,

[2] Insbesondere sei der spezielle Wert $\bullet$ in Z enthalten.

falls N ein M-Netz und M_0 eine Markierung von N bezeichnen.
Für eine Transition $t \in T$ eines M-Netzsystems $\Upsilon = (N, M_0)$ mit $N = (S, T, F, \iota)$ bezeichne

$$V(t) = \bigcup_{(s,t)\in F} \{v \mid v \in \iota((s,t))\} \bigcup_{(t,s)\in F} \{v \mid v \in \iota((t,s))\} \cup Var(\iota(t))$$

die Menge der Variablen, die in dem Wächter sowie den Beschriftungen der Eingangs- und Ausgangskanten von t vorkommen. Eine Abbildung

$$\varsigma \colon V(t) \to Z$$

heißt *Schaltmodus* einer Transition t, falls die folgenden Bedingungen gelten:

(i) $\forall s \in {}^\bullet t \cup t^\bullet \colon v \in \iota((s,t)) \cup \iota((t,s)) \Rightarrow \varsigma(v) \in \iota(s)$

(ii) $\varsigma \models \iota(t)$

Die Menge der *Schaltinstanzen* eines M-Netzsystems ist durch

$$\mathcal{I}_T = \{(t, \varsigma) \mid t \in T \wedge (\varsigma \text{ ist ein Schaltmodus von } t)\}$$

gegeben.[3] Eine Markierung M *aktiviert* eine Transition t, falls es einen Schaltmodus ς von t gibt, sodaß

$$\forall s \in {}^\bullet t \colon \iota((s,t))[\varsigma] \subseteq M(s),$$

wobei $\iota((s,t))[\varsigma]$ die Multimenge von Werten beschreibt, die aus der Multimenge $\iota((s,t))$ durch Substitution jedes Elementes $v \in \iota((s,t))$ mit dem Wert $\varsigma(v)$ hervorgeht. Eine Markierung M heißt *tot*, falls sie keine Transition aktiviert.
Eine unter einer Markierung M aktivierte Transition t kann *schalten*, wodurch die Markierung M in eine *Folgemarkierung* M' überführt wird. Ein solcher Schaltvorgang wird duch die Notation $M \xrightarrow{t^\varsigma} M'$ ausgedrückt. Dabei berechnet sich die Folgemarkierung M' nach folgender *Schaltregel*:

$$\forall s \in S \colon M'(s) = \begin{cases} M(s) \setminus \iota((s,t))[\varsigma] & \text{falls } s \in {}^\bullet t \setminus t^\bullet \\ M(s) + \iota((t,s))[\varsigma] & \text{falls } s \in t^\bullet \setminus {}^\bullet t \\ M(s) \setminus \iota((s,t))[\varsigma] + \iota((t,s))[\varsigma] & \text{falls } s \in {}^\bullet t \cap t^\bullet \\ M(s) & \text{sonst} \end{cases}$$

Eine endliche Folge $\sigma = t_1^{\varsigma_1} t_2^{\varsigma_2} \ldots t_n^{\varsigma_n}$ von Schaltinstanzen wird *endliche Schaltfolge* genannt, falls Markierungen $M_1, M_2, \ldots, M_n$ existieren, sodaß

$$M_{i-1} \xrightarrow{t_i^{\varsigma_i}} M_i \quad \text{für} \quad 1 \leq i \leq n.$$

[3] Eine Schaltinstanz (t, ς) wird alternativ auch t^ς geschrieben.

M_n beschreibt die Markierung, die von der Anfangsmarkierung M_0 durch Ausführen der Schaltfolge σ erreicht wird. Notationell wird dieser Vorgang durch $M_0 \xrightarrow{\sigma} M_n$ ausgedrückt. Eine *unendliche Schaltfolge* σ wird in analoger Weise zur endlichen Schaltfolge definiert und als $M_0 \xrightarrow{\sigma}$ geschrieben.
Die Sprache $L(\Upsilon)$ enthält alle endlichen und unendlichen Schaltfolgen von Υ. Die Sprache $L^{\omega}(\Upsilon)$ enthält alle unendlichen Schaltfolgen von Υ.
Eine Markierung M ist *erreichbar*, falls eine Schaltfolge σ existiert, sodaß $M_0 \xrightarrow{\sigma} M$. Die Menge aller von einer Markierung M aus erreichbaren Markierungen ist durch

$$\mathcal{R}(M) = \{M' \mid \exists \sigma : M \xrightarrow{\sigma} M'\}$$

gegeben.

Definition 2.2.6 (Verklemmungsfreiheit, Beschränktheit, Sicherheit)
Gegeben sei ein M-Netzsystem $\Upsilon = (N, M_0)$ mit $N = (S, T, F, \iota)$. Υ heißt

- *verklemmungsfrei*, falls keine tote Markierung von M_0 aus erreichbar ist.
- *n-beschränkt* (*beschränkt*), falls alle erreichbaren Markierungen $M \in \mathcal{R}(M_0)$ n-beschränkt (beschränkt) sind. Eine Markierung M ist *n-beschränkt* (*beschränkt*), falls gilt:
$$\exists n \in \mathbb{N} \colon \forall s \in S \; \forall z \in Z \colon (M(s))(z) \leq n$$
- *streng n-beschränkt* (*streng beschränkt*), falls alle erreichbaren Markierungen $M \in \mathcal{R}(M_0)$ streng n-beschränkt (streng beschränkt) sind. Eine Markierung M ist *streng n-beschränkt* (*streng beschränkt*), falls gilt:
$$\exists n \in \mathbb{N} \colon \forall s \in S \colon |M(s)| \leq n$$
- *(streng) sicher*, falls es (streng) 1-beschränkt ist.

Bemerkung 2.2.2 Eine Markierung M eines streng sicheren M-Netzsystems Υ wird alternativ auch mit der Menge $M' \subseteq \{(s, z) \mid s \in S \wedge z \in \iota(s)\} = \mathcal{I}_S$ identifiziert, sodaß

$$\forall s \in S \; \forall z \in Z \colon (s, z) \in M' \Leftrightarrow z \in M(s)$$

Dabei repräsentiert $\mathcal{I}_S$ die Menge der *Stelleninstanzen*[4] von Υ.

[4]Eine Stelleninstanz (s, z) wird alternativ auch s^z geschrieben.

Kapitel 3

Erreichbarkeitsanalyse mittels Netzentfaltungen

Eines der Hauptprobleme auf dem Gebiet der automatischen Verifikation beschäftigt sich mit der Verifikation von Sicherheitseigenschaften, also der Zusicherung, daß eine bestimmte Eigenschaft eines Systems immer gewährleistet ist oder eine spezielle Situation niemals eintreten kann. Ein typisches Beispiel hierfür liefert das Kriterium zum wechselseitigen Ausschluß („*mutual-exclusion*“) zweier oder mehrerer Prozesse, die gleichzeitig auf dasselbe Betriebsmittel (beispielsweise eine Druckerwarteschlange) zugreifen wollen.[1] Hierbei muß sichergestellt werden, daß niemals zwei oder mehrere Prozesse gleichzeitig einen Auftrag in die Warteschlange einfügen können. Diese Sicherheitseigenschaft kann aber auf eine einfache Erreichbarkeitseigenschaft, also die Fragestellung, ob ein bestimmter Systemzustand eintreten kann, zurückgeführt werden, indem überprüft wird, ob das System in einen Zustand gelangen kann, sodaß zwei oder mehrere Prozesse gleichzeitig auf die Warteschlange zugreifen können. In diesem Fall liefert die Gültigkeit der Erreichbarkeitseigenschaft zugleich ein Gegenbeispiel, welches die Sicherheitseigenschaft verletzt. Auf diese Weise können die meisten Sicherheitseigenschaften eines Systems auf einfache Erreichbarkeitseigenschaften zurückgeführt werden, weshalb die Verifikation von Erreichbarkeitsfragen eine zentrale Stellung innerhalb des Gebiets der automatischen Verifikation einnimmt.
Systeme können in vielen verschiedenen Formalismen beschrieben werden, unter anderem mit Automaten, die durch Rendezvous oder Kanäle endlicher Kapazität miteinander kommunizieren, synchronen Produkten von Transitionssystemen [ER99, Röm00] oder sicheren S/T-Netzsystemen [DE95, Rei85]. Alle diese Modelle weisen jedoch dieselbe Mächtigkeit auf, und das Erreichbarkeitsproblem für mit diesen Formalismen beschrie-

[1] Eine ausführliche Beschreibung solcher Algorithmen findet sich in [Ray86].

bene Systeme ist PSPACE-vollständig [CEP95].
In diesem Kapitel werden ausschließlich Modellierungen mit sicheren S/T-Netzsystemen betrachtet. Das *Erreichbarkeitsproblem* soll dann als die folgende Fragestellung verstanden werden:

> *Gegeben sei eine Menge S' von Stellen. Existiert eine von der Anfangsmarkierung aus erreichbare Markierung M, die auf mindestens jede Stelle von S' eine Marke legt.*

Dabei bleibt das Erreichbarkeitsproblem auch für den Fall PSPACE-vollständig, daß S' nur aus einer Stelle besteht.
Bereits in der Vergangenheit sind Halbordnungstechniken, und hier insbesondere die von McMillan eingeführten Netzentfaltungen [McM92], sehr erfolgreich zur Überprüfung von Verklemmungsfreiheit sicherer Netzsysteme eingesetzt worden [McM92, MR97, Mel98, Hel99, Hel02, Kho03]. Dabei wird das Netzsystem zunächst in ein azyklisches Netz „entfaltet“, bis ein endliches, vollständiges Präfix vorliegt, welches zwei wichtige Eigenschaften aufweist [McM92, ERV02, Röm00]:

> *Das Präfix ist endlich* und *kodiert dieselben erreichbaren Markierungen wie das ursprüngliche Netzsystem.*

Anschließend können verschiedene Verfahren zur Überprüfung von Verklemmungsfreiheit auf das Präfix angewendet werden:

- Ein Branch-and-Bound-Algorithmus von McMillan [McM92],
- ein von Melzer und Römer entwickelter Algorithmus, der auf Techniken der linearen Programmierung zurückgreift [MR97, Mel98], und
- ein Algorithmus von Heljanko, der zur Lösung des Problems einen SAT-Solver verwendet [Hel99, Hel02].

Eine Vergleichsstudie dieser Verfahren hat ergeben, daß die auf dem SAT-Solver basierende Methode den anderen in den meisten Beispielen überlegen ist. Entsprechend soll in diesem Kapitel eine Studie für das Erreichbarkeitsproblem durchgeführt werden.
Wie bereits erwähnt wurde, ist das Erreichbarkeitsproblem für sichere S/T-Netzsysteme PSPACE-vollständig in der Größe der S/T-Netze [CEP95], jedoch kann es in ein (möglicherweise exponentiell größeres) NP-vollständiges Problem überführt werden, sofern endliche, vollständige Präfixe der Netzsysteme als Eingangsgrößen vorliegen [ES01b]. Daher wird zunächst gezeigt, daß das Erreichbarkeitsproblem für sichere S/T-Netzsysteme NP-vollständig in der Größe deren Präfixe ist, sofern die Präfixe als Eingangsgrößen vorliegen.[2] Anschließend werden verschiedene Algorithmen zur Lösung des Erreichbarkeitsproblems betrachtet:

[2] Die Überprüfung von Verklemmungsfreiheit ist ebenfalls NP-vollständig in der Größe des Präfixes [McM92].

- Das in [ES01b] vorgestellte graphentheoretische Verfahren führt das Erreichbarkeitsproblem auf das CLIQUE-Problem zurück. Ideen dazu sind in gewisser Weise implizit in [Mel98, KKTT96] vorhanden, aber es liegen keine Erkenntnisse darüber vor, daß ein derartiger Algorithmus jemals zuvor explizit ausformuliert oder implementiert wurde.
- Melzer hat sein Verfahren zur Überprüfung von Verklemmungsfreiheit unter Einsatz linearer Programmierungstechniken auf das Erreichbarkeitsproblem erweitert [Mel98].
- Heljanko hat einen Algorithmus beschrieben [Hel99, Hel02], der Erreichbarkeitsfragen unter Verwendung eines SAT-Solvers löst.
- McMillan hat einen „On-the-Fly"-Algorithmus skizziert [McM92], über den jedoch auch keine Erkenntnisse darüber vorliegen, daß er jemals zuvor implementiert wurde.

Während die Algorithmen von Melzer und Heljanko exponentielle Laufzeit in der Größe des endlichen, vollständigen Präfixes aufweisen, löst der graphentheoretische Algorithmus das Erreichbarkeitsproblem in Zeit $\mathcal{O}(|B|^k)$, wobei $|B|$ die Anzahl der Bedingungen des Präfixes und k die Kardinalität der Stellenmenge beschreibt, deren Erreichbarkeit nachgewiesen werden soll. Da $|B|$ normalerweise sehr viel größer als k ist, stellt dieser Algorithmus eine Verbesserung bezüglich der Komplexität dar.[3]
Die Algorithmen für das CLIQUE-Problem und die „On-the-Fly"-Verifikation wurden implementiert und anhand von Testreihen mit den anderen Methoden verglichen. Aus den Ergebnissen soll eine Aussage abgeleitet werden, welche der obigen Verfahren für die Erreichbarkeitsanalyse von Teilmarkierungen am geeignetsten zu sein scheinen.

3.1 S/T-Netzentfaltungen

Die eben angesprochenen Verfahren weisen die Gemeinsamkeit auf, daß sie Netzentfaltungen, genauer gesagt endliche, vollständige Präfixe, als Eingabe verwenden. Ursprünglich sind Netzentfaltungen als Halbordnungssemantik für Petrinetze eingeführt worden [NPW80]. Später sind sie dann von Engelfriet unter dem Namen Branching-Prozesse intensiver studiert worden [Eng91]. McMillan war jedoch der erste, der überhaupt eine Verifikationsmethode auf der Basis von Netzentfaltungen vorgeschlagen hat [McM92]. Diese dient zur Erkennung von Verklemmungen in sicheren S/T-Netzsystemen.
Im folgenden werden nun die Definitionen und Konzepte eingeführt, die zum Verständnis von entfaltungsbasierten Verifikationsmethoden benötigt werden.

[3] Im Gegensatz zu dem graphentheoretischen Algorithmus gehen die Verfahren von Melzer und Heljanko nicht davon aus, daß die Co-Relation des endlichen, vollständigen Präfixes vorliegt, weshalb die Komplexitätsschranke des graphentheoretischen Algorithmus' besser wird. Es wäre jedoch möglich, diese Information so in die anderen Algorithmen einfließen zu lassen, daß sie auf dieselbe obere Schranke für die Laufzeit kommen.

Definition 3.1.1 (Konfliktrelation)
Gegeben seien ein S/T-Netz (S, T, F) und zwei Knoten $x, y \in S \cup T$. Die Knoten x und y stehen in *Konflikt* $(x \# y)$, falls es zwei verschiedene Transitionen $t_1, t_2 \in T$ gibt, sodaß ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ und $(t_1, x), (t_2, y)$ zu der reflexiven, transitiven Hülle von F gehören. Ein Knoten $x \in S \cup T$ befindet sich in *Selbstkonflikt*, falls $x \# x$.

Anschaulich gesehen besagt die Konfliktrelation, daß es für in Konflikt stehende Knoten x und y zwei Pfade gibt, die aus derselben Stelle entspringen, sich dort direkt an den Transitionen t_1 und t_2 verzweigen und zu den Knoten x und y führen. Dabei können sich die Pfade zwischenzeitlich auch wieder kreuzen.
Unter Verwendung der Konfliktrelation werden nun sogenannte Occurrence-Netze definiert, die als Vorstufe zu den von Engelfriet betrachteten Branching-Prozessen angesehen werden können.

Definition 3.1.2 (Occurrence-Netz [NPW80])
Ein S/T-Netz $N' = (B, E, F')$ wird als *Occurrence-Netz* bezeichnet (wobei die Stellen jetzt *Bedingungen* und die Transitionen *Ereignisse* genannt werden), falls die folgenden Eigenschaften erfüllt sind:

- jede Bedingung hat maximal ein Ereignis im Vorbereich, d.h.,

$$\forall b \in B\colon |{}^\bullet b| \leq 1$$

- die Flußrelation F' ist azyklisch, d.h., die (irreflexive) transitive Hülle von F' bildet eine Halbordnung $<$ in der Menge der Knoten von N'. Die Halbordnung $<$ wird auch als *Kausalrelation* bezeichnet.

- jeder Knoten von N' hat nur endlich viele Vorgängerknoten, d.h.,

$$\text{für jeden Knoten } x \in B \cup E \text{ ist die Menge } \{y \in B \cup E \mid y < x\} \text{ endlich.}$$

- kein Ereignis befindet sich in Selbstkonflikt, d.h.,

$$\forall e \in E\colon \neg(x \# x)$$

Die *Anfangsmarkierung* eines Occurrence-Netzes N', unter der genau die Bedingungen eine Marke enthalten, welche minimal bezüglich der Kausalrelation $<$ sind, wird mit $Min(N')$ bezeichnet.

Mit Hilfe der Konflikt- und Kausalrelationen kann nun eine dritte Relation, die sogenannte Co-Relation, definiert werden, die sich aus der strukturellen Eigenheit eines Occurrence-Netzes ergibt. Sie bildet die Grundlage für das in [ES01b] beschriebene graphentheoretische Verfahren.

Definition 3.1.3 (Co-Relation, Co-Menge [Eng91, BF88])
Gegeben sei ein Occurrence-Netz (B, E, F'). Die *Co-Relation* $co \subseteq B \times B$ ist definiert durch

$$(b_1, b_2) \in co \Leftrightarrow (b_1 \not< b_2 \wedge b_2 \not< b_1 \wedge \neg(b_1 \# b_2)),$$

d.h., zwei Bedingungen sind *nebenläufig* („*concurrent*“), wenn sie weder kausal voneinander abhängig sind, noch in Konflikt stehen. Eine Teilmenge $B' \subseteq B$ von Bedingungen heißt *Co-Menge*, falls alle Elemente aus B' paarweise nebenläufig sind.

Ein Branching-Prozeß eines S/T-Netzsystems wird als beschriftetes Occurrence-Netz verstanden, dessen Bedingungen und Ereignisse mit Stellen und Transitionen des S/T-Netzes annotiert sind. Für die Beschriftung der Bedingungen und Ereignisse wird das folgende Schema verwendet, das von Engelfriet eingeführt wurde [Eng91].

Definition 3.1.4 (Mengen $\mathcal{B}$ und $\mathcal{E}$ von Bezeichnern [Eng91])
Gegeben sei ein S/T-Netz (S, T, F). Die Mengen $\mathcal{B}$ und $\mathcal{E}$ von Bezeichnern sind wie folgt induktiv definiert:

- $\perp \in \mathcal{E}$, wobei $\perp$ ein spezielles Symbol darstellt;
- falls $e \in \mathcal{E}$, dann $(s, e) \in \mathcal{B}$ für jedes $s \in S$;
- falls $\emptyset \subset B' \subseteq \mathcal{B}$, dann $(t, B') \in \mathcal{E}$ für jedes $t \in T$.

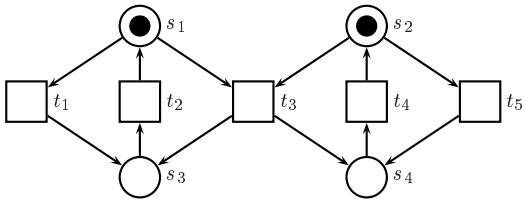
Ein Branching-Prozeß eines S/T-Netzsystems (N, M_0) wird nun mittels zweier Teilmengen $B \subseteq \mathcal{B}$ und $E \subseteq \mathcal{E}$ dieser Bezeichner definiert. Dabei werden die Bezeichner der Mengen $\mathcal{B}$ und $\mathcal{E}$ folgendermaßen verwendet: Die minimalen Bedingungen (die Anfangsmarkierung) des Branching-Prozesses werden (wird) durch die Bezeichner $(s, \perp) \in B$ beschrieben, für die $M_0(s) = 1$. Die Beschriftung einer Bedingung $(s, e) \in B$ lautet s, und e stellt deren einziges Eingangsereignis dar. Entsprechendes gilt für ein Ereignis $(t, B') \in E$, das einen Repräsentanten für die Transition t darstellt und dessen Vorbereich aus der Menge B' von Bedingungen besteht.
Mit dieser Notation können Branching-Prozesse vollständig durch ihre Mengen von Bedingungen und Ereignissen als Tupel (B, E) mit $B \subseteq \mathcal{B}$ und $E \subseteq \mathcal{E}$ beschrieben werden.

Definition 3.1.5 (Branching-Prozeß [Eng91])
Gegeben sei ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, der 1-beschränkten Anfangsmarkierung $M_0 = \{s_1, \ldots, s_n\}$ und einfachen Kantengewichten. Die Menge der endlichen *Branching-Prozesse* von Σ ist induktiv definiert durch:

- $(\{(s_1, \perp), \ldots, (s_n, \perp)\}, \emptyset)$ ist ein Branching-Prozeß von Σ.
- Falls (B, E) ein Branching-Prozeß von Σ ist, $t \in T$ eine Transition und $B' \subseteq B$ eine Co-Menge bezeichnen, sodaß in B' für jedes $s \in {}^\bullet t$ genau ein mit s beschriftetes Element vorkommt, dann stellt auch

$$(B \cup \{(s', e) \mid s' \in t^\bullet\}, E \cup \{e\}) \quad \text{mit} \quad e = (t, B')$$

Abbildung 3.1 *Sicheres S/T-Netzsystem*



einen Branching-Prozeß von Σ dar. Das Ereignis e wird als *mögliche Erweiterung* von (B, E) bezeichnet, falls $e \notin E$.

Die Menge aller Branching-Prozesse eines S/T-Netzsystems erhält man durch die Vereinigung beliebiger endlicher oder unendlicher Mengen von Branching-Prozessen, da diese unter komponentenweiser Vereinigung von Bedingungen und Ereignissen abgeschlossen sind. Zudem garantiert die Abgeschlossenheit unter Vereinigung die Existenz eines eindeutigen, maximalen Branching-Prozesses, welcher als *Entfaltung* des S/T-Netzsystems bezeichnet wird [Eng91].

Bemerkung 3.1.1 Aus Gründen der Übersichtlichkeit wird im weiteren Verlauf der Arbeit für Bedingungen b anstatt (s, e) und für Ereignisse e anstatt (t, B') geschrieben. In diesem Fall werden die Beschriftungen für Bedingungen und Ereignisse durch die Abbildung $\lambda\colon (B \cup E) \to (S \cup T)$ beschrieben, wobei

$$\lambda(b) = s, \quad \text{falls } b = (s, e) \in B$$
$$\lambda(e) = t, \quad \text{falls } e = (t, B') \in E$$

Die Abbildung λ kann auf beliebige Wörter über der Knotenmenge eines Branching-Prozesses erweitert werden, d.h., $\lambda\colon (B \cup E)^+ \to (S \cup T)^+$ mit

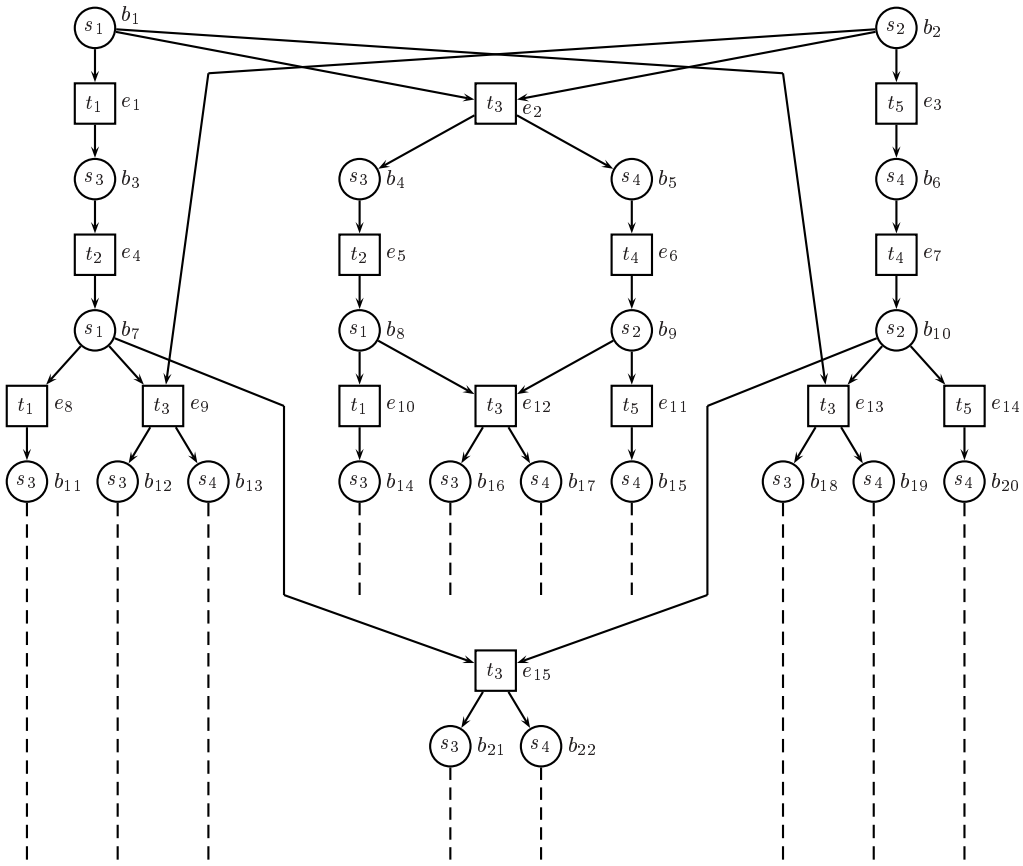
$$\lambda(x_1 x_2 \dots) = \lambda(x_1)\lambda(x_2)\dots \quad \text{für } x_1 x_2 \dots \in (B \cup E)^+$$

Desweiteren wird ein Branching-Prozeß (B, E) auch mit dem beschrifteten Occurrence-Netzsystem $(N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ identifiziert, wobei sich die Flußrelation F' folgendermaßen ergibt:

- $(b, e) \in F'$, falls $e = (t, B') \in E$ und $b \in B' \cap B$
- $(e, b) \in F'$, falls $b = (s, e) \in B$ und $e \in E$

Abbildung 3.1 zeigt ein sicheres S/T-Netzsystem Σ. Ein Anfangsstück der nach Definition 3.1.5 auf der vorherigen Seite erhaltenen unendlichen Entfaltung von Σ ist in Abbildung 3.2 auf der nächsten Seite dargestellt. Dabei wird die Beschriftung λ durch

Abbildung 3.2 *Anfangsstück der Entfaltung für das Netzsystem aus Abbildung 3.1 auf der vorherigen Seite*



die innerhalb der Bedingungen und Ereignisse stehenden Stellen und Transitionen angegeben. Die Entfaltung soll als Halbordnungssemantik für S/T-Netzsysteme verstanden werden. Von daher muß gewährleistet sein, daß sich alle Abläufe und erreichbaren Markierungen eines S/T-Netzsystems in dessen Entfaltung widerspiegeln, und umgekehrt. Um diesen Sachverhalt formal beschreiben zu können, werden jedoch noch einige Definitionen benötigt.

Definition 3.1.6 (Konfigurationen, Schnitte)
Gegeben sei ein Branching-Prozeß $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$. Eine Teilmenge $C \subseteq E$ von Ereignissen wird als *Konfiguration* von β bezeichnet, falls sie die folgenden Eigenschaften erfüllt:

- C ist kausal abgeschlossen, d.h.,

$$e \in C \Rightarrow \forall e' \leq e : e' \in C$$

- C ist konfliktfrei, d.h.,
$$\forall e, e' \in C\colon \neg(e \# e')$$

Dementsprechend beschreibt die *lokale Konfiguration* $[e]$ eines Ereignisses $e \in E$ die Menge aller kausalen Vorgängerereignisse von e, d.h.,

$$[e] = \{e' \in E \mid e' \leq e\}$$

Eine beliebige Folge σ aller Elemente einer Konfiguration C wird als *Linearisierung von C* bezeichnet, falls σ eine Schaltfolge von β repräsentiert.[4] Eine Co-Menge $B' \subseteq B$, die maximal bezüglich Mengeninklusion ist, wird als *Schnitt* von β bezeichnet. Für eine endliche Konfiguration C definiert die Menge

$$Cut(C) = (Min(N') \cup C^\bullet) \setminus {}^\bullet C$$

einen solchen Schnitt. Die Menge von Stellen

$$Mark(C) = \{\lambda(b) \mid b \in Cut(C)\}$$

definiert die entsprechende Markierung in dem S/T-Netzsystem, dessen Semantik durch β gegeben ist.

Mit Hilfe dieser Definitionen ist es nun möglich, formal eine Beziehung zwischen den Abläufen und erreichbaren Markierungen eines Netzsystems und dessen Entfaltung herzustellen.

Proposition 3.1.1 (Eigenschaft einer Entfaltung [Eng91])
Gegeben seien ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und dessen Entfaltung $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$. Für jede Schaltfolge $M_0 \xrightarrow{\sigma} M$ von Σ gibt es eine Konfiguration C von β, sodaß gilt:

- $Min(N') \xrightarrow{\sigma'} Cut(C)$
- $Mark(C) = M$
- $\lambda(\sigma') = \sigma$

und umgekehrt, wobei σ' eine Linearisierung von C darstellt.

Dieses Resultat soll nun anhand des Netzsystems Σ aus Abbildung 3.1 auf Seite 40 und dessen Entfaltung β (Abbildung 3.2 auf der vorherigen Seite) erläutert werden. Zum Beispiel gibt es für den Ablauf

$$M_0 = \{s_1, s_2\} \xrightarrow{t_3 t_2 t_4 t_3} \{s_3, s_4\} = M$$

von Σ die Konfiguration $C = \{e_2, e_5, e_6, e_{12}\}$ in β, sodaß mit der Linearisierung $\sigma' = e_2 e_5 e_6 e_{12}$ von C gilt:

[4] Für eine Konfiguration kann es mehrere Linearisierungen geben. Beispielsweise besitzt die Konfiguration $\{e_1, e_3, e_4\}$ der Entfaltung aus Abbildung 3.2 auf der vorherigen Seite die Linearisierungen $e_1 e_4 e_3$, $e_1 e_3 e_4$ und $e_3 e_1 e_4$.

- $Min(N') = \{b_1, b_2\} \xrightarrow{e_2 e_5 e_6 e_{12}} \{b_{16}, b_{17}\} = Cut(C)$
- $Mark(C) = \{s_3, s_4\} = M$
- $\lambda(e_2 e_5 e_6 e_{12}) = \lambda(e_2)\lambda(e_5)\lambda(e_6)\lambda(e_{12}) = t_3 t_2 t_4 t_3$

Nun ergibt sich das Problem, daß die Entfaltung des Netzsystems aus Abbildung 3.1 auf Seite 40 unendlich ist und somit für automatische Verifikationszwecke nicht verwendet werden kann. Da das Netzsystem allerdings eine endliche Anzahl von erreichbaren Markierungen aufweist, muß es in der Entfaltung Teilnetze geben, die mehrmals (möglicherweise unendlich oft) vorkommen. Die sich wiederholenden Teilnetze liefern jedoch keine neuen Informationen mehr bezüglich des Systemverhaltens, da sie bereits durch das erstmalige Vorkommen der Teilnetze vollständig in der Entfaltung kodiert sind. McMillan hat in seiner Arbeit [McM92] eine Möglichkeit zur Erkennung sich wiederholender Teilnetze vorgeschlagen. Mittels dieser Methodik können solche „überflüssigen" Teilnetze ausgeblendet werden, und als Ergebnis erhält man ein endliches, vollständiges Präfix der Entfaltung, das alle zu Verifikationszwecken notwendigen Informationen enthält. Das Präfix spiegelt möglicherweise nicht mehr alle im Netzsystem vorhandenen Abläufe wider, aber es ist zumindest in dem Sinne vollständig, daß es alle erreichbaren Markierungen repräsentiert. Den Kernpunkt von McMillans Betrachtungen bilden sogenannte *Terminalereignisse* („*cut-off events*"). Das bedeutet, daß die Zukunft eines Ereignisses e nicht weiter betrachtet werden muß, falls es bereits ein korrespondierendes Ereignis e' in der Entfaltung gibt, sodaß die durch die lokalen Konfigurationen $[e]$ und $[e']$ induzierten Markierungen gleich sind, d.h.,

$$Mark([e]) = Mark([e']).$$

Das Ereignis e wird in diesem Fall als *Terminalereignis* bezeichnet. Um zu entscheiden, ob zu einem Ereignis bereits ein korrespondierendes Ereignis in der Entfaltung existiert, muß die Möglichkeit bestehen, die Ereignisse (genauer gesagt deren lokale Konfigurationen) miteinander zu vergleichen. Dies kann mittels einer adäquaten Ordnung in der Menge der lokalen Konfigurationen einer Entfaltung geschehen.
Um den Begriff einer adäquaten Ordnung verständlich einführen zu können, muß zunächst noch auf weitere Eigenschaften von Branching-Prozessen eingegangen werden.

Definition 3.1.7 (Suffix $\uparrow C$ einer Konfiguration [ERV02, Röm00])
Gegeben seien ein Branching-Prozeß $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ und eine Konfiguration C von β. Das eindeutige Teilnetz $\uparrow C$ von β, dessen Knoten aus den Elementen der Menge

$$\{x \in B \cup E \mid x \notin C \cup {}^\bullet C \wedge \forall y \in C \colon \neg(x \# y)\}$$

bestehen, wird als *Suffix* von C bezeichnet.

Das Suffix $\uparrow C$ einer Konfiguration C von β bezeichnet also den Teil von β, der „hinter" der Konfiguration C liegt. Als direkte Konsequenz aus Definition 3.1.7 ergibt sich Proposition 3.1.2 auf der nächsten Seite.

Proposition 3.1.2 (Eigenschaften von $\uparrow C$ [ERV02, Röm00])
Gegeben sei ein S/T-Netzsystem $\Sigma = (N, M_0)$. Bezeichnen β einen Branching-Prozeß von Σ und C eine Konfiguration von β, dann stellt das Suffix $\uparrow C$ der Konfiguration C einen Branching-Prozeß des Netzsystems $(N, Mark(C))$ dar.
Bezeichnet β darüberhinaus die Entfaltung von Σ, dann stellt das Suffix $\uparrow C$ der Konfiguration C die bis auf Isomorphie eindeutige Entfaltung des Netzsystems $(N, Mark(C))$ dar.

Für eine Konfiguration C eines Branching-Prozesses β wird mit $C \oplus E$ der Sachverhalt ausgedrückt, daß $C \cup E$ eine Konfiguration von β beschreibt und zudem $C \cap E = \emptyset$ gilt. $C \oplus E$ wird dann als *Erweiterung* der Konfiguration C bezeichnet.
Stellen nun C_1 und C_2 Konfigurationen der Entfaltung β des S/T-Netzsystems (N, M_0) dar, deren Schnitte dieselbe Markierung induzieren, d.h.,

$$\exists M \in \mathcal{R}(M_0) \colon Mark(C_1) = M = Mark(C_2),$$

dann folgt aus Proposition 3.1.2 die Isomorphie von $\uparrow C_1$ und $\uparrow C_2$ zu dem entsprechenden Ausschnitt von β. Folglich gibt es auch einen Isomorphismus f zwischen $\uparrow C_1$ und $\uparrow C_2$. Dieser induziert eine Abbildung von der Menge der endlichen Erweiterungen von C_1 in die Menge der endlichen Erweiterungen von C_2 mit der Abbildungsvorschrift

$$C_1 \oplus E \mapsto C_2 \oplus f(E)$$

Definition 3.1.8 (Adäquate Halbordnung [ERV96, ERV02])
Gegeben sei ein Branching-Prozeß β eines S/T-Netzsystems. Eine Halbordnung $\prec_H$ in der Menge der endlichen Konfigurationen von β heißt *adäquat*, falls gilt:

- $\prec_H$ ist fundiert
- $C_1 \subset C_2$ impliziert $C_1 \prec_H C_2$
- $\prec_H$ bleibt in der Menge der endlichen Erweiterungen von C_1 und C_2 bewahrt, d.h., aus $C_1 \prec_H C_2$ und $Mark(C_1) = Mark(C_2)$ folgt

  $$C_1 \oplus E \prec_H C_2 \oplus f(E)$$

  für alle endlichen Erweiterungen $C_1 \oplus E$ von C_1.

McMillan [McM92] hat eine Halbordnung $\prec_{McM}$ in der Menge der lokalen Konfigurationen der Entfaltung vorgeschlagen, welche diese bezüglich ihrer Kardinalität vergleicht, d.h., für zwei lokale Konfigurationen $[e]$ und $[e']$ gilt

$$[e] \prec_{McM} [e'] \Leftrightarrow |[e]| < |[e']|$$

Ein Nachteil bei Verwendung dieser Halbordnung besteht darin, daß manchmal ein viel größeres Präfix erzeugt wird als eigentlich nötig ist. In einigen Fällen wird ein Präfix der Größenordnung $\mathcal{O}(2^n)$ erzeugt, wohingegen ein minimales endliches, vollständiges

Präfix in der Größenordnung $\mathcal{O}(n)$ (bezüglich des Netzsystems) liegt. Später haben dann Esparza, Römer und Vogler [ERV96, ERV02, Röm00] eine adäquate Ordnung $\prec$ definiert und gezeigt, daß die mit dieser Ordnung erzeugten endlichen, vollständigen Präfixe höchstens die Größe der von McMillan erhaltenen Präfixe aufweisen, meistens jedoch kleiner sind. Desweiteren haben sie gezeigt, daß es immer möglich ist, für ein sicheres S/T-Netzsystem ein endliches, vollständiges Präfix zu erzeugen, das höchstens soviele Nicht-Terminalereignisse besitzt wie das Netzsystem erreichbare Markierungen aufweist.

Definition 3.1.9 (Terminalereignis [ERV02, Röm00])
Gegeben seien ein Branching-Prozeß $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ und eine adäquate Ordnung $\prec$ in der Menge der lokalen Konfigurationen von β. Ein Ereignis $e \in E$ heißt *Terminalereignis*, falls es ein Ereignis $e' \in E$ gibt, sodaß gilt:

- $[e'] \prec [e]$ und
- $Mark([e]) = Mark([e'])$

In diesem Fall wird e' das *korrespondierende Ereignis* von e genannt. Die Menge aller Terminalereignisse wird als $\mathit{CutOffs}(E)$ bezeichnet.

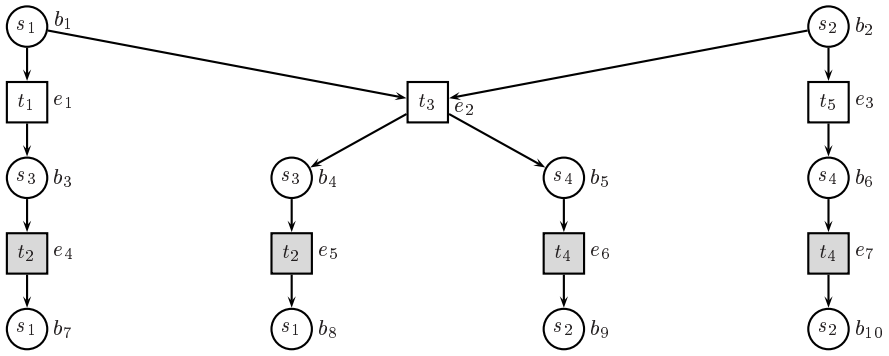
Definition 3.1.10 (Endliches, vollständiges Präfix [ERV02, Röm00])
Gegeben sei ein beschränktes S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Ein Branching-Prozeß $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ heißt genau dann *endliches, vollständiges Präfix* von Σ, wenn es für jede erreichbare Markierung $M \in \mathcal{R}(M_0)$ eine Konfiguration $C \subseteq E \setminus \mathit{CutOffs}(E)$ von β gibt, sodaß gilt:

- $Mark(C) = M$ und
- für jede unter M aktivierte Transition $t \in T$ gibt es eine Konfiguration $C \cup \{e\}$, sodaß gilt:
  - $e \notin C$ und
  - $\lambda(e) = t$

Definition 3.1.10 besagt, daß jede erreichbare Markierung eines Netzsystems in dessen endlichem, vollständigen Präfix repräsentiert ist und durch die Ausführung einer Schaltfolge ohne Terminalereignisse erreicht werden kann. Ein endliches, vollständiges Präfix enthält also die gesamte Information über den Erreichbarkeitsgraphen[5] eines Netzsystems und kann deshalb als symbolische Repräsentation des Erreichbarkeitsgraphen verstanden werden. Allerdings kann das Präfix manchmal exponentiell kleiner als der Erreichbarkeitsgraph sein, weshalb die Entfaltungsmethode einen vielversprechenden Ansatz für automatische Verifikationszwecke darstellt.

[5]Der Erreichbarkeitsgraph eines S/T-Netzsystems (N, M_0) ist definiert als ein gerichteter Graph $(\mathcal{R}(M_0), E)$, wobei $(M, M') \in E \Leftrightarrow \exists t \in T\colon M \xrightarrow{t} M'$ [Sta90].

Abbildung 3.3 *Endliches, vollständiges Präfix für das Netzsystem aus Abbildung 3.1 auf Seite 40, wobei die Terminalereignisse grau unterlegt sind*



Abschließend soll nun anhand des Netzsystems aus Abbildung 3.1 auf Seite 40 gezeigt werden, wie dessen endliches, vollständiges Präfix (Abbildung 3.3, wobei die Terminalereignisse grau unterlegt sind) gebildet werden kann. Das virtuelle Ereignis $\perp$ induziert die Anfangsmarkierung $Min(N')$, und deshalb besteht der Branching-Prozeß β zunächst nur aus

$$\beta = (\{b_1, b_2\}, \emptyset),$$

wobei $b_1 = (s_1, \perp)$ und $b_2 = (s_2, \perp)$. Daraus ergibt sich die Menge der möglichen Erweiterungen von β zu

$$\{(t_1, \{b_1\}), (t_3, \{b_1, b_2\}), (t_5, \{b_2\})\}.$$

Nun wird das Ereignis $e_1 = (t_1, \{b_1\})$ zu β hinzugefügt, und man erhält den Branching-Prozeß

$$\beta = (\{b_1, b_2, b_3\}, \{e_1\}),$$

wobei $b_3 = (s_3, e_1)$. Die Menge der möglichen Erweiterungen wird aktualisiert zu

$$\{(t_3, \{b_1, b_2\}), (t_5, \{b_2\}), (t_2, \{b_3\})\}.$$

Durch Einfügen des Ereignisses $e_2 = (t_3, \{b_1, b_2\})$ wird der Branching-Prozeß

$$\beta = (\{b_1, b_2, b_3, b_4, b_5\}, \{e_1, e_2\})$$

mit $b_4 = (s_3, e_2)$ und $b_5 = (s_4, e_2)$ erzeugt. Als mögliche Erweiterungen kommen nun Ereignisse der Menge

$$\{(t_5, \{b_2\}), (t_2, \{b_3\}), (t_2, \{b_4\}), (t_4, \{b_5\})\}$$

in Betracht. Durch Einfügen des Ereignisses $e_3 = (t_5, \{b_2\})$ entsteht der Branching-Prozeß

$$\beta = (\{b_1, b_2, b_3, b_4, b_5, b_6\}, \{e_1, e_2, e_3\}),$$

wobei $b_6 = (s_4, e_3)$. Folglich ergibt sich die Menge der möglichen Erweiterungen zu

$$\{(t_2, \{b_3\}), (t_2, \{b_4\}), (t_4, \{b_5\}), (t_4, \{b_6\})\}.$$

Als nächstes wird das Ereignis $e_4 = (t_2, \{b_3\})$ in β eingefügt. Daraus ergibt sich

$$\beta = (\{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}, \{e_1, e_2, e_3, e_4\})$$

mit $b_7 = (s_1, e_4)$. An dieser Stelle gilt jetzt aber

$$Mark([e_4]) = (s_1, s_2) = Mark([\bot]),$$

und nach der in [ERV02] verwendeten Ordnung gilt auch $[\bot] \prec [e_4]$. Das Ereignis e_4 ist somit als Terminalereignis erkannt worden, und deshalb braucht die Zukunft dieses Ereignisses nicht weiter untersucht zu werden. Folglich bleiben als mögliche Erweiterungen von β die Elemente der Menge

$$\{(t_2, \{b_4\}), (t_4, \{b_5\}), (t_4, \{b_6\})\}$$

übrig. Diese Ereignisse werden nacheinander in das Präfix eingefügt und jeweils als Terminalereignisse identifiziert. Für $e_5 = (t_2, \{b_4\})$ gilt

$$Mark([e_5]) = (s_1, s_4) = Mark([e_3])$$

und $[e_3] \prec [e_5]$. Für $e_6 = (t_4, \{b_5\})$ ergibt sich

$$Mark([e_6]) = (s_2, s_3) = Mark([e_1])$$

und $[e_1] \prec [e_6]$. Schließlich bleibt noch $e_7 = (t_4, \{b_6\})$ zu betrachten. In diesem Fall gibt es jedoch wiederum das virtuelle Ereignis $\bot$, sodaß

$$Mark([e_7]) = (s_1, s_2) = Mark([\bot])$$

sowie $[\bot] \prec [e_7]$. Da es nun keine möglichen Erweiterungen von β mehr gibt, ist die Berechnung des endlichen, vollständigen Präfixes abgeschlossen.

3.2 Das Erreichbarkeitsproblem

Wie in der Einleitung bereits erwähnt wurde, beschäftigt sich das Erreichbarkeitsproblem mit der Fragestellung, ob es für eine gegebene Menge von Stellen eine erreichbare Markierung gibt, die auf jede dieser Stellen eine Marke legt. Manchmal möchte man jedoch auch die Abwesenheit von Marken auf einzelnen Stellen nachweisen, d.h., es wird nach einer erreichbaren Markierung gesucht, die für eine vorgegebene Stellenmenge auf keine Stelle dieser Menge eine Marke legt. Zur Formalisierung dieser Problematik wird das Konzept der Teilmarkierung herangezogen.

Definition 3.2.1 (Teilmarkierung [ES01b])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine *Teilmarkierung* von Σ wird durch eine Abbildung

$$M_{par} : (S^0 \cup S^1) \to \{0, 1\}$$

beschrieben, wobei $S^0, S^1 \subseteq S, S^0 \cap S^1 = \emptyset$ sowie

$$\forall s \in S^0 : M_{par}(s) = 0$$
$$\forall s \in S^1 : M_{par}(s) = 1$$

Eine Teilmarkierung M_{par} wird mit dem Tupel (S^0, S^1) assoziiert.

Definition 3.2.2 (Erreichbarkeitsproblem für S/T-Netzsysteme [ES01b])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ sowie eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$. Das Erreichbarkeitsproblem behandelt die Fragestellung, ob es eine von M_0 aus erreichbare Markierung M gibt, die M_{par} überdeckt, d.h.,

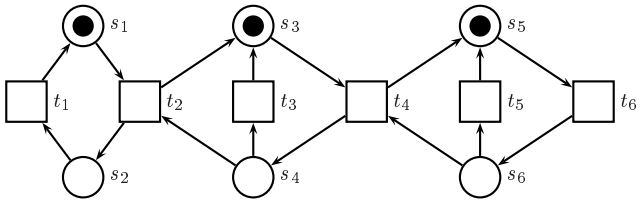
$$\exists M \in \mathcal{R}(M_0) : \forall s \in (S^0 \cup S^1) : M(s) = M_{par}(s)$$

Mit anderen Worten behandelt das Erreichbarkeitsproblem die Fragestellung, ob für zwei disjunkte Stellenmengen S^0 und S^1 eine erreichbare Markierung existiert, die jede Stelle aus S^1, aber keine Stelle aus S^0 mit einer Marke belegt.
Das Erreichbarkeitsproblem für Teilmarkierungen kann sehr einfach auf das Erreichbarkeitsproblem für nur eine Stelle zurückgeführt werden: Man fügt für jede Stelle $s \in S^0$ dessen Komplement $\bar{s}$ in das Netz ein. Anschließend wird eine neue Transition t' hinzugefügt, die von den Komplementen und jeder Stelle aus S^1 eine Marke abzieht und auf einer neuen Stelle s' eine Marke erzeugt, d.h.,

$${}^\bullet t' = \bigcup_{s \in S^0} \bar{s} \cup S^1 \quad \text{und} \quad t'^\bullet = s'$$

Nun genügt es, das Erreichbarkeitsproblem für die Stelle s' zu lösen, denn damit ist es auch für die Teilmarkierung (S^0, S^1) entschieden. Dieser Ansatz wird in dem On-the-Fly-Verfahren aus Abschnitt 3.4.3 auf Seite 76 verwendet, jedoch weist er einen erheblichen Nachteil auf. Das endliche, vollständige Präfix, auf dem die Algorithmen zur Überprüfung von Erreichbarkeit basieren, muß dann für jede Teilmarkierung neu berechnet werden. Die Präfixerzeugung dauert jedoch sehr viel länger als die eigentliche Erreichbarkeitsanalyse. Von daher besteht der Grundgedanke entfaltungsbasierter Verifikationstechniken darin, das Präfix nur einmal zu erzeugen und dann für sämtliche Verifikationsaufgaben zu verwenden. Mit dem Konzept der Teilmarkierungen braucht das Präfix eines Netzsystems nur einmal gebildet zu werden und kann dann von den Algorithmen CheckCo, CheckLin und McSmodels, die in den nachfolgenden Abschnitten vorgestellt werden, zur Überprüfung der Erreichbarkeit beliebiger Teilmarkierungen eingesetzt werden. Darüberhinaus würde das Einfügen komplementärer Stellen in

Abbildung 3.4 *Sicheres S/T-Netzsystem*



das Netzsystem einen nachteiligen Mehraufwand für die Algorithmen CheckLin und McSmodels bedeuten, da diese keine zusätzlichen komplementären Stellen zur Problemlösung benötigen.
In der Einleitung wurde bereits angedeutet, daß das Erreichbarkeitsproblem für sichere S/T-Netzsysteme von einem PSPACE-vollständigen in ein NP-vollständiges Problem überführt werden kann, sofern die endlichen, vollständigen Präfixe der Netzsysteme als Eingangsgrößen vorliegen. Deshalb wird der restliche Teil dieses Abschnitts dem Beweis der NP-Vollständigkeit gewidmet.

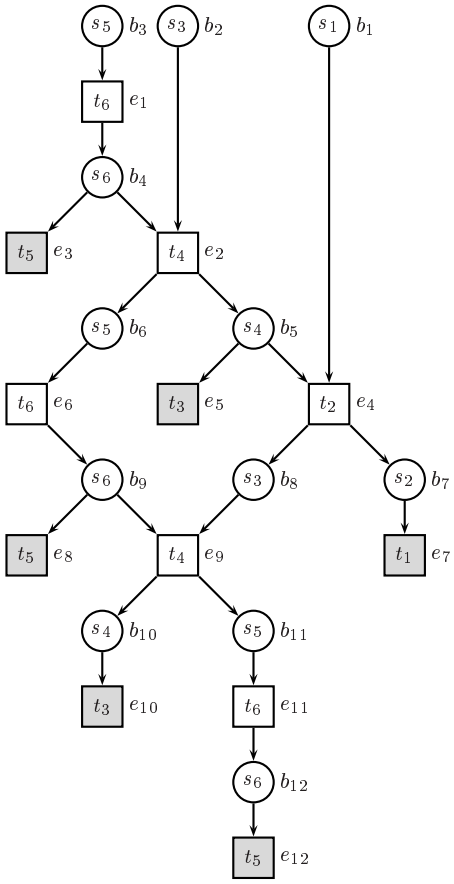
Satz 3.2.1 (NP-Vollständigkeit des Erreichbarkeitsproblems [ES01b])
Gegeben sei ein sicheres S/T-Netzsystem Σ, ein endliches, vollständiges Präfix der Entfaltung von Σ und eine Markierung M. Die Frage zu entscheiden, ob M von der Anfangsmarkierung von Σ aus erreichbar ist, ist ein NP-vollständiges Problem.

Beweis: Zunächst muß gezeigt werden, daß das Erreichbarkeitsproblem für sichere S/T-Netzsysteme durch die Zuhilfenahme von endlichen, vollständigen Präfixen innerhalb der Komplexitätsklasse NP liegt, d.h., daß es ein nichtdeterministisches Verfahren gibt, welches die Erreichbarkeitsfrage in polynomieller Zeit löst. Dieses kann dadurch geschehen, daß das Erreichbarkeitsproblem in polynomieller Zeit auf ein beliebiges NP-vollständiges Problem reduziert wird. Als Beispiel sei hierfür das Erfüllbarkeitsproblem für aussagenlogische Formeln erwähnt, welches in der Literatur im allgemeinen unter dem SAT-Problem [Sch92] bekannt ist. Im folgenden wird also gezeigt, wie das Erreichbarkeitsproblem durch die Zuhilfenahme von endlichen, vollständigen Präfixen in polynomieller Zeit in das SAT-Problem überführt werden kann.[6] Zusätzlich wird die Reduktion beispielhaft anhand der Teilmarkierung $(\{s_5\}, \{s_2, s_4\})$ für das Netzsystem aus Abbildung 3.4 und dessen endlichem, vollständigen Präfix (Abbildung 3.5 auf der nächsten Seite) erläutert.
Gegeben seien also ein endliches, vollständiges Präfix

$$\beta = (N', \lambda, Min(N')) \text{ mit } N' = (B, E, F')$$

[6]Dieser Beweis ist im Prinzip unnötig kompliziert, jedoch wird die Reduktion in Abschnitt 3.4.2 auf Seite 71 implizit wiederverwendet. Für die Zugehörigkeit zu NP hätte es genügt zu argumentieren, daß es für jede erreichbare Markierung eine Schaltfolge polynomieller Länge in Bezug auf das Präfix gibt, die zu der gesuchten Markierung führt.

Abbildung 3.5 *Endliches, vollständiges Präfix für das Netzsystem aus Abbildung 3.4 auf der vorherigen Seite, wobei die Terminalereignisse grau unterlegt sind. Der Übersichtlichkeit halber werden die Ausgangsbedingungen von Terminalereignissen nicht dargestellt.*



sowie eine Teilmarkierung

$$M_{par} = (\{s_1, \ldots, s_k\}, \{s_{k+1}, \ldots, s_n\}).$$

Zunächst wird für jede Bedingung und für jedes Nicht-Terminalereignis von N' eine entsprechende Variable eingeführt.[7] Der Einfachheit halber werden die Namen der Bedingungen und Ereignisse als Variablennamen übernommen, d.h., es wird

- für jede Bedingung $b \in B$ eine Variable b und
- für jedes Nicht-Terminalereignis $e \in E \setminus \mathit{CutOffs}(E)$ eine Variable e eingeführt.

[7] Terminalereignisse brauchen nicht betrachtet zu werden, da nach Definition 3.1.10 auf Seite 45 jede erreichbare Markierung ohne das Schalten eines Terminalereignisses erreichbar ist.

Auf das Beispiel bezugnehmend erhält man somit die Variablen

$$b_1, \ldots, b_{12} \quad \text{und} \quad e_1, e_2, e_4, e_6, e_9, e_{11}.$$

Im weiteren Verlauf werden die Variablen derart interpretiert, daß die zu der Variablen b ($\neg b$) korrespondierende Bedingung eine (keine) Marke enthält und das zu e ($\neg e$) korrespondierende Ereignis geschaltet (nicht geschaltet) hat. Als nächstes wird für jede Bedingung $b \in B$ eine Regel definiert, die angibt, daß b genau dann eine Marke enthält, nachdem das Ereignis im Vorbereich geschaltet hat und bevor eines der Ereignisse aus dem Nachbereich schalten wird:

$$\forall b \in B\colon b \leftrightarrow \bigwedge_{e \in {}^\bullet b} e \bigwedge_{e \in b^\bullet} \neg e$$

Die Bedingung b_4 beispielsweise enthält genau dann eine Marke, nachdem das Ereignis e_1 geschaltet hat und bevor das Ereignis e_2 schalten wird. Man erhält also die Formel

$$b_4 \leftrightarrow e_1 \wedge \neg e_2.$$

Als nächstes muß die Kausalrelation des Präfixes kodiert werden, d.h., falls ein Ereignis e geschaltet hat, müssen zuvor auch alle direkten Vorgängerereignisse geschaltet haben. Dabei können die Nicht-Terminalereignisse wieder außer Acht gelassen werden:

$$\forall e \in E \setminus \mathit{CutOffs}(E)\colon e \rightarrow \bigwedge_{e_i \in {}^\bullet({}^\bullet e)} e_i$$

Hat beispielsweise das Ereignis e_9 geschaltet, so müssen bereits zuvor die Ereignisse e_4 und e_6 geschaltet haben:

$$e_9 \rightarrow e_4 \wedge e_6$$

Schließlich muß noch die Teilmarkierung

$$(\{s_1, \ldots, s_k\}, \{s_{k+1}, \ldots, s_n\})$$

kodiert werden. Dabei darf kein Repräsentant einer Stelle s_i für $1 \leq i \leq k$ eine Marke enthalten. Hingegen muß es für jede Stelle s_j mit $k+1 \leq j \leq n$ jeweils mindestens einen markierten Repräsentanten geben. Somit erhält man folgende Formeln:

$$\forall s \in \{s_1, \ldots, s_k\}\colon \bigwedge_{\lambda(b)=s} \neg b$$

$$\forall s \in \{s_{k+1}, \ldots, s_n\}\colon \bigvee_{\lambda(b)=s} b$$

Für die Teilmarkierung $(\{s_5\}, \{s_2, s_4\})$ aus dem Beispiel erhält man die Formeln:

$$\neg b_3 \wedge \neg b_6 \wedge \neg b_{11}$$
$$b_7$$
$$b_5 \vee b_{10}$$

Abbildung 3.6 *SAT-Formel für das Präfix aus Abbildung 3.5 auf Seite 50 und die Teilmarkierung* $(\{s_5\}, \{s_2, s_4\})$

$b_1 \leftrightarrow \neg e_4$	$b_7 \leftrightarrow e_4$	$e_2 \rightarrow e_1$	$\neg b_3 \wedge \neg b_6 \wedge \neg b_{11}$
$b_2 \leftrightarrow \neg e_2$	$b_8 \leftrightarrow e_4 \wedge \neg e_9$	$e_4 \rightarrow e_2$	b_7
$b_3 \leftrightarrow \neg e_1$	$b_9 \leftrightarrow e_6 \wedge \neg e_9$	$e_6 \rightarrow e_2$	$b_5 \vee b_{10}$
$b_4 \leftrightarrow e_1 \wedge \neg e_2$	$b_{10} \leftrightarrow e_9$	$e_9 \rightarrow e_4 \wedge e_6$	
$b_5 \leftrightarrow e_2 \wedge \neg e_4$	$b_{11} \leftrightarrow e_9 \wedge \neg e_{11}$	$e_{11} \rightarrow e_9$	
$b_6 \leftrightarrow e_2 \wedge \neg e_6$	$b_{12} \leftrightarrow e_{11}$		

Das vollständige Regelsystem ist in Abbildung 3.6 dargestellt. Von der Konstruktionsweise ist unmittelbar klar, daß die Teilmarkierung M_{par} genau dann erreichbar ist, wenn eine Variablenbelegung existiert, die alle Formeln erfüllt. Zum Beispiel erfüllt die Belegung, welche die Variablen der Menge

$$\{b_7, b_{10}, b_{12}, e_1, e_2, e_4, e_6, e_9, e_{11}\}$$

mit 1 und alle anderen Variablen mit 0 belegt, alle Formeln, woraus unmittelbar die Erreichbarkeit der Teilmarkierung $(\{s_5\}, \{s_2, s_4\})$ hergeleitet werden kann.
Nachdem nun gezeigt wurde, daß das Erreichbarkeitsproblem unter Verwendung von Präfixen in der Komplexitätsklasse NP liegt, muß nun dessen NP-Härte nachgewiesen werden. Dieses geschieht, indem das SAT-Problem in polynomieller Zeit auf das Erreichbarkeitsproblem reduziert wird.
Gegeben sei eine Formel φ in konjunktiver Normalform mit den

$$\text{Variablen } x_1, \ldots, x_n \text{ und den Klauseln } c_1, \ldots, c_m.$$

Nun wird folgendermaßen ein sicheres S/T-Netzsystem $(N_\varphi, M_{0_\varphi})$ mit $N_\varphi = (S_\varphi, T_\varphi, F_\varphi)$ konstruiert: N_φ enthalte

- für jede Variable x_i eine Stelle s_{x_i}, sodaß

$$^\bullet s_{x_i} = \emptyset \quad \text{und} \quad s_{x_i}^\bullet = \{t_{x_i}, t_{\overline{x_i}}\}.$$

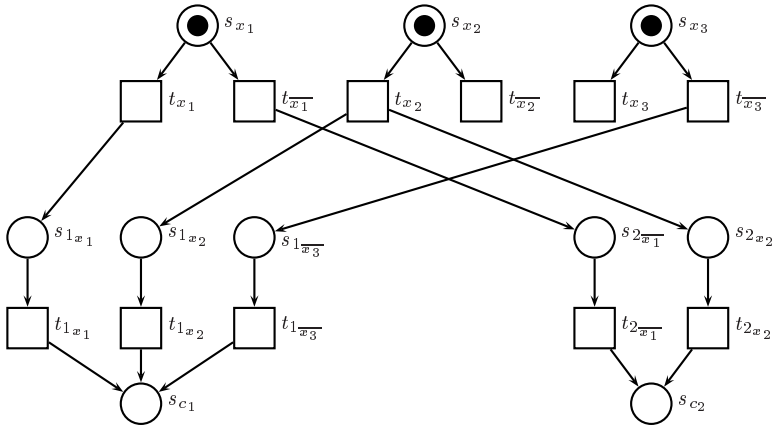
- für jede Klausel c_j und jedes Literal l von c_j eine Stelle s_{j_l}, sodaß

$$^\bullet s_{j_l} = \{t_l\} \quad \text{und} \quad s_{j_l}^\bullet = \{t_{j_l}\}.$$

- für jede Klausel c_j eine Stelle s_{c_j}, sodaß

$$^\bullet s_{c_j} = \bigcup_{l \in c_j} \{t_{j_l}\} \quad \text{und} \quad s_{c_j}^\bullet = \emptyset.$$

Abbildung 3.7 *Netzsystem* $(N_\varphi, M_{0_\varphi})$ *für die Formel* $\varphi = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2)$



M_{0_φ} belege genau jede Stelle s_{x_i} mit einer Marke. Abbildung 3.7 zeigt ein Netzsystem, das nach obigem Verfahren für die Formel

$$\varphi = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2)$$

konstruiert wurde. Diese Konstruktion beinhaltet jetzt noch die Problematik, daß das erhaltene Netzsystem nicht sicher ist, da der Fall eintreten kann, daß zwei Transitionen t_{j_l} und t_{j_m} unabhängig voneinander schalten und somit beide eine Marke auf die Stelle s_{c_j} legen. Zum Beispiel führt die Ausführung der Schaltfolge

$$t_{x_1} t_{x_2} t_{1_{x_1}} t_{1_{x_2}}$$

im Netzsystem aus Abbildung 3.7 zu einer Markierung, welche die Stelle s_{c_1} mit zwei Marken belegt. Dieser unerwünschte Effekt kann beseitigt werden, indem eine zusätzliche Stelle eingefügt wird, welche garantiert, daß nur eine der genannten Transitionen schalten kann. Daher füge man

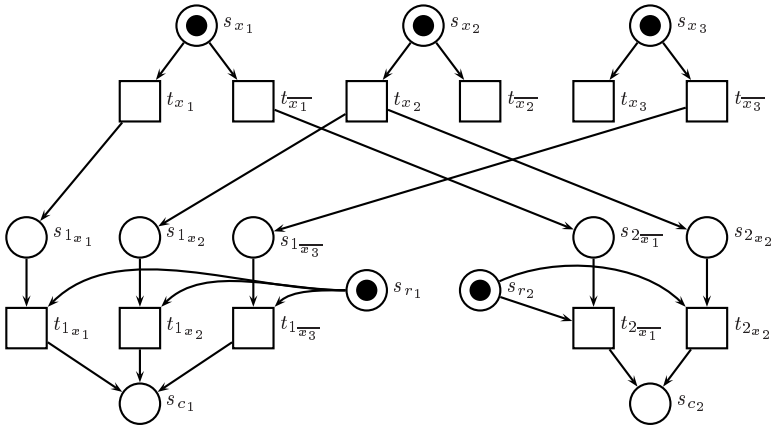
- für jede Klausel c_j eine Stelle s_{r_j} hinzu, so daß

$$^\bullet s_{r_j} = \emptyset \quad \text{und} \quad s_{r_j}^\bullet = \bigcup_{l \in c_j} \{t_{j_l}\}.$$

Abbildung 3.8 auf der nächsten Seite zeigt das modifizierte Netzsystem, welches jetzt die zusätzliche Eigenschaft aufweist, daß es sicher ist. Aus dieser Vorgehensweise folgt unmittelbar, daß die Formel φ genau dann erfüllbar ist, wenn in dem sicheren Netzsystem $(N_\varphi, M_{0_\varphi})$ eine von M_{0_φ} aus erreichbare Markierung M existiert, die auf jede Stelle s_{c_j} eine Marke legt, d.h.,

$$\exists M \in \mathcal{R}(M_{0_\varphi}) \colon \forall s_{c_j} (1 \leq j \leq m) \colon M(s_{c_j}) = 1$$

Abbildung 3.8 *Sicheres Netzsystem* $(N_\varphi, M_{0_\varphi})$ *für Formel* $\varphi = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2)$



Nun gilt es noch zu zeigen, daß $(N_\varphi, M_{0_\varphi})$ in polynomieller Zeit in ein endliches, vollständiges Präfix entfaltet werden kann. Eine strukturelle Eigenschaft von Präfixen besteht darin, daß jede Bedingung höchstens ein Ereignis in ihrem Vorbereich aufweisen darf. Betrachtet man das Netzsystem aus Abbildung 3.8, so wird deutlich, daß nur die Stellen s_{c_j} diese Eigenschaft verletzen, da deren Vorbereiche aus mehr als nur einer Transition bestehen. Dieser Mißstand kann aber leicht behoben werden, indem diese Stellen durch entsprechende Kopien ersetzt werden. Genauer gesagt,

- man ersetze jede Stelle s_{c_j} durch $k = |{}^\bullet s_{c_j}|$ Stellen, d.h., man entferne s_{c_j} und füge für jedes Literal $l \in c_j$ eine Stelle $s_{c_{j_l}}$ hinzu, sodaß

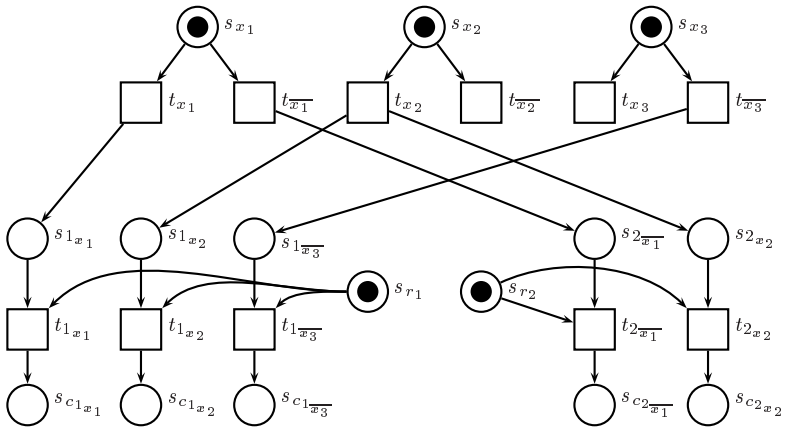
$$ {}^\bullet s_{c_{j_l}} = \{t_{j_l}\} \quad \text{und} \quad s^\bullet_{c_{j_l}} = \emptyset. $$

Das so erhaltene endliche, vollständige Präfix weist die Struktur des in Abbildung 3.9 auf der nächsten Seite dargestellten Netzsystems auf. Folglich ist die Formel φ jetzt genau dann erfüllbar, wenn in dem Präfix eine erreichbare Markierung existiert, die für jede Stelle s_{c_j} des ursprünglichen Netzsystems jeweils eine Marke auf genau eine der entsprechenden Stellen $s_{c_{j_l}}$ legt. ■

3.3 Ein graphentheoretischer Algorithmus

Betrachtet man für eine Teilmarkierung $M_{par} = (S^0, S^1)$ eines sicheren S/T-Netzsystems Σ die SAT-Formel, welche zur Nachweisbarkeit der Erreichbarkeit von M_{par} aus dem endlichen, vollständigen Präfix $\beta = (B, E)$ von Σ gewonnen wird (siehe beispielsweise Abbildung 3.6 auf Seite 52), so wird deutlich, daß die SAT-Formel für jede Bedingung und für jedes Ereignis von β eine Variable aufweist, also ingesamt $|B| + |E|$.

Abbildung 3.9 *„Endliches, vollständiges Präfix“ für das Netzsystem aus Abbildung 3.8 auf der vorherigen Seite*



Daraus folgt unmittelbar, daß die Zeitkomplexität eines SAT-Solvers für die Suche nach einer erfüllenden Belegung der SAT-Formel bei

$$\mathcal{O}(2^{|B|+|E|})$$

liegt. Bei Kenntnis der Teilmarkierung M_{par} ist jedoch unmittelbar klar, daß bestimmte Belegungen die Formel nicht erfüllen und von vornherein ausgeschlossen werden können. So kann es für keine die SAT-Formel erfüllende Belegung vorkommen, daß die Variable einer Bedingung, die eine Stelle aus S^0 repräsentiert, mit **wahr** belegt wird. Desweiteren können wegen der Sicherheit von Σ alle Fälle ausgeschlossen werden, in denen zwei oder mehrere Variablen von Bedingungen, welche dieselbe Stelle von Σ repräsentieren, mit **wahr** belegt werden, das heißt, für jede gültige Belegung der Formel gilt, daß für jede Stelle aus S^1 die Variable von genau einer diese Stelle repräsentierenden Bedingung mit **wahr** belegt werden muß. Die Auswahl für solche Variablen liegt in der Größenordnung von

$$\mathcal{O}(|B|^{|S^1|}).$$

Ein SAT-Solver besitzt dieses „zusätzliche Wissen“ nicht, sondern muß es sich durch Ausprobieren erst erarbeiten.
Unter diesem Hintergrund wird im folgenden Abschnitt ein graphentheoretischer Algorithmus vorgestellt, der das zusätzliche Wissen ausnutzt, daß für die Auswahl der die Stellen aus S^1 repräsentierenden Bedingungen lediglich $\mathcal{O}(|B|^{|S^1|})$ Möglichkeiten zur Verfügung stehen. Das dem graphentheoretischen Algorithmus CheckCo [ES01b] zugrundeliegende Konzept stellt die in Definition 3.1.3 auf Seite 39 eingeführte *Co-Relation* dar. Eine wichtige Rolle dabei spielt die Tatsache, daß Co-Mengen und erreichbare Markierungen eng miteinander verknüpft sind.

Proposition 3.3.1 (Erreichbarkeit von Co-Mengen [NPW80, BF88])
Gegeben seien ein endliches, vollständiges Präfix $(N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ und eine Teilmarkierung $M_{par} = (\emptyset, S^1)$ mit $S^1 \subseteq B$. M_{par} ist genau dann erreichbar, wenn S^1 eine Co-Menge darstellt.

Das Ergebnis von Proposition 3.3.1 kann nun direkt auf den Nachweis der Erreichbarkeit von Teilmarkierungen in S/T-Netzsystemen übertragen werden.

Satz 3.3.1 (Erreichbarkeit von Teilmarkierungen [Mel98, ES01b])
Gegeben seien ein sicheres S/T-Netzsystem (N, M_0) mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $(N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, sowie eine Teilmarkierung $M_{par} = (\emptyset, S^1)$ mit $S^1 \subseteq S$. M_{par} ist genau dann erreichbar, wenn eine Co-Menge $B' \subseteq B$ existiert, sodaß es für jede Stelle $s \in S^1$ eine Bedingung $b \in B'$ mit $\lambda(b) = s$ gibt.

Satz 3.3.1 soll anhand eines Beispiels näher erläutert werden. Dazu werden erneut das Netzsystem aus Abbildung 3.4 auf Seite 49 und dessen endliches, vollständiges Präfix (Abbildung 3.5 auf Seite 50) betrachtet. Die Co-Relation des Präfixes ergibt sich als symmetrische Hülle der Menge

$$\{(b_1, b_2), (b_1, b_3), (b_1, b_4), (b_1, b_5), (b_1, b_6), (b_1, b_9), (b_2, b_3), (b_2, b_4), (b_5, b_6), (b_5, b_9),$$
$$(b_6, b_7), (b_6, b_8), (b_7, b_8), (b_7, b_9), (b_7, b_{10}), (b_7, b_{11}), (b_7, b_{12}), (b_8, b_9), (b_{10}, b_{11}), (b_{10}, b_{12})\}.$$

Nun soll die Erreichbarkeit der Teilmarkierung

$$M_{par} = (\emptyset, \{s_2, s_4, s_6\})$$

nachgewiesen werden. Nach Satz 3.3.1 ist M_{par} genau dann erreichbar, wenn für jede der Stellen s_2, s_4 und s_6 entsprechende Repräsentanten im Präfix existieren, die zusammen eine Co-Menge bilden. Die Menge

$$\{b_7, b_{10}, b_{12}\}$$

erfüllt genau diese Kriterien, denn es gilt:

$$\begin{aligned} s_2 &= \lambda(b_7) & (b_7, b_{10}) &\in co \\ s_4 &= \lambda(b_{10}) & (b_7, b_{12}) &\in co \\ s_6 &= \lambda(b_{12}) & (b_{10}, b_{12}) &\in co \end{aligned}$$

3.3.1 Reduktion auf das CLIQUE-Problem

Die Suche nach einer geeigneten Co-Menge entspricht genau der graphentheoretischen Problemstellung, eine CLIQUE der Größe k [Sch92] in einem k-partiten Graphen zu finden. Diese Beziehung soll im folgenden näher beleuchtet werden. Zunächst wird ein Verfahren angegeben, das einen k-partiten Graphen definiert, der das Erreichbarkeitsproblem hinreichend beschreibt.

Definition 3.3.1 (k-partiter Graph $G_k = (V, E)$)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, eine Teilmarkierung $M_{par} = (\emptyset, \{s_1, s_2, \ldots, s_k\})$ über den Stellen von Σ sowie die Co-Relation $co \subseteq B \times B$ von β. Der Graph $G_k = (V, E)$ wird dann wie folgt konstruiert:

(i) Für jede Stelle $s_i \in \{s_1, s_2, \ldots, s_k\}$ berechne man die Menge $B_i = \{b_{i_1}, b_{i_2}, \ldots, b_{i_m}\}$ von Bedingungen, sodaß $\lambda(b_{i_j}) = s_i$ für alle $1 \leq j \leq m$.

(ii) Die Menge der Knoten von G_k ergebe sich zu $V = \bigcup_{1 \leq i \leq k} B_i$.

(iii) Für $i \neq j$ verbinde man die Knoten b_{i_m}, $b_{j_n} \in V$ mit einer Kante, falls $(b_{i_m}, b_{j_n}) \in co$, d.h., zwei in verschiedenen Partitionen liegende Knoten werden durch eine Kante verbunden, falls sie nebenläufig sind.

Im folgenden wird der Graph G_k für das Netzsystem aus Abbildung 3.4 auf Seite 49 und die Teilmarkierung $(\emptyset, \{s_2, s_4, s_6\})$ konstruiert. Die Mengen B_i der Stellen s_2, s_4 und s_6 können direkt aus dem Präfix (Abbildung 3.5 auf Seite 50) abgeleitet werden, denn mit

$$\begin{aligned} s_2 &= \lambda(b_7) \\ s_4 &= \lambda(b_5) = \lambda(b_{10}) \\ s_6 &= \lambda(b_4) = \lambda(b_9) = \lambda(b_{12}) \end{aligned}$$

erhält man sofort

$$\begin{aligned} B_1 &= \{b_7\} \\ B_2 &= \{b_5, b_{10}\} \\ B_3 &= \{b_4, b_9, b_{12}\} \end{aligned}$$

Nun werden genau die Knoten durch Kanten verbunden, die in verschiedenen Partitionen liegen und nebenläufig sind, d.h., in Co-Relation stehen. Folglich erhält man die Kanten

$$(b_7, b_{10}), (b_7, b_9), (b_7, b_{12}), (b_5, b_9), (b_{10}, b_{12}).$$

Da die Co-Relation symmetrisch ist, wird ein Graph mit ungerichteten Kanten betrachtet. Abbildung 3.10 auf der nächsten Seite zeigt den 3-partiten Graphen G_3, der sich aus Definition 3.3.1 für das Netzsystem aus Abbildung 3.4 auf Seite 49 und die Teilmarkierung $(\emptyset, \{s_2, s_4, s_6\})$ ergibt. Die Knoten b_7, b_{10} und b_{12} bilden eine CLIQUE der Größe 3, woraus unmittelbar die Erreichbarkeit der Teilmarkierung $(\emptyset, \{s_2, s_4, s_6\})$ gefolgert werden kann.

Satz 3.3.2 (Erreichbarkeit von Teilmarkierungen [ES01b])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, eine Teilmarkierung $M_{par} = (\emptyset, S^1)$ mit $S^1 \subseteq S$ sowie die Co-Relation $co \subseteq B \times B$ von β. M_{par} ist genau dann erreichbar, wenn der nach Definition 3.3.1 erhaltene k-partite Graph G_k eine CLIQUE der Größe k besitzt.

Beweis: Direkte Folgerung aus Definition 3.3.1. ■

Abbildung 3.10 *3-partiter Graph G_3 für das Präfix aus Abbildung 3.5 auf Seite 50 und die Teilmarkierung $(\emptyset, \{s_2, s_4, s_6\})$*

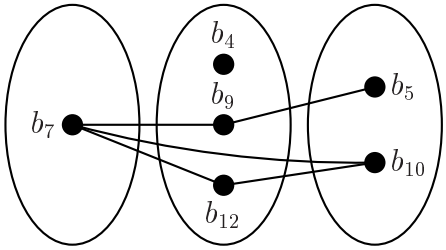
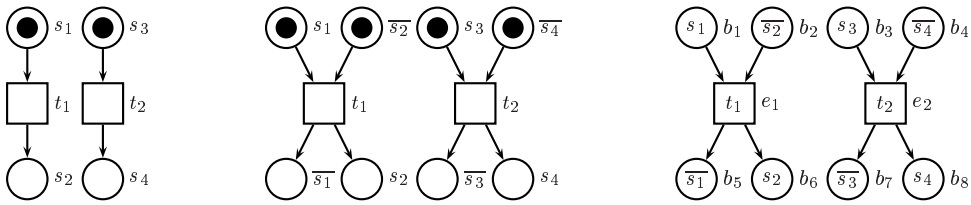


Abbildung 3.11 *Sicheres S/T-Netzsystem (a), das mit komplementären Stellen erweiterte Netzsystem (b) und das endliche, vollständige Präfix für das Netzsystem mit komplementären Stellen (c)*



(a) (b) (c)

3.3.2 Komplementäre Stellen

Das in den vorangegangenen Abschnitten vorgestellte Verfahren kann die Erreichbarkeit für Teilmarkierungen der Art $(\emptyset, S^1)$ entscheiden, versagt aber, wenn Teilmarkierungen (S^0, S^1) mit $S^0 \neq \emptyset$ betrachtet werden. Für das Netzsystem aus Abbildung 3.11 (a) soll beispielsweise die Erreichbarkeit der Teilmarkierung $(\{s_4\}, \{s_2\})$ gezeigt werden. Es ist jedoch nicht möglich, allein unter Verwendung der Co-Relation zu überprüfen, ob eine Markierung erreichbar ist, die eine Marke auf s_2, jedoch keine Marke auf s_4 legt. Aus diesem Grund wird das Konzept der komplementären Stellen eingeführt, dessen Grundidee darin besteht, daß eine Stelle genau dann eine Marke enthält, wenn ihr Komplement keine Marke besitzt. Mit dieser Vorgehensweise kann die Erreichbarkeit der Teilmarkierung $(\{s_4\}, \{s_2\})$ entschieden werden, indem nach einer erreichbaren Markierung gesucht wird, die jeweils eine Marke auf s_2 und das Komplement von s_4 legt.

Definition 3.3.2 (Komplement einer Stelle [ES01b])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine

Stelle $s \in S$. Eine Stelle $\bar{s} \in S$ wird als *Komplement* von s bezeichnet, falls

$$\forall M \in \mathcal{R}(M_0)\colon M(\bar{s}) = 1 - M(s)$$

Definition 3.3.2 auf der vorherigen Seite besagt, daß eine Stelle unter jeder erreichbaren Markierung genau dann eine Marke besitzt, wenn ihr Komplement keine Marke enthält. Mit Hilfe dieses Konzeptes kann nun ein Kriterium für die Erreichbarkeit von Teilmarkierungen in sicheren S/T-Netzsystemen hergeleitet werden.

Satz 3.3.3 (Erreichbarkeit von Teilmarkierungen [ES01b])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine Teilmarkierung $M_{par} = (\{s_1, \ldots, s_k\}, \{s_{k+1}, \ldots, s_n\})$ über den Stellen von Σ. M_{par} ist genau dann von M_0 aus erreichbar, wenn die Teilmarkierung

$$(\emptyset, \{\overline{s_1}, \ldots, \overline{s_k}, s_{k+1}, \ldots, s_n\})$$

von M_0 aus erreichbar ist.

Beweis: Direkte Folgerung aus Definition 3.3.2 auf der vorherigen Seite. ■

Mit dem Ergebnis von Satz 3.3.3 kann nun die Erreichbarkeit der Teilmarkierung $(\{s_4\}, \{s_2\})$ für das Netzsystem aus Abbildung 3.11 (a) auf der vorherigen Seite gezeigt werden, indem das Komplement $\overline{s_4}$ von s_4 eingefügt (Abbildung 3.11 (b)) und die Erreichbarkeit der Teilmarkierung $(\emptyset, \{\overline{s_4}, s_2\})$ nachgewiesen wird. Dieses geschieht wieder nach dem herkömmlichen Verfahren (Satz 3.3.1 auf Seite 56), welches in dem Präfix (Abbildung 3.11 (c)) nach einer Co-Menge sucht, sodaß die Stellen s_2 und $\overline{s_4}$ jeweils durch genau eine Bedingung aus dieser Co-Menge repräsentiert werden. Aus

$$\lambda(b_6) = s_2, \quad \lambda(b_4) = \overline{s_4} \quad \text{sowie} \quad (b_4, b_6) \in co$$

folgt nach Satz 3.3.1 auf Seite 56, daß die Teilmarkierung $(\emptyset, \{\overline{s_4}, s_2\})$ und somit nach Satz 3.3.3 auch die Teilmarkierung $(\{s_4\}, \{s_2\})$ erreichbar ist.
Prinzipiell könnte man meinen, daß für eine Stelle s und deren Komplement $\bar{s}$ die Beziehung

$$^\bullet s = \bar{s}^\bullet \quad \text{und} \quad s^\bullet = {}^\bullet\bar{s}$$

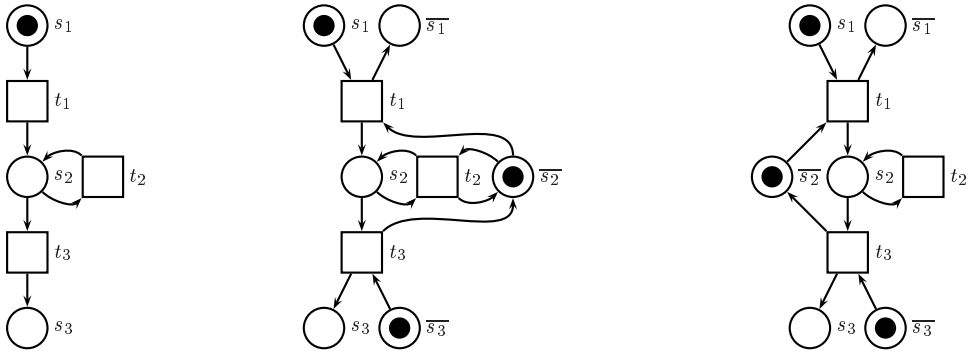
gilt. Eine Ausnahme stellt jedoch die Situation dar, daß die Stelle s eine Schlinge besitzt, d.h.,

$$\exists t \in T\colon t \in {}^\bullet s \cap s^\bullet$$

In Abbildung 3.12 (a) auf der nächsten Seite ist ein Netzsystem dargestellt, in dem die Stelle s_2 eine Schlinge besitzt. Würde man in dieses Netz nun die Stellenkomplemente unter Berücksichtigung der Beziehungen $^\bullet s = \bar{s}^\bullet$ und $s^\bullet = {}^\bullet\bar{s}$ einfügen, wäre die Transition t_2 tot, d.h., sie könnte nie schalten, wie in Abbildung 3.12 (b) zu sehen ist. Dieser Mißstand kann jedoch leicht behoben werden, indem die Kanten zwischen $\overline{s_2}$ und t_2 entfernt werden (Abbildung 3.12 (c)). Das Komplement $\bar{s}$ einer Stelle s wird beim Einfügen in ein Netzsystem also nur dann mit einer Transition t verbunden, falls

$$t \in {}^\bullet s \setminus s^\bullet \quad \text{oder} \quad t \in s^\bullet \setminus {}^\bullet s.$$

Abbildung 3.12 *Netzsystem mit Schlinge (a), falsche Konstruktion des Komplements von s_2 (b) und korrekte Konstruktion des Komplements von s_2 (c)*



(a) (b) (c)

Hieraus ergibt sich dann unmittelbar die Beziehung

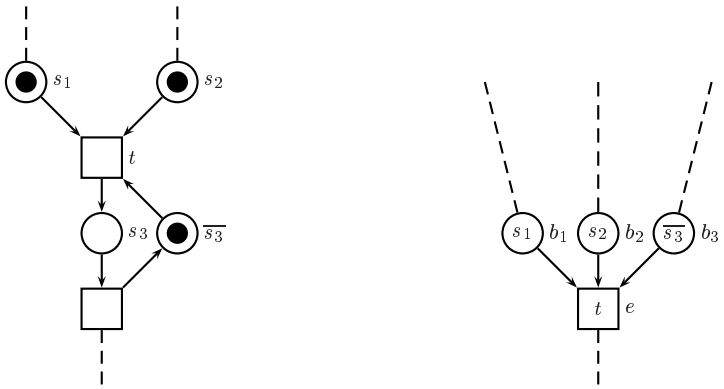
$$t \in \bar{s}^\bullet \Leftrightarrow t \in {}^\bullet s \setminus s^\bullet \quad \text{und} \quad t \in {}^\bullet \bar{s} \Leftrightarrow t \in s^\bullet \setminus {}^\bullet s.$$

An dieser Stelle ist es wichtig, darauf hinzuweisen, daß die Stärken und Vorteile der Entfaltungstechniken durch diese Vorgehensweise nicht beschnitten werden, da das Einfügen von komplementären Stellen keinesfalls die Nebenläufigkeit eines verteilten Systems einschränkt.

3.3.3 Komplementäre Bedingungen im Präfix

Einer der Hauptvorteile bei der Verwendung von Netzentfaltungen im Vergleich zu anderen Techniken liegt darin, daß die Entfaltung (das Präfix) für das zu verifizierende System nur einmal erzeugt werden muß und anschließend für Verifikationsaufgaben wie zum Beispiel dem Nachweis von Verklemmungsfreiheit oder der Erreichbarkeit von Teilmarkierungen verwendet werden kann. Da bei dem co-relationsbasierten Ansatz komplementäre Stellen benötigt werden, um Erreichbarkeit für Teilmarkierungen (S^0, S^1) mit $S^0 \neq \emptyset$ zu entscheiden, wurde in [ES00] vorgeschlagen, zusätzlich die Komplemente aller Netzstellen zu erzeugen und dann das Präfix zu bilden. Experimentelle Ergebnisse haben jedoch gezeigt, daß die Entfaltungszeit für ein Netz mit den zusätzlichen komplementären Stellen bis zu 20-mal größer sein kann als die Entfaltungszeit für das Netz ohne Komplemente [ES00]. Dieses Ergebnis ist nicht weiter verwunderlich, wie anhand des Netzausschnitts aus Abbildung 3.13 (a) auf der nächsten Seite ersichtlich wird. Nach Definition 3.1.5 auf Seite 39 kann ein Ereignis e mit $\lambda(e) = t$ dann in die Entfaltung eingefügt werden, wenn eine Co-Menge von Bedingungen existiert, sodaß jede Eingangsstelle von t durch eine der Bedingungen aus der Co-Menge repräsentiert wird. Dieses

Abbildung 3.13 *Ausschnitt aus einem S/T-Netzsystem (a) und dessen Präfix (b)*



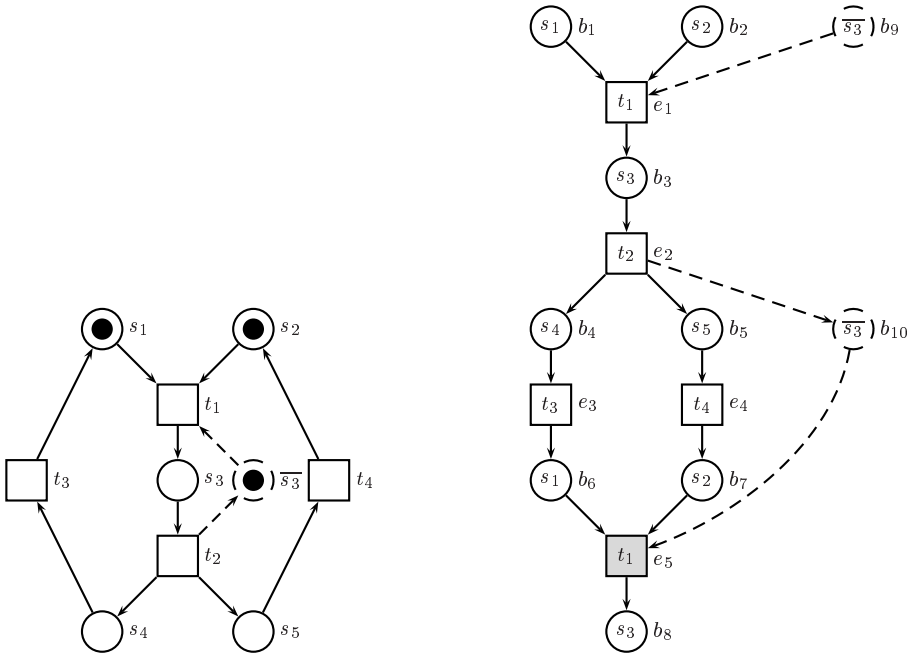
(a) (b)

bedeutet bezüglich des Beispiels aus Abbildung 3.13 (a) auf dieser Seite, daß das Ereignis e mit $\lambda(e) = t$ in die Entfaltung (Abbildung 3.13 (b) auf dieser Seite) eingefügt werden kann, wenn es drei Bedingungen b_1, b_2 und b_3 gibt, so daß

$$\lambda(b_1) = s_1 \qquad (b_1, b_2) \in co$$
$$\lambda(b_2) = s_2 \qquad (b_1, b_3) \in co$$
$$\lambda(b_3) = \overline{s_3} \qquad (b_2, b_3) \in co$$

Wie bereits erwähnt wurde, entspricht die Suche nach einer solchen Co-Menge dem CLIQUE-Problem, und die Anzahl möglicher Co-Mengen steigt mit der Anzahl der Eingangsstellen von t. Die Berechnung der Co-Relation und die kombinatorische Suche nach geeigneten Co-Mengen machen die Hauptkosten des Entfaltungsverfahrens aus. Von daher ist die Zeitexplosion nicht weiter verwunderlich, die durch das zusätzliche Einfügen von komplementären Stellen in das Netz entstehen kann. Deshalb würde das Einfügen aller Stellenkomplemente in das Netz vor dessen Entfaltung das Verfahren für praktische Belange bedeutungslos erscheinen lassen.
Diese Problematik kann umgangen werden, indem die für die Verifikation notwendigen Komplemente direkt in das Präfix eingefügt werden. Die Hauptidee besteht also darin, zuerst das ursprüngliche Netzsystem zu entfalten und anschließend diejenigen Stellenkomplemente in das Präfix einzufügen, welche zur Lösung des Erreichbarkeitsproblems benötigt werden. Ein großer Vorteil dieser Vorgehensweise besteht darin, daß der Algorithmus zunächst dasselbe Präfix als Eingabe verwendet wie die anderen Verfahren. Von daher stellt dieser Ansatz eine deutliche Verbesserung zu der in [ES00] beschriebenen Vorgehensweise dar.
Anhand des Netzsystems aus Abbildung 3.14 (a) und dessen endlichem, vollständigen Präfix aus Abbildung 3.14 (b) auf der nächsten Seite soll nun erläutert werden, wie die

Abbildung 3.14 *Sicheres S/T-Netzsystem (a) und dessen endliches, vollständiges Präfix (b), wobei die Terminalereignisse grau unterlegt sind*



Repräsentanten für die komplementäre Stelle $\overline{s_3}$ von s_3 direkt in das Präfix eingefügt werden können. Nach Definition 3.3.2 auf Seite 58 gilt

$$M_0(\overline{s_3}) = 1 = 1 - M_0(s_3),$$

woraus unmittelbar folgt, daß eine Bedingung

$$b_9 = (\overline{s_3}, \perp) \quad \text{mit} \quad b_9 \in Min(N')$$

in das Präfix eingefügt werden muß. Als nächstes wird für jedes Ereignis e mit $\lambda(e) \in s_3^\bullet$ eine Bedingung $b = (\overline{s_3}, e)$ hinzugefügt. In dem Beispiel erhält man somit die Bedingung

$$b_{10} = (\overline{s_3}, e_2).$$

Schließlich muß der Vorbereich jedes Ereignisses e mit $\lambda(e) \in {}^\bullet s_3$ mit genau einer der bereits eingefügten Repräsentanten für $\overline{s_3}$ erweitert werden. Satz 3.3.4 besagt, daß es für jedes Ereignis e mit $\lambda(e) \in {}^\bullet s_3$ genau eine solche Bedingung b mit $\lambda(b) = \overline{s_3}$ gibt, sodaß ${}^\bullet e \cup \{b\}$ eine Co-Menge darstellt. Desweiteren befindet sich diese Bedingung in der Anfangsmarkierung $Min(N')$, oder deren Eingangsereignis ist Bestandteil der vereinigten

lokalen Konfigurationen von den Ereignissen aus ${}^\bullet({}^\bullet e)$. Mit diesem Ergebnis erhält man

$$e_1 = (t_1, \{b_1, b_2\} \cup \{b_9\}) \quad \text{und}$$
$$e_5 = (t_1, \{b_6, b_7\} \cup \{b_{10}\}).$$

Satz 3.3.4 (Komplemente im Präfix [ES01b])
Gegeben seien ein sicheres S/T-Netzsystem (N, M_0) mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $(N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, eine Stelle $s \in S$, deren Komplement $\bar{s} \in S$, eine Transition $t \in {}^\bullet s \setminus s^\bullet$ mit ${}^\bullet t = \{s_1, \ldots, s_n, \bar{s}\}$ sowie eine Menge $\{b_1, \ldots, b_n, \bar{b}\} \subseteq B$ mit $\lambda(b_i) = s_i$ für $1 \leq i \leq n$ und $\lambda(\bar{b}) = \bar{s}$. Dann gelten die folgenden zwei Eigenschaften:

(i) $\{b_1, \ldots, b_n, \bar{b}\}$ *Co-Menge* $\Rightarrow {}^\bullet\bar{b} \subseteq \bigcup_{1 \leq i \leq n} [{}^\bullet b_i] \vee \bar{b} \in Min(N')$

(ii) Bildet $\{b_1, \ldots, b_n\}$ eine Co-Menge, dann existiert höchstens ein Ereignis $\bar{b}$ mit $\lambda(\bar{b}) = \bar{s}$, sodaß $\{b_1, \ldots, b_n, \bar{b}\}$ eine Co-Menge bildet.

Beweis:

(i) Gegeben sei die Co-Menge $\{b_1, \ldots, b_n, \bar{b}\}$.

(a) $\bar{b} \in Min(N')$:
Dieser Fall ist trivial.

(b) $\bar{b} \notin Min(N')$:
Dann gibt es eine Bedingung $b \in {}^\bullet({}^\bullet\bar{b})$ mit $\lambda(b) = s$. Aus der Sicherheit des Netzsystems folgt, daß mindestens ein b_i $(1 \leq i \leq n)$ existiert, so daß $(b, b_i) \notin co$. Ohne Beschränkung der Allgemeinheit gelte $(b, b_1) \notin co$. Dann müssen b und b_1 jedoch entweder in Konflikt stehen oder kausal voneinander abhängen.

(b1) $b \# b_1$ (b und b_1 stehen in Konflikt):
Dann stehen b_1 und $\bar{b}$ ebenfalls in Konflikt, was jedoch einen Widerspruch zu $(b_1, \bar{b}) \in co$ ergibt.

(b2) $b_1 < b$ (b und b_1 sind kausal abhängig):
Daraus folgt $b_1 < \bar{b}$, was jedoch einen Widerspruch zu $(b_1, \bar{b}) \in co$ ergibt.

(b3) $b < b_1$ (b und b_1 sind kausal abhängig):
Hier müssen zwei Fälle betrachtet werden:

(b3i) es gibt zwei Ereignisse $e_1, e_2 \in b^\bullet$, so daß $e_1 \in [{}^\bullet b_1]$ and $e_2 \in {}^\bullet\bar{b}$.
Daraus folgt direkt $b_1 \# \bar{b}$, was jedoch im Widerspruch zu $(b_1, \bar{b}) \in co$ steht.

(b3ii) es gibt ein Ereignis $e \in b^\bullet$, so daß $e \in [{}^\bullet b_1]$ und $e \in {}^\bullet\bar{b}$.
Dieser Fall liefert das gewünschte Ergebnis.

(ii) Angenommen, es gibt zwei Bedingungen $\bar{b}, \bar{b}'$ mit $\lambda(\bar{b}) = \lambda(\bar{b}') = \bar{s}$ und zwei Co-Mengen $C_1 = \{b_1, \ldots, b_n, \bar{b}\}, C_2 = \{b_1, \ldots, b_n, \bar{b}'\}$.
Aus der Sicherheit des Netzsystems folgt unmittelbar $(\bar{b}, \bar{b}') \notin co$. Dann müssen sie jedoch in Konflikt stehen oder kausal voneinander abhängen.

(a) $\bar{b} \# \bar{b}'$ ($\bar{b}$ und $\bar{b}'$ stehen in Konflikt):
Dann gilt jedoch $\bar{b}, \bar{b}' \notin Min(N')$. Aus (i) folgt

$$^\bullet\bar{b} \subseteq \bigcup_{1 \le i \le n} [^\bullet b_i] \quad \text{und} \quad ^\bullet\bar{b}' \subseteq \bigcup_{1 \le i \le n} [^\bullet b_i].$$

(a1) $^\bullet\bar{b}, ^\bullet\bar{b}' \subseteq [^\bullet b_i]$:
Dieses ergibt sofort einen Widerspruch, da zwei in Konflikt stehende Ereignisse nicht zu derselben lokalen Konfiguration gehören können.

(a2) $^\bullet\bar{b} \subseteq [^\bullet b_i]$ und $^\bullet\bar{b}' \subseteq [^\bullet b_j]$ für $i \neq j$:
Dann stehen b_i und b_j in Konflikt, welches aber einen Widerspruch zu der Annahme darstellt, daß C_1 und C_2 Co-Mengen bilden.

(b) $\bar{b} < \bar{b}'$ ($\bar{b}$ und $\bar{b}'$ sind kausal abhängig):
Aus (i) folgt

$$^\bullet\bar{b} \subseteq \bigcup_{1 \le i \le n} [^\bullet b_i] \vee \bar{b} \in Min(N') \quad \text{und}$$
$$^\bullet\bar{b}' \subseteq \bigcup_{1 \le i \le n} [^\bullet b_i]$$

Desweiteren gibt es ein Ereignis e, sodaß $e \in \bar{b}^\bullet \cap [^\bullet\bar{b}']$. Dann gilt jedoch auch

$$e \in \bigcup_{1 \le i \le n} [^\bullet b_i]$$

und somit folgt unmittelbar $\bar{b} < b_i$ für ein i. Dieses steht jedoch im Widerspruch zu der Annahme, daß C_1 eine Co-Menge bildet.
(Der Fall $\bar{b}' < \bar{b}$ verhält sich analog).

■

3.3.4 Der Algorithmus

Der graphentheoretische Algorithmus CheckCo ist in Abbildung 3.15 auf der nächsten Seite dargestellt. Als Eingabe dienen ein sicheres S/T-Netzsystem (N, M_0) und eine Teilmarkierung

$$M_{par} = (\{s_1, \ldots, s_k\}, \{s_{k+1}, \ldots, s_n\}),$$

deren Erreichbarkeit überprüft werden soll. Dazu wird das Netzsystem zunächst in dessen endliches, vollständiges Präfix entfaltet. Anschließend wird in der anfangs gegebenen Teilmarkierung M_{par} jede Stelle $s_i \in \{s_1, \ldots, s_k\}$, die unter M_{par} keine Marke

Abbildung 3.15 *Graphentheoretischer Algorithmus* CheckCo

Algorithmus 1
Eingabe: Sicheres S/T-Netzsystem (N, M_0)
Teilmarkierung $M_{par} = (\{s_1, \ldots, s_k\}, \{s_{k+1}, \ldots, s_n\})$
Ausgabe: Ja/Nein

```
1  β := Präfix((N, M_0));
2  Ersetze M_par durch (∅, {s̄_1, ..., s̄_k, s_{k+1}, ..., s_n});
3  for all s̄_i ∈ {s̄_1, ..., s̄_k} do
4     FügeBedingungenEin(s̄_i, β);
5  od
6  for all b_i beschriftet mit s̄_i ∈ {s̄_1, ..., s̄_k} do
7     BerechneCoRelation(b_i);
8  od
9  for all s_i ∈ {s̄_1, ..., s̄_k, s_{k+1}, ..., s_n} do
10    Berechne(B_i); /* ∀b_{i_j} ∈ B_i : λ(b_{i_j}) = s_i */
11 od
12 L := ∅;
13 Check(1);
14 ausgabe(Nein);
```

enthält, d.h., $M_{par}(s_i) = 0$, durch deren Komplement $\overline{s_i}$ ersetzt, für das dann folglich $M_{par}(\overline{s_i}) = 1$ gilt. Für jedes dieser Komplemente $\overline{s_1}, \ldots, \overline{s_k}$ werden dann nach der im vorherigen Abschnitt beschriebenen Vorgehensweise die entsprechenden Bedingungen in das Präfix eingefügt. Danach wird für die neu hinzugekommenen Bedingungen die Co-Relation berechnet. Effiziente Verfahren für die Berechnung der Co-Relation sind in [Röm00] beschrieben worden. Schließlich wird für jede Stelle

$$s_i \in \{\overline{s_1}, \ldots, \overline{s_k}, s_{k+1}, \ldots, s_n\}$$

die Menge B_i ihrer Repräsentanten ermittelt und nach einer Co-Menge der Größe n gesucht, die jeweils genau ein Element aus jeder Menge B_i enthält. Die Suche nach einer solchen Co-Menge wird innerhalb der Prozedur *Check* aus Abbildung 3.16 auf der nächsten Seite realisiert.
Der Algorithmus soll nun anhand eines Beispiels näher erläutert werden. Dazu werden wiederum das Netzsystem aus Abbildung 3.4 auf Seite 49 und die Teilmarkierung

$$M_{par} = (\{s_5\}, \{s_2, s_4\})$$

betrachtet. Zunächst wird das Netzsystem in dessen endliches, vollständiges Präfix entfaltet, das in Abbildung 3.5 auf Seite 50 dargestellt ist. Dann wird M_{par} in die Teilmarkierung

$$M'_{par} = (\emptyset, \{\overline{s_5}, s_2, s_4\})$$

Abbildung 3.16 *Prozedur Check*

Algorithmus 2

```
Proc Check(int i)
 1  j := 1;
 2  while j ≤ |B_i| do
 3     L := L ∪ {b_{i_j}}
 4     if L ist Co-Menge then
 5        if |L| = |S^0 ∪ S^1| then
 6           ausgabe(Ja);
 7           exit;
 8        else
 9           Check(i+1);
10        fi
11     fi
12     L := L \ {b_{i_j}}
13     j := j + 1;
14  od
```

überführt, für die jetzt die Erreichbarkeit nachgewiesen werden soll. Nun müssen entsprechende Bedingungen für die Stelle $\overline{s_5}$ in das Präfix eingefügt werden. Da allerdings die Stelle s_6 bereits das Komplement zu s_5 bildet, d.h., $s_6 = \overline{s_5}$, brauchen jetzt keine weiteren Bedingungen eingefügt zu werden. Nun müssen die Mengen B_i für $1 \leq i \leq 3$ berechnet werden. Mit

$$\begin{aligned}\overline{s_5} = s_6 &= \lambda(b_4) = \lambda(b_9) = \lambda(b_{12})\\ s_2 &= \lambda(b_7)\\ s_4 &= \lambda(b_5) = \lambda(b_{10})\end{aligned}$$

ergeben sich die B_i zu

$$\begin{aligned}B_1 &= \{b_4, b_9, b_{12}\}\\ B_2 &= \{b_7\}\\ B_3 &= \{b_5, b_{10}\}\end{aligned}$$

In Tabelle 3.1 auf der nächsten Seite werden die Aufrufreihenfolge für die rekursive Prozedur *Check* mit den jeweiligen Variablenbelegungen sowie der jeweils aktuelle Inhalt der Lösungsmenge L dargestellt. Mit dem Aufruf von *Check*(1) ergibt sich $L = \{b_4\}$, und es wird rekursiv mit dem Aufruf *Check*(2) fortgefahren. Es folgt $L = \{b_4, b_7\}$, aber da $(b_4, b_7) \notin co$, stellt L keine Co-Menge dar. b_7 wird aus L entfernt, und da in B_2 keine weiteren Elemente vorkommen, wird der *Check*(2)-Aufruf beendet, zu *Check*(1) zurückgekehrt und das nächste Element aus B_1 getestet ($L = \{b_9\}$). Ein erneuter Aufruf

Tabelle 3.1 *Variablenbelegungen für die Prozedur Check und der jeweilige Inhalt der Lösungsmenge L*

i	j	L
1	1	$\{b_4\}$
2	1	$\{b_4, b_7\}$
1	2	$\{b_9\}$
2	1	$\{b_9, b_7\}$
3	1	$\{b_9, b_7, b_5\}$
3	2	$\{b_9, b_7, b_{10}\}$
1	3	$\{b_{12}\}$
2	1	$\{b_{12}, b_7\}$
3	1	$\{b_{12}, b_7, b_5\}$
3	2	$\{b_{12}, b_7, b_{10}\}$

von *Check*(2) liefert $L = \{b_9, b_7\}$. Da $(b_9, b_7) \in co$ und L somit eine Co-Menge bildet, wird mit dem Aufruf *Check*(3) fortgefahren, was die Hinzunahme des Elements b_5 zu L bewirkt. Aus $(b_7, b_5) \notin co$ folgt, daß $L = \{b_9, b_7, b_5\}$ keine Co-Menge darstellt. Daraufhin wird b_5 wieder aus L entfernt, und es wird das nächste Element aus B_3 hinzugefügt. Es gilt jetzt $L = \{b_9, b_7, b_{10}\}$, aber aufgrund von $(b_9, b_{10}) \notin co$ ergibt L auch diesmal keine Co-Menge. Die Prozeduraufrufe *Check*(3) und *Check*(2) terminieren und führen das Programm auf *Check*(1)-Ebene zurück. Nun wird $L = \{b_{12}\}$ erzeugt und erneut der Aufruf *Check*(2) ausgeführt. Daraus ergibt sich dann $L = \{b_{12}, b_7\}$, welches auch eine Co-Menge darstellt. Anschließend wird *Check*(3) ausgeführt, und man erhält $L = \{b_{12}, b_7, b_5\}$. Da aber $(b_7, b_5) \notin co$, wird als nächstes die Menge $L = \{b_{12}, b_7, b_{10}\}$ gebildet. Diese Menge ergibt jetzt die gewünschte Co-Menge, und der Algorithmus kann mit der Antwort terminieren, daß die Teilmarkierung erreichbar ist.

3.3.5 Komplexitätsbetrachtung

In diesem Abschnitt soll die Komplexität von CheckCo diskutiert werden. Gegeben seien ein sicheres S/T-Netzsystem (N, M_0) mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $(N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, die Co-Relation $co \subseteq B \times B$ sowie eine Teilmarkierung (S^0, S^1) mit $S^0, S^1 \subseteq S$. Jede Stelle der Menge $S^0 \cup S^1$ besitzt höchstens $|B|$ Repräsentanten im Präfix, und somit ergeben sich höchstens

$$|B|^{|S^0 \cup S^1|}$$

mögliche Co-Mengen. Desweiteren werden höchstens

$$|S^0 \cup S^1|^2$$

Vergleiche benötigt, um zu überprüfen, ob die Elemente einer möglichen Lösungsmenge alle paarweise nebenläufig sind, d.h., in Co-Relation stehen. Das Einfügen der Bedingungen für die komplementären Stellen und die Berechnung der Co-Relation können in Zeit

$$\mathcal{O}(|B|^2 \cdot |E|)$$

vorgenommen werden. Insgesamt führt dieses zu einer Zeitkomplexität von

$$\mathcal{O}(|B|^{|S^0 \cup S^1|} \cdot |S^0 \cup S^1|^2 + |B|^2 \cdot |E|).$$

Da $|B|$ und $|E|$ gewöhnlich viel größer als $|S^0 \cup S^1|$ sind, bedeutet dieses Ergebnis eine signifikante Verbesserung im Vergleich zu den in den Abschnitten 3.4.1 (CheckLin) und 3.4.2 auf Seite 71 (McSmodels) beschriebenen Verfahren, die beide exponentielle Komplexität in der Größe des Präfixes aufweisen.[8]

3.4 Weitere Algorithmen für Erreichbarkeit

3.4.1 Lineare Programmierung

Das in diesem Abschnitt betrachtete Verfahren CheckLin wurde von Melzer [Mel98] entwickelt. Als grundlegendes Konzept dient hier die sogenannte *Markierungsgleichung*, welche als algebraische Repräsentation für die Menge der erreichbaren Markierungen eines azyklischen Netzes, insbesondere also eines Präfixes, verstanden werden kann.[9] Die Markierungsgleichung läßt sich folgendermaßen beschreiben:

$$M(s) = M_0(s) + \sum_{t \in {}^\bullet s} \#t^\sigma - \sum_{t \in s^\bullet} \#t^\sigma,$$

wobei $\#t^\sigma$ die Anzahl des Auftretens der Transition t in der Schaltfolge σ bedeutet, die M_0 nach M überführt, d.h., $M_0 \xrightarrow{\sigma} M$. Allgemein gesagt berechnet sie für eine von der Anfangsmarkierung M_0 aus erreichbare Markierung M und jede Stelle s die Anzahl an Marken, die s unter M enthält.[10] Gewöhnlich wird die Markierungsgleichung in Matrixschreibweise

$$M = M_0 + \mathbf{N} \cdot \vec{\sigma}$$

notiert, wobei

$$\vec{\sigma} = (\#t_1^\sigma, \ldots, \#t_n^\sigma)^t$$

[8] Im Gegensatz zu CheckCo gehen CheckLin und McSmodels nicht davon aus, daß die Co-Relation des endlichen, vollständigen Präfixes vorliegt, weshalb die Komplexitätsschranke von CheckCo besser wird. Es wäre jedoch möglich, diese Information so in CheckLin und McSmodels einfließen zu lassen, daß sie auf dieselbe obere Schranke für die Laufzeit kommen.

[9] Für den Fall azyklischer Netze stellt die Markierungsgleichung ein hinreichendes Kriterium für Erreichbarkeit dar [Mur89]. Im Gegensatz dazu liefert sie im Fall zyklischer Netze lediglich ein notwendiges Kriterium .

[10] Bei sicheren Netzsystemen kann die Anzahl der Marken auf den Stellen nur 0 oder 1 sein.

als *Parikhvektor* von σ und $\mathbf{N}$ als *Inzidenzmatrix* bezeichnet werden. $\mathbf{N}$ beschreibt eine $(S \times T)$-Matrix, die durch

$$\mathbf{N}(s,t) = \begin{cases} 1 & \text{falls } t \in {}^\bullet s \setminus s^\bullet \\ -1 & \text{falls } t \in s^\bullet \setminus {}^\bullet s \\ 0 & \text{sonst} \end{cases}$$

gegeben ist.
Die Erreichbarkeit einer Teilmarkierung (S^0, S^1) kann nun nachgewiesen werden, indem die Markierungsgleichung um die Ungleichungen

$$\begin{aligned} M(s) &\leq 0 \quad \text{für jede Stelle } s \in S^0 \\ M(s) &\geq 1 \quad \text{für jede Stelle } s \in S^1 \end{aligned}$$

erweitert wird. Allgemein können auch diese Ungleichungen in Matrixschreibweise als

$$\mathbf{A} \cdot M \geq b$$

dargestellt werden, wobei $\mathbf{A}$ eine $((S^0 \cup S^1) \times S)$-Matrix beschreibt, die durch

$$\mathbf{A}(s,s') = \begin{cases} 1 & \text{falls } s = s' \text{ und } s \in S^1 \\ -1 & \text{falls } s = s' \text{ und } s \in S^0 \\ 0 & \text{sonst} \end{cases}$$

gegeben ist.
Für azyklische Netzsysteme gilt, daß die Teilmarkierung (S^0, S^1) genau dann erreichbar ist, wenn das so erhaltene Ungleichungssystem eine Lösung für M und $\vec{\sigma}$ besitzt.

Satz 3.4.1 (Erreichbarkeit auf Präfixen)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, sowie eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$. M_{par} ist genau dann von M_0 aus erreichbar, wenn das lineare Ungleichungssystem

$$\begin{aligned} &\textit{Variablen: } M', X \textit{ binär} \\ &M' = Min(N') + \mathbf{N}' \cdot X \\ &\lambda(\boldsymbol{A}) \cdot M' \geq b \end{aligned}$$

eine Lösung für M' und X besitzt. Dabei bezeichne $\mathbf{N}'$ die Inzidenzmatrix von β und $\lambda(\boldsymbol{A})$ eine $((S^0 \cup S^1) \times B)$-Matrix, die durch

$$\lambda(\boldsymbol{A})(s,b) = \begin{cases} 1 & \textit{falls } \lambda(b) = s \textit{ und } s \in S^1 \\ -1 & \textit{falls } \lambda(b) = s \textit{ und } s \in S^0 \\ 0 & \textit{sonst} \end{cases}$$

gegeben ist.

BEWEIS: Direkte Folgerung von Satz 3.7 aus [Mel98]. ■

Anhand des Beispielnetzes aus Abbildung 3.4 auf Seite 49 wird nun für die Teilmarkierung

$$M_{par} = (\{s_5\}, \{s_2, s_4\})$$

gezeigt, wie obiges Verfahren abläuft. Es wird nach einer erreichbaren Markierung M gesucht, die jeweils eine Marke auf die Stellen s_2 und s_4, aber keine Marke auf s_5 legt. Zunächst wird für jede Stelle der Teilmarkierung M_{par} eine entsprechende Ungleichung aufgestellt, d.h.,

$$\begin{aligned} M(s_5) &\leq 0 \\ M(s_2) &\geq 1 \\ M(s_4) &\geq 1 \end{aligned}$$

Diese können unter Verwendung der Beschriftungsfunktion λ direkt auf eine Markierung M' des Präfixes übertragen werden. Mit

$$\begin{aligned} \lambda(b_3) = \lambda(b_6) = \lambda(b_{11}) &= s_5 \\ \lambda(b_7) &= s_2 \\ \lambda(b_5) = \lambda(b_{10}) &= s_4 \end{aligned}$$

folgt unmittelbar

$$\begin{aligned} M'(b_3) + M'(b_6) + M'(b_{11}) &\leq 0 \\ M'(b_7) &\geq 1 \\ M'(b_5) + M'(b_{10}) &\geq 1 \end{aligned}$$

Nun wird für jede Bedingung des Präfixes die Markierungsgleichung hinzugefügt. Für die Bedingung b_4 beispielsweise ergibt sich die Gleichung

$$M'(b_4) = X(e_1) - X(e_2) - X(e_3).$$

Schließlich können noch alle Terminalereignisse eliminiert werden, da nach Definition 3.1.10 auf Seite 45 alle erreichbaren Markierungen ohne das Schalten eines Terminalereignisses erhalten werden können. Folglich wird das Ungleichungssystem für jedes Terminalereignis e um die Gleichung

$$X(e) = 0$$

erweitert. Das vollständige Ungleichungssystem ist in Abbildung 3.17 auf der nächsten Seite dargestellt, wobei die Gleichungen auf der linken Seite die Markierungsgleichung des Präfixes beschreiben. Das Ungleichungssystem besitzt eine Lösung für

$$M' = \{b_7, b_{10}, b_{12}\} \quad \text{und} \quad X = (1,1,0,1,0,1,0,0,1,0,1,0)^t,$$

woraus unmittelbar die Erreichbarkeit der Teilmarkierung $(\{s_5\}, \{s_2, s_4\})$ folgt.

Abbildung 3.17 *Lineares Ungleichungssystem nach Satz 3.4.1 auf Seite 69 für das Präfix aus Abbildung 3.5 auf Seite 50 und die Teilmarkierung* $(\{s_5\}, \{s_2, s_4\})$

$$
\begin{aligned}
M'(b_1) &= 1 - X(e_4) \\
M'(b_2) &= 1 - X(e_2) \\
M'(b_3) &= 1 - X(e_1) \\
M'(b_4) &= X(e_1) - X(e_2) - X(e_3) \\
M'(b_5) &= X(e_2) - X(e_4) - X(e_5) \\
M'(b_6) &= X(e_2) - X(e_6) \\
M'(b_7) &= X(e_4) - X(e_7) \\
M'(b_8) &= X(e_4) - X(e_9) \\
M'(b_9) &= X(e_6) - X(e_8) - X(e_9) \\
M'(b_{10}) &= X(e_9) - X(e_{10}) \\
M'(b_{11}) &= X(e_9) - X(e_{11}) \\
M'(b_{12}) &= X(e_{11}) - X(e_{12})
\end{aligned}
\qquad
\begin{aligned}
M'(b_3) + M'(b_6) + M'(b_{11}) &\leq 0 \\
M'(b_7) &\geq 1 \\
M'(b_5) + M'(b_{10}) &\geq 1 \\
X(e_3) &= 0 \\
X(e_5) &= 0 \\
X(e_7) &= 0 \\
X(e_8) &= 0 \\
X(e_{10}) &= 0 \\
X(e_{12}) &= 0
\end{aligned}
$$

Komplexitätsanalyse

Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ sowie eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$. Durch die Eliminierung der Nicht-Terminalereignisse reduziert sich die Anzahl der binären Variablen des nach Satz 3.4.1 auf Seite 69 erstellten Ungleichungssystems. Von daher löst der Algorithmus CheckLin das Erreichbarkeitsproblem in Zeit

$$\mathcal{O}(2^{|E \setminus CutOffs(E)|}).$$

3.4.2 Logikprogrammierung

Das in diesem Abschnitt vorgestellte Verfahren McSmodels wurde von Heljanko entwickelt [Hel99, Hel02]. Die Hauptidee dieses Ansatzes besteht darin, das Erreichbarkeitsproblem in ein *Logikprogramm* zu überführen und anschließend zu testen, ob dieses ein *stabiles Modell* [GL88] besitzt. Dafür wird ein Programm namens Smodels [NS97, Sim00] eingesetzt, welches eine Umgebung für constraint-basierte Logikprogrammierung [Nie99] darstellt. Grundsätzlich besteht ein Logikprogramm aus Regeln der Form

$$a \leftarrow b_1, \ldots, b_n, not(c_1), \ldots, not(c_m), \tag{3.4.1}$$

wobei $a, b_1, \ldots, b_n, c_1, \ldots, c_m$ propositionelle Atome bezeichnen. Diese Regeln werden als Constraints für die Lösungsmenge des Programms interpretiert und bedeuten, daß ein

Atom a Bestandteil der Lösungsmenge sein muß, falls die Atome $b_1, \ldots, b_n$ ebenfalls in der Lösungsmenge enthalten sind, jedoch die Atome $c_1, \ldots, c_m$ nicht dazugehören. Eine natürliche Definitionsmöglichkeit für solche Lösungsmengen ist durch das Konzept der sogenannten stabilen Modelle gegeben, die eine der führenden deklarativen Semantiken für Logikprogramme darstellen. Daher wird das Konzept der stabilen Modellsemantik im folgenden näher vorgestellt.

Definition 3.4.1 (Reduktion P^A eines Logikprogramms P [GL88])
Gegeben seien ein Logikprogramm P und eine Menge A von Atomen. Die Reduktion P^A von P bezüglich der Menge A beschreibt das Programm, welches aus P durch Streichen

- jeder Regel, die für ein Atom $a \in A$ ein *not*-Atom $not(a)$ enthält, und
- aller *not*-Atome in den restlichen Regeln

gewonnen wird.

Definition 3.4.2 (Stabiles Modell eines Logikprogramms [GL88])
Gegeben seien ein Logikprogramm P und eine Menge A von Atomen. A wird genau dann als stabiles Modell von P bezeichnet, wenn A der deduktiven Hülle von P^A entspricht, sofern die Regeln von P^A als Inferenzregeln verstanden werden.

Dieses Konzept soll nun anhand zweier Beispiele erläutert werden. Zunächst wird das Logikprogramm P_1 betrachtet:

$$a \leftarrow b, not(c)$$
$$c \leftarrow not(a)$$
$$d \leftarrow c$$

Es soll nun getestet werden, ob die Menge $\{c, d\}$ ein stabiles Modell für P_1 darstellt. Dazu wird unter Verwendung von Definition 3.4.1 die Reduktion $P^{\{c,d\}}$ gebildet:

$$c \leftarrow$$
$$d \leftarrow c$$

Die deduktive Hülle von $P^{\{c,d\}}$ entspricht genau der Menge $\{c, d\}$, und somit ist nach Definition 3.4.2 erfolgreich ein stabiles Modell für P_1 ermittelt worden. Im Gegensatz dazu stellt beispielsweise die Menge $\{a, b\}$ kein stabiles Modell von P_1 dar, denn für die Reduktion $P^{\{a,b\}} =$

$$a \leftarrow b$$
$$d \leftarrow c$$

erhält man die leere Menge als deren deduktive Hülle. Als weiteres Beispiel soll noch das Logikprogramm P_2 betrachtet werden, da diesem an späterer Stelle noch eine besondere

Bedeutung zuteil wird. P_2 ist wie folgt definiert:

$$f \leftarrow g, not(f)$$
$$g \leftarrow$$

Offensichtlich muß jedes stabile Modell von P_2 das Atom g enthalten. Die deduktive Hülle der Reduktion $P^{\{g\}} =$

$$f \leftarrow g$$
$$g \leftarrow$$

ergibt sich zu $\{f, g\}$, weshalb $\{g\}$ kein stabiles Modell von P_2 darstellt. Andererseits bildet die Menge $\{f, g\}$ auch kein stabiles Modell von P_2, da die Reduktion $P^{\{f,g\}} =$

$$g \leftarrow$$

die Menge $\{g\}$ als deduktive Hülle besitzt. Folglich existiert für P_2 kein stabiles Modell. Neben den herkömmlichen Logikprogrammregeln der Form 3.4.1 auf Seite 71 kann SMODELS [Sim00] noch zwei weitere Regelarten verarbeiten. Es gibt Regeln der Art

$$a \leftarrow 2\{b_1, \ldots, b_n\},$$

deren Semantik besagt, daß das Atom a zu dem Modell gehört, falls mindestens zwei oder mehrere Atome der Menge $\{b_1, \ldots, b_n\}$ in dem Modell enthalten sind. Desweiteren ist es zulässig, Integritätsregeln der Art

$$\leftarrow a_1, \ldots, a_n, not(b_1), \ldots, not(b_n) \tag{3.4.2}$$

anzugeben. Die Semantik dieser Regeln kann folgendermaßen aufgefaßt werden: Es werden zwei neue Atome f und g sowie die Regel

$$f \leftarrow g, not(f)$$

hinzugefügt. Dann wird jede Regel der Form 3.4.2 durch die Regel

$$g \leftarrow a_1, \ldots, a_n, not(b_1), \ldots, not(b_n)$$

ersetzt. Wegen des Beispielprogramms P_2 ist unmittelbar klar, daß das Atom g zu keinem stabilen Modell gehören kann. Daraus läßt sich schließen, daß es auch kein stabiles Modell geben kann, sodaß alle $a_1, \ldots, a_n$, aber keines der $b_1, \ldots, b_n$ dazugehören. Durch die Verwendung von Integritätsregeln können niemals zusätzliche stabile Modelle erzeugt werden, aber es können unerwünschte stabile Modelle ausgeschlossen werden.
Im folgenden wird nun ein Logikprogramm definiert, das zur Überprüfung der Erreichbarkeit von Teilmarkierungen eingesetzt werden kann.

Definition 3.4.3 (Logikprogramm $P_R(\beta, M_{par})$ [Hel99, Hel02])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ sowie eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$. Das Logikprogramm $P_R(\beta, M_{par})$ besteht aus folgenden Regeln:

1. $\forall e_i \in E \setminus \mathit{CutOffs}(E)$:
   $e_i \leftarrow e_k, \ldots, e_n, not(be_i)$,
   sodaß $\{e_k, \ldots, e_n\} = {}^\bullet({}^\bullet e_i)$

2. $\forall e_i \in E \setminus \mathit{CutOffs}(E)$:
   $be_i \leftarrow not(e_i)$

3. $\forall b_i \in B$ mit $|b_i^\bullet \setminus \mathit{CutOffs}(E)| \geq 2$:
   $\leftarrow 2\{e_k, \ldots, e_n\}$,
   sodaß $\{e_k, \ldots, e_n\} = b_i^\bullet \setminus \mathit{CutOffs}(E)$

4. $\forall b_i \in \{b \in B \mid \lambda(b) \in S^0 \cup S^1 \wedge {}^\bullet b \in E \setminus \mathit{CutOffs}(E)\}$:
   $b_i \leftarrow e_i, not(e_k), \ldots, not(e_n)$,
   sodaß $\{e_i\} = {}^\bullet b_i$ und $\{e_k, \ldots, e_n\} = b_i^\bullet \setminus \mathit{CutOffs}(E)$

5. $\forall b_i \in \{b \in B \mid \lambda(b) \in S^0 \cup S^1 \wedge {}^\bullet b \in E \setminus \mathit{CutOffs}(E)\}$:
   $s_i \leftarrow b_i$,
   sodaß $s_i = \lambda(b_i)$

6. $\forall s_i \in S^1$:
   $\leftarrow not(s_i)$

7. $\forall s_i \in S^0$:
   $\leftarrow s_i$

Mit den Regeln vom Typ 1 und 2 wird garantiert, daß ein Ereignis e nur dann zum Modell gehören kann (aber nicht notwendigerweise muß), falls dessen unmittelbaren kausalen Vorgängerereignisse im Modell enthalten sind. Die Regeln vom Typ 3 stellen sicher, daß ein Modell keine in Konflikt stehenden Ereignisse beinhaltet. Die Regeln vom Typ 4 besagen, daß eine Bedingung b Bestandteil des Modells ist, falls deren Eingangsereignis, aber keins der Ausgangsereignisse von b zum Modell gehört. Die Beschriftungsfunktion λ wird mit den Regeln vom Typ 5 kodiert. Schließlich garantieren die Regeln vom Typ 6 und 7, daß jede Stelle aus S^1, aber keine Stelle aus S^0 in dem Modell enthalten ist.

Satz 3.4.2 (Erreichbarkeit von Teilmarkierungen [Hel99, Hel02])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$ sowie das nach Definition 3.4.3 erhaltene

Abbildung 3.18 *Logikprogram $P_R(\beta, M_{par})$ nach Definition 3.4.3 auf der vorherigen Seite für das Präfix aus Abbildung 3.5 auf Seite 50 und die Teilmarkierung $(\{s_5\}, \{s_2, s_4\})$*

$e_1 \leftarrow not(be_1)$	$be_1 \leftarrow not(e_1)$	$b_3 \leftarrow not(e_1)$	$s_5 \leftarrow b_3$	$\leftarrow not(s_2)$
$e_2 \leftarrow e_1, not(be_2)$	$be_2 \leftarrow not(e_2)$	$b_5 \leftarrow e_2, not(e_4)$	$s_5 \leftarrow b_6$	$\leftarrow not(s_4)$
$e_4 \leftarrow e_2, not(be_4)$	$be_4 \leftarrow not(e_4)$	$b_6 \leftarrow e_2, not(e_6)$	$s_5 \leftarrow b_{11}$	$\leftarrow s_5$
$e_6 \leftarrow e_2, not(be_6)$	$be_6 \leftarrow not(e_6)$	$b_7 \leftarrow e_4$	$s_2 \leftarrow b_7$	
$e_9 \leftarrow e_4, e_6, not(be_9)$	$be_9 \leftarrow not(e_9)$	$b_{10} \leftarrow e_9$	$s_4 \leftarrow b_5$	
$e_{11} \leftarrow e_9, not(be_{11})$	$be_{11} \leftarrow not(e_{11})$	$b_{11} \leftarrow e_9, not(e_{11})$	$s_4 \leftarrow b_{10}$	

Logikprogramm $P_R(\beta, M_{par})$. $P_R(\beta, M_{par})$ besitzt genau dann ein stabiles Modell, wenn in Σ eine erreichbare Markierung existiert, die M_{par} überdeckt, d.h.,

$$P_R(\beta, M_{par}) \text{ hat stabiles Modell } \Leftrightarrow \exists M \in \mathcal{R}(M_0) \colon \forall s \in S^0 \cup S^1 \colon M(s) = M_{par}(s)$$

Zusätzlich gilt für jedes stabile Modell Δ von $P_R(\beta, M_{par})$, daß

$$C = \{e \in E \mid e \in \Delta\}$$

eine Konfiguration von β bildet, für die $Mark(C)$ eine erreichbare Markierung von Σ darstellt, welche M_{par} überdeckt.

Abbildung 3.18 zeigt das Logikprogramm $P_R(\beta, M_{par})$, welches nach Definition 3.4.3 auf der vorherigen Seite für das Präfix β aus Abbildung 3.5 auf Seite 50 und die Teilmarkierung

$$M_{par} = (\{s_5\}, \{s_2, s_4\})$$

aufgestellt worden ist. $P_R(\beta, M_{par})$ besitzt das stabile Modell

$$\Delta = \{s_2, s_4, b_7, b_{10}, e_1, e_2, e_4, e_6, e_9, e_{11}\}.$$

Die Menge

$$C = \{e \in E \mid e \in \Delta\} = \{e_1, e_2, e_4, e_6, e_9, e_{11}\}$$

bildet eine Konfiguration von β, und

$$Mark(C) = \{s_2, s_4, s_6\}$$

stellt eine erreichbare Markierung M in dem Netz aus Abbildung 3.4 auf Seite 49 dar, welche die Eigenschaft

$$\forall s \in \{s_5, s_2, s_4\} \colon M(s) = M_{par}(s)$$

erfüllt. Somit konnte die Erreichbarkeit der Teilmarkierung

$$M_{par} = (\{s_5\}, \{s_2, s_4\})$$

nach Satz 3.4.2 auf der vorherigen Seite erfolgreich nachgewiesen werden.

Komplexitätsanalyse

Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, dessen endliches, vollständiges Präfix $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$, eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$ sowie das nach Definition 3.4.3 auf Seite 74 erhaltene Logikprogramm $P_R(\beta, M_{par})$. $P_R(\beta, M_{par})$ weist für jede Bedingung, jedes Nicht-Terminalereignis und jede Stelle der Teilmarkierung eine Variable auf. Daher löst McSmodels das Erreichbarkeitsproblem in Zeit

$$\mathcal{O}(2^{|B|+|E\setminus CutOffs(E)|+|S^0\cup S^1|}).$$

3.4.3 On-the-Fly-Verifikation

Bislang verwenden alle vorgestellten Verfahren ein endliches, vollständiges Präfix als Eingabe. Ein Nachteil dieser Methoden besteht jedoch darin, daß zunächst das Präfix vollständig berechnet werden muß, bevor mit der eigentlichen Verifikation begonnen werden kann. Deshalb soll nun eine Methode erwähnt werden, die den Nachweis der Erreichbarkeit einer Teilmarkierung bereits während der Präfixerzeugung ermöglicht. Nach Satz 3.3.3 auf Seite 59 kann die Erreichbarkeit für eine Teilmarkierung

$$(\{s_1, \ldots, s_k\}, \{s_{k+1}, \ldots, s_n\})$$

entschieden werden, indem die zu $s_1, \ldots, s_k$ komplementären Stellen $\overline{s_1}, \ldots, \overline{s_k}$ in das Netzsystem eingefügt werden und die Erreichbarkeitsfrage für die Teilmarkierung

$$(\emptyset, \{\overline{s_1}, \ldots, \overline{s_k}, s_{k+1}, \ldots, s_n\})$$

gestellt wird. McMillan [McM92] hat eine Möglichkeit beschrieben, wie die Erreichbarkeit einer solchen Teilmarkierung bereits während der Entfaltung des Netzsystems in dessen Präfix nachgewiesen werden kann. Dazu wird eine zusätzliche Transition t' in das Netzsystem eingefügt, deren Vorbereich aus der zu verifizierenden Teilmarkierung besteht und deren Nachbereich leer ist, d.h.,

$${}^\bullet t' = \{\overline{s_1}, \ldots, \overline{s_k}, s_{k+1}, \ldots, s_n\} \quad \text{und} \quad t'^\bullet = \emptyset.$$

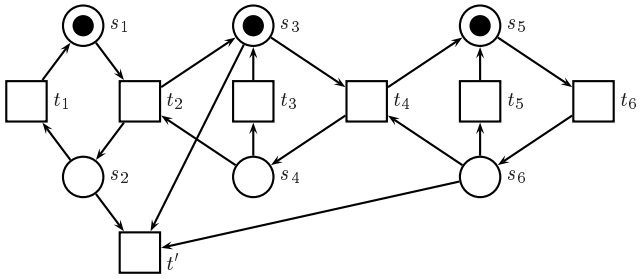
Nun wird das Netzsystem mit den beispielsweise von McMillan [McM92] oder Römer [ERV02, Röm00] beschriebenen Verfahren entfaltet. Die Berechnung des Präfixes kann beendet werden, sobald ein Ereignis e mit

$$\lambda(e) = t'$$

eingefügt werden kann, denn der Vorbereich von e besteht aus einer Co-Menge von Bedingungen, welche die Teilmarkierung

$$(\emptyset, \{\overline{s_1}, \ldots, \overline{s_k}, s_{k+1}, \ldots, s_n\})$$

Abbildung 3.19 *Modifiziertes S/T-Netzsystem für die On-the-Fly-Verifikation der Teilmarkierung* $(\emptyset, \{s_2, s_3, s_6\})$



repräsentieren. Mit Satz 3.3.1 auf Seite 56 folgt dann unmittelbar deren Erreichbarkeit. Für den Fall, daß die Teilmarkierung nicht erreichbar ist, wird das Netzsystem vollständig entfaltet, wobei das Präfix keinen Repräsentanten für t' enthält.
Anhand des Netzsystems aus Abbildung 3.4 auf Seite 49 soll erläutert werden, wie die Erreichbarkeit der Teilmarkierung

$$M_{par} = (\{s_5\}, \{s_2, s_3\})$$

während der Präfixbildung nachgewiesen werden kann. Der erste Schritt besteht darin, die Erreichbarkeitsfrage für die Teilmarkierung $(\emptyset, \{\overline{s_5}, s_2, s_3\})$ bzw. wegen $s_6 = \overline{s_5}$ für

$$M'_{par} = (\emptyset, \{s_6, s_2, s_3\})$$

zu betrachten. Als nächstes wird die Transition t' mit

$${}^\bullet t' = \{s_6, s_2, s_3\} \quad \text{und} \quad t'^\bullet = \emptyset$$

in das Netzsystem eingefügt. Das auf diese Weise modifizierte Netzsystem ist in Abbildung 3.19 dargestellt. Abbildung 3.20 auf der nächsten Seite zeigt das Präfix, welches während der On-the-Fly-Verifikation erzeugt wird. Der Entfaltungsvorgang wird mit dem Einfügen des Ereignisses e_7 als Repräsentant für die Transition t' beendet, da dessen Vorbereich von der Co-Menge $\{b_9, b_7, b_8\}$ gebildet wird, die ihrerseits wiederum die Menge $\{s_6, s_2, s_3\}$ repräsentiert. Somit konnte die Erreichbarkeit für die Teilmarkierungen

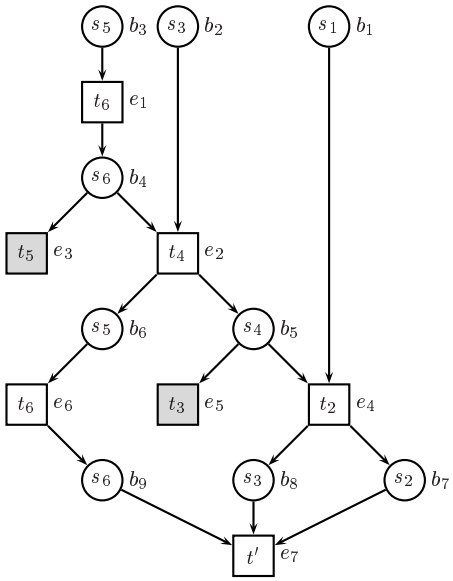
$$(\emptyset, \{s_6, s_2, s_3\}) \quad \text{und} \quad (\{s_5\}, \{s_2, s_3\})$$

erfolgreich nachgewiesen werden.

Komplexitätsanalyse

Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine Teilmarkierung (S^0, S^1). Desweiteren bezeichne $N' = (S', T', F')$ das Netz, welches

Abbildung 3.20 *Präfix für die On-the-Fly-Verifikation des Netzsystems aus Abbildung 3.19 auf der vorherigen Seite*



durch Einfügen der Komplemente zu den Stellen aus S^0 und einer Transition mit den Komplementen und den Stellen aus S^1 im Vorbereich entstanden ist. Die Hauptkosten während der Entfaltung des Netzsystems fallen für die Berechnung der möglichen Erweiterungen an. Dabei muß im ungünstigsten Fall für jede Transition t jede Teilmenge $B' \subseteq B$ von Bedingungen daraufhin überprüft werden, ob sie den Vorbereich von t repräsentiert. Es müssen also höchstens

$$\mathcal{O}\left(|T| \cdot \left(\frac{|B|}{\xi}\right)^{\xi}\right)$$

viele Teilmengen betrachtet werden, falls ξ die maximale Größe der Vor- oder Nachbereiche der Transitionen bezeichnet. Nun muß noch eine geeignete Abschätzung für die Anzahl $|B|$ von Bedingungen ermittelt werden. Im Falle sicherer Netzsysteme und bei Verwendung einer adäquaten Ordnung ist die Anzahl der Nicht-Terminalereignisse durch die Anzahl $|\mathcal{R}(M_0)|$ von erreichbaren Markierungen begrenzt. Durch das Einfügen eines Nicht-Terminalereignisses können höchstens ξ neue Bedingungen entstehen, weshalb

$$|B| \leq |\mathcal{R}(M_0)| \cdot \xi + |S|.$$

Da $|S|$ im Vergleich zu $|\mathcal{R}(M_0)|$ normalerweise vernachlässigbar klein ist, ergibt sich

$$\mathcal{O}(|T| \cdot |\mathcal{R}(M_0)|^{\xi})$$

als Abschätzung für die Lösung des Erreichbarkeitsproblems mit OnTheFly.

3.5 Experimenteller Vergleich der Algorithmen

In diesem Abschnitt werden die in den vorherigen Abschnitten vorgestellten Verfahren anhand von Testergebnissen miteinander verglichen. Dabei soll ergründet werden, für welche Teilmarkierungen die Verwendung des einen oder anderen Verfahrens geeigneter zu sein scheint als die Benutzung der übrigen Methoden.
Als Testbeispiele diente eine repräsentative Auswahl der Modellierungen von Corbett [Cor94], die auch in [MR97, Mel98, Hel99, Hel02] für die Vergleichsstudien bezüglich der Nachweisbarkeit von Verklemmungsfreiheit verwendet wurden. Eine Beschreibung der Testbeispiele befindet sich in Anhang A auf Seite 223.
Die Teilmarkierungen, welche den Algorithmen als Eingabe dienten, wurden automatisch generiert. Dabei wurde besonderer Wert darauf gelegt, keine Markierungen zu erzeugen, denen man bereits durch einfaches Hinsehen angesehen hätte, daß sie unerreichbar gewesen wären.
Die Modellierungen von Corbett [Cor94] lagen in ihrer ursprünglichen Form als endliche, miteinander kommunizierende Automaten vor, die unter Verwendung eines von Römer entwickelten Übersetzers in sichere S/T-Netze überführt wurden. Die S/T-Netze waren derart in Komponenten unterteilt, daß jede Netzkomponente einem Automaten der ursprünglichen Modellierung zugeordnet werden konnte. Jede Netzkomponente enthielt daher genau eine Marke, die innerhalb dieser Komponente „umherwandern“ konnte. Vorausgesetzt, daß die Übersetzung der Automatendarstellung in die Netzdarstellung korrekt durchgeführt wurde, wäre es beispielsweise sinnlos gewesen, eine Teilmarkierung auf Erreichbarkeit zu testen, unter der zwei oder mehrere Stellen derselben Komponente eine Marke aufgewiesen hätten. Für eine solche Markierung hätte man bereits im voraus gewußt, daß sie unerreichbar gewesen wäre. Um die Generierung solcher Teilmarkierungen auszuschließen, wurde daher zunächst die Anzahl n der Komponenten eines Netzes ermittelt. Anschließend wurden beliebig k der n Komponenten und für jede der k Komponenten beliebig eine ihrer Stellen ausgewählt, die dann zusammen die Teilmarkierung der Größe k ergaben. Die Größe einer Teilmarkierung war somit zwar durch die Anzahl der Komponenten eines Netzes beschränkt, aber das stellte keine beträchtliche Einschränkung dar. Die Entscheidung, ob für eine Stelle s nun $M_{par}(s) = 0$ oder $M_{par}(s) = 1$ getestet werden sollte, wurde im Verhältnis 1:4 festgelegt, da es natürlicher erschien, die Anwesenheit von Marken auf Stellen zu überprüfen als deren Abwesenheit.
Alle Testläufe wurden auf einem SUN SPARC20 Rechner mit 96 MByte Hauptspeicher durchgeführt. Der Algorithmus CheckLin verwendete Cplex^{TM} [Cpl97] in der Version 6.5.1 zur Lösung der linearen Ungleichungssysteme, und McSmodels setzte Smodels [NS97, Sim00] als constraint-basierte Programmierumgebung ein. Die Entfaltung eines Netzsystems in dessen Präfix wurde mit einem effizienten Algorithmus von Römer [Röm00] vorgenommen. Die Versuchsreihen wurden für mindestens 20 verschiedene Teilmarkierungen mit jeweils 2, 4 und 6 Stellen durchgeführt.
Die Ergebnisse sind in Tabelle 3.2 auf der nächsten Seite dargestellt. Für jedes System wird in der mit β beschrifteten Zeile die Zeit in Sekunden angegeben, die für die Präfixgenerierung benötigt wurde. In den mit R_i ($i \in \{2, 4, 6\}$) beschrifteten Zeilen sind die

Tabelle 3.2 *Experimentelle Ergebnisse für erreichbare Markierungen der Größe 2, 4, 6 und unerreichbare Markierungen der Größe 4*

		OnTheFly	McSmodels	CheckLin	CheckCo	n
Key(3)	β	-	15.64	15.64	15.64	-
	R_2	3.46	0.74	13.72	1.78	6
	R_4	2.99	1.05	15.41	2.21	9
	R_6	5.71	1.47	15.15	4.08	4
	N_4	16.93	1.79	8.15	2.41	2
Elevator(3)	β	-	3.69	3.69	3.69	-
	R_2	1.04	0.43	4.19	0.63	7
	R_4	1.61	0.56	4.28	0.69	3
	R_6	3.01	0.84	6.10	0.87	2
	N_4	5.57	0.31	2.53	0.65	1
Dpd(7)	β	-	7.73	7.73	7.73	-
	R_2	0.17	0.45	30.73	1.08	-
	R_4	0.25	0.50	29.92	1.43	-
	R_6	0.28	0.54	30.48	2.59	-
	N_4	8.08	2.47	24.54	1.32	2
Dph(6)	β	-	14.75	14.75	14.75	-
	R_2	0.22	0.66	28.28	1.61	-
	R_4	0.37	0.83	33.65	3.58	-
	R_6	0.51	1.03	40.07	5.80	-
	N_4	15.47	2.38	34.63	1.85	2
Furnace(3)	β	-	54.76	54.76	54.76	-
	R_2	0.39	1.33	27.39	4.12	-
	R_4	1.27	1.30	36.09	5.66	-
	R_6	4.52	1.50	30.68	17.06	19
	N_4	57.00	8.91	27.75	5.61	2
Over(5)	β	-	5.52	5.52	5.52	-
	R_2	0.20	0.30	14.28	0.73	-
	R_4	0.28	0.34	15.13	0.90	-
	R_6	0.32	0.36	15.72	1.14	-
	N_4	6.17	0.57	14.04	0.78	1

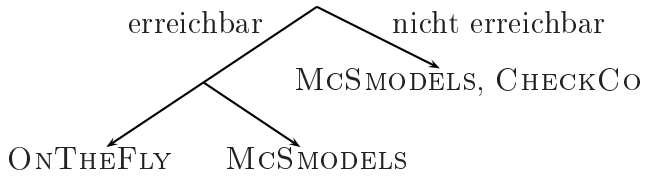
Durchschnittszeiten in Sekunden für die Verifikation von erreichbaren Markierungen mit i Stellen aufgelistet. Entsprechendes gilt für die mit N_4 beschrifteten Zeilen, nur daß hier unerreichbare Markierungen betrachtet wurden. Die Bedeutung der als n gekennzeichneten Spalte wird zu einem späteren Zeitpunkt erläutert.
Zunächst werden die Algorithmen CheckCo, CheckLin und McSmodels miteinan-

der verglichen, da diese Verfahren dasselbe Präfix als Eingabe verwenden. Die Ergebnisse verdeutlichen, daß in den meisten Fällen der Algorithmus McSmodels die schnellsten Verifikationszeiten aufweist, unabhängig von der Größe der Markierungen und der Eigenschaft, ob diese erreichbar oder unerreichbar sind. Allerdings sind die Zeiten für CheckCo bei den meisten Beispielen nur geringfügig schlechter, und für unerreichbare Markierungen scheint CheckCo in den Beispielen Dpd(7), Dph(6) und Furnace(3) die Oberhand gegenüber McSmodels zu behalten. Den schlechtesten Eindruck hinterläßt der Algorithmus CheckLin, dessen Verifikationszeiten in nahezu allen Testreihen deutlich hinter den Zeiten von McSmodels und CheckCo zurückbleiben.
Nun sollen noch die Algorithmen OnTheFly und McSmodels miteinander verglichen werden. Der Algorithmus OnTheFly beendet die Präfixbildung, sobald ein Schnitt gefunden wird, der die zu testende Teilmarkierung repräsentiert. Dieses bedeutet, daß für erreichbare Markierungen höchstens und für unerreichbare Markierungen mindestens das vollständige Präfix berechnet werden muß. Unter dieser Voraussetzung liegt die Vermutung nahe, daß die Verwendung von OnTheFly für erreichbare Markierungen auf jeden Fall geeigneter zu sein scheint als der Einsatz von McSmodels. In der Tat zeigen die experimentellen Ergebnisse, daß OnTheFly erreichbare Markierungen für die Systeme Dpd(7), Dph(6) und Over(5) schneller als McSmodels verifizieren kann. Allerdings verhält es sich für die Systeme Key(3) und Elevator(3) umgekehrt, und in diesen Fällen läßt sich eine kleinste natürliche Zahl n bestimmen, für die

$$n \cdot t_{\text{OnTheFly}} \geq \beta + n \cdot t_{\text{McSmodels}}$$

gilt, wobei t_{OnTheFly} und $t_{\text{McSmodels}}$ die Verifikationszeiten von OnTheFly und McSmodels, sowie β die Zeit für die Präfixgenerierung angeben. Der Wert n gibt dann an, ab welcher Anzahl zu verifizierender Markierungen es günstiger ist, McSmodels anstelle von OnTheFly einzusetzen. Genauer gesagt, werden n oder mehr als n Markierungen getestet, dann ist die Gesamtzeit von McSmodels inklusive der für die Präfixbildung benötigten Zeit kleiner als die Gesamtzeit von OnTheFly. Die Werte für n sind in der mit n bezeichneten Spalte von Tabelle 3.2 auf der vorherigen Seite eingetragen. Die Ergebnisse zeigen, daß für die Verifikation von Markierungen, die im voraus als unerreichbar eingeschätzt werden, in jedem Fall McSmodels oder CheckCo eingesetzt werden sollten, da sie in allen Testergebnissen bereits für die Verifikation von nur zwei Markierungen geringere Zeiten als OnTheFly aufweisen.
Die Graphik in Abbildung 3.21 auf der nächsten Seite verdeutlicht abschließend noch einmal, welche Schlußfolgerungen aus den experimentellen Ergebnissen gezogen werden können. Werden die Markierungen im Vorfeld der Verifikation als unerreichbar eingestuft, empfiehlt es sich, die Verfahren McSmodels oder CheckCo einzusetzen. Liegt die Vermutung nahe, daß die Markierungen erreichbar sind, dann stellt OnTheFly eine sehr effiziente Verifikationsmethode dar. Allerdings kann es ab einer gewissen Anzahl zu verifizierender Markierungen sinnvoll sein, McSmodels anstelle von OnTheFly zu verwenden.

Abbildung 3.21 *Vorschlag für die Verwendung der in diesem Kapitel behandelten Verfahren zur Lösung des Erreichbarkeitsproblems*



Kapitel 4

Erreichbarkeitsanalyse mittels Fallen und Schaltnetzen

Alle in Kapitel 3 vorgestellten Verfahren zur Analyse von Erreichbarkeitsfragen in sicheren S/T-Netzsystemen verwenden halbordnungsbasierte Entfaltungstechniken. Die experimentellen Ergebnisse haben gezeigt, daß diese Verfahren äußerst effizient für den Nachweis von Erreichbarkeitseigenschaften eingesetzt werden können, sobald die endlichen, vollständigen Präfixe der zu untersuchenden Systemmodelle vorliegen. Darin liegt jedoch auch zugleich ein Nachteil dieser Methoden, da auf die Berechnung der Präfixe ein Großteil der gesamten Verifikationszeit entfällt.
Unter diesem Hintergrund wäre es erstrebenswert, einfache Erreichbarkeitsfragen effizient auf den ursprünglichen S/T-Netzsystemmodellen untersuchen zu können, ohne explizit den Weg über die Netzentfaltungen gehen zu müssen. In diesem Zusammenhang wurden in [Mel98] bereits Überlegungen dahingehend angestellt, Erreichbarkeitseigenschaften in sicheren S/T-Netzsystemen unter Verwendung der Markierungsgleichung mittels Methoden der linearen Programmierung zu überprüfen. Während die Markierungsgleichung für azyklische Netzsysteme ein hinreichendes Kriterium für Erreichbarkeit darstellt [Mur89] (siehe auch Abschnitt 3.4.1 auf Seite 68), entsteht jetzt allerdings die Problematik, daß sie für nicht notwendigerweise azyklische Netzsysteme lediglich ein notwendiges, jedoch keinesfalls hinreichendes Kriterium für Erreichbarkeit liefert. Das bedeutet, daß eine Markierung M eines sicheren S/T-Netzsystems Σ von der Anfangsmarkierung M_0 aus unerreichbar ist, falls die Markierungsgleichung von Σ keine Lösung besitzt. Andernfalls muß noch überprüft werden, ob der durch die Lösung der Markierungsgleichung induzierte Schaltvektor in Σ ausgeführt werden kann. Ist dies der Fall, dann ist die Markierung M von der Anfangsmarkierung M_0 aus erreichbar, andernfalls kann keine Aussage bezüglich der Erreichbarkeit von M getroffen werden.
Da die Überprüfung der „Schaltbarkeit“ für eine Lösung der Markierungsgleichung auf

ein PSPACE-vollständiges Erreichbarkeitsproblem zurückgeführt werden kann, möchte man im Vorfeld möglichst viele falsch-positive Lösungen effizient ausschließen. Dazu wurden in [Mel98] zwei Ansätze skizziert, die mittels der strukturellen Konzepte von *Fallen* [DE95, Rei85] und *Schaltnetzen* [Mur89] eine Ausgrenzung falsch-positiver Lösungen der Markierungsgleichung gestatten. Allerdings liegen keine Erkenntnisse darüber vor, daß diese Ideen jemals zuvor vollständig implementiert bzw. Praxistests unterzogen wurden, weshalb sie in diesem Kapitel noch einmal aufgegriffen und erweitert werden. Darüberhinaus sind beide Verfahren im Rahmen der vorliegenden Arbeit vollständig implementiert und anhand von Testreihen auf ihre Praxistauglichkeit untersucht worden.

4.1 Das Konzept der Fallen

Im folgenden wird zunächst das strukturelle Konzept der Falle formal definiert, um anschließend auf eine fundamentale Eigenschaft derselben eingehen zu können, der eine grundlegende Bedeutung für die weiteren Ausführungen in diesem Abschnitt zukommt.

Definition 4.1.1 (Falle [DE95, Rei85])
Gegeben sei ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Teilmenge $\Theta \subseteq S$ von Stellen wird als *Falle* von Σ bezeichnet, falls

$$\Theta^\bullet \subseteq {}^\bullet\Theta$$

Als direkte Konsequenz aus Definition 4.1.1 ergibt sich Proposition 4.1.1, die eine fundamentale Eigenschaft von Fallen beschreibt.

Proposition 4.1.1 (Fundamentale Eigenschaft von Fallen [DE95, Rei85])
Gegeben seien ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine Falle $\Theta \subseteq S$ von Σ. Dann gilt für alle $M_1 \in \mathcal{R}(M_0)$:

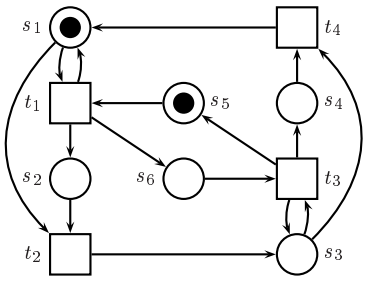
$$\Big(\sum_{s\in\Theta} M_1(s)\Big) > 0 \quad \Rightarrow \quad \Big(\sum_{s\in\Theta} M_2(s)\Big) > 0 \quad \text{für jedes } M_2 \in \mathcal{R}(M_1)$$

Proposition 4.1.1 besagt also, daß eine Falle Θ eine Stellenmenge von Σ beschreibt, die, sofern sie unter einer von der Anfangsmarkierung M_0 aus erreichbaren Markierung M_1 markiert ist, auch unter jeder von M_1 aus erreichbaren Markierung M_2 markiert sein wird. Mittels dieser Eigenschaft können nun falsch-positive Lösungen der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot X$$

entdeckt werden, indem überprüft wird, ob das S/T-Netzsystem Σ eine unter M_0 markierte Falle aufweist, die unter M unmarkiert ist. In diesem Fall wäre die Markierung M wegen Proposition 4.1.1 nicht erreichbar, und die gefundene Lösung der Markierungsgleichung könnte somit ausgeschlossen werden.
Eine solche Vorgehensweise ist natürlich nur dann sinnvoll, falls es möglich ist, Fallen auf effiziente Art und Weise zu detektieren und somit den Lösungsraum effizient einschränken zu können. Dieser Aspekt wird in Abschnitt 4.1.1 nun ausführlich behandelt.

Abbildung 4.1 *Sicheres S/T-Netzsystem mit Fallen*



4.1.1 Erkennen von Fallen mittels linearer Programmierung

In der Literatur sind verschiedene Verfahren bekannt, um Fallen in S/T-Netzsystemen mittels Techniken der linearen Programmierung effizient zu erkennen, siehe beispielsweise [AT85, ECS93]. Auf die Herleitung und Korrektheit dieser Methoden soll an dieser Stelle nicht näher eingegangen werden. Der Hauptunterschied zwischen den beiden Ansätzen besteht jedoch darin, daß für ein S/T-Netz (S, T, F) in [AT85] $(S \times F)$-Matrizen zur Fallendetektion aufgestellt werden, wohingegen in [ECS93] lediglich $(S \times T)$-Matrizen verwendet werden. Deshalb wurde im Hinblick auf eine auch den Speicherbedarf optimierende Implementierung die Entscheidung getroffen, den in [ECS93] vorgestellten Ansatz zur Fallenerkennung weiterzuverfolgen.

Satz 4.1.1 (Detektion von Fallen [ECS93])
Gegeben sei ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Menge

$$\Theta = \{s \in S \mid Y(s) > 0\}$$

bildet genau dann eine Falle von Σ, wenn es einen rationalen Stellenvektor $Y \in \mathbb{Q}^{|S|}$ mit $Y \geq \mathbf{0}$ gibt, sodaß

$$Y^t \cdot \mathbf{N}_\Theta \geq \mathbf{0},$$

wobei $\mathbf{N}_\Theta$ eine $(S \times T)$-Matrix mit

$$\mathbf{N}_\Theta(s,t) = \begin{cases} -1 & \textit{falls } s \in {}^\bullet t \setminus t^\bullet \\ |{}^\bullet t| & \textit{falls } s \in t^\bullet \setminus {}^\bullet t \\ |{}^\bullet t| - 1 & \textit{falls } s \in {}^\bullet t \cap t^\bullet \\ 0 & \textit{sonst} \end{cases}$$

beschreibt.

Satz 4.1.1 soll nun anhand des sicheren S/T-Netzsystems Σ aus Abbildung 4.1 ([Mel98]) näher erläutert werden. Um eine Falle in Σ zu finden, muß nach einem rationalen Stel-

lenvektor $Y \geq \mathbf{0}$ des folgenden linearen Ungleichungssystems gesucht werden[1]:

$$\begin{aligned} Y(s_1) + 2 \cdot Y(s_2) - Y(s_5) + 2 \cdot Y(s_6) &\geq 0 \\ -Y(s_1) - Y(s_2) + 2 \cdot Y(s_3) &\geq 0 \\ Y(s_3) + 2 \cdot Y(s_4) + 2 \cdot Y(s_5) - Y(s_6) &\geq 0 \\ 2 \cdot Y(s_1) - Y(s_3) - Y(s_4) &\geq 0 \end{aligned}$$

Der Vektor $Y = (0, 0, 0, 0, 1, 1)^t$ stellt beispielsweise eine Lösung des linearen Ungleichungssystems dar, und die von Y induzierte Stellenmenge $\Theta = \{s_5, s_6\}$ beschreibt nach Definition 4.1.1 auf Seite 84 wegen

$$\Theta^\bullet = \{t_1, t_3\} = {}^\bullet\Theta$$

eine Falle des S/T-Netzsystems Σ.
Die in Satz 4.1.1 auf der vorherigen Seite behandelte Methode liefert ein effizientes Verfahren zur Erkennung von Fallen, da lineare Ungleichungssysteme über der Menge $\mathbb{Q}$ der rationalen Zahlen in polynomieller Zeit mittels eines von Renegar beschriebenen Algorithmus' [Ren88] gelöst werden können.

4.1.2 Ein iterativer Algorithmus

Nachdem in Abschnitt 4.1.1 auf der vorherigen Seite eine effiziente Methode zur automatischen Erkennung von Fallen in einem S/T-Netzsystem vorgestellt wurde, kann nun unter Verwendung des Ergebnisses von Satz 4.1.2 ein Verfahren hergeleitet werden, das unter iterativem Ausschluß falsch-positiver Lösungen der Markierungsgleichung versucht, die (Un-)Erreichbarkeit einer Teilmarkierung in einem sicheren S/T-Netzsystem nachzuweisen.

Satz 4.1.2 (Falsch-positive Lösungen der Markierungsgleichung)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine Lösung $X \in \mathbb{N}^{|T|}$ der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot X$$

von Σ. Die Markierung M ist von M_0 aus unerreichbar, d.h., $M \notin \mathcal{R}(M_0)$, falls das folgende lineare Ungleichungssystem eine rationale Lösung $Y \in \mathbb{Q}^{|S|}$ mit $Y \geq \mathbf{0}$ besitzt:

$$Y^t \cdot \mathbf{N}_\Theta \geq \mathbf{0} \tag{4.1.1}$$
$$Y^t \cdot M_0 > 0 \tag{4.1.2}$$
$$Y^t \cdot M = 0 \tag{4.1.3}$$

[1] Da $Y = \mathbf{0}$ immer eine Lösung des Ungleichungssystems darstellt, sollte zusätzlich $Y \neq \mathbf{0}$ angenommen werden.

BEWEIS: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Lösung $X \in \mathbb{N}^{|T|}$ der Markierungsgleichung von Σ und eine Lösung $Y \in \mathbb{Q}^{|S|}$ mit $Y \geq \mathbf{0}$ des obigen Ungleichungssystems.
Angenommen, die Markierung M ist von der Anfangsmarkierung M_0 aus erreichbar, d.h., $M \in \mathcal{R}(M_0)$. Aus Zeile 4.1.1 auf der vorherigen Seite und Satz 4.1.1 auf Seite 85 folgt, daß die Stellenmenge

$$\Theta = \{s \in S \mid Y(s) > 0\}$$

eine Falle von Σ darstellt. Aus Zeile 4.1.2 auf der vorherigen Seite folgt unmittelbar

$$\exists s \in \Theta \colon M_0(s) = 1,$$

d.h., mindestens eine Stelle aus Θ ist anfänglich markiert. Unter der Annahme $M \in \mathcal{R}(M_0)$ folgt damit aus Proposition 4.1.1 auf Seite 84

$$\exists s \in \Theta \colon M(s) = 1,$$

d.h., mindestens eine Stelle aus Θ ist unter der Markierung M markiert. Dieses steht jedoch im unmittelbaren Widerspruch zu Zeile 4.1.3 auf der vorherigen Seite, aus der

$$\forall s \in \Theta \colon M(s) = 0$$

folgt, d.h., keine Stelle aus Θ ist unter der Markierung M markiert. Somit gilt $M \notin \mathcal{R}(M_0)$. ■

Iterativer Algorithmus TRAPCHECK
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$, die auf Erreichbarkeit überprüft werden soll.

(1) Berechne eine Lösung des linearen Ungleichungssystems

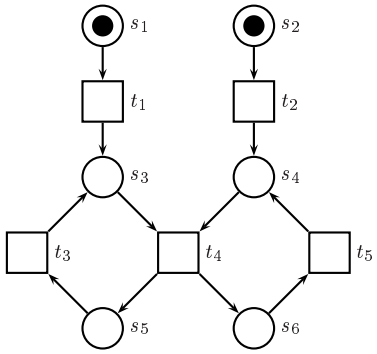
$$\mathcal{L}_0 \colon \begin{cases} \text{Variablen: } M \text{ binär}, X \geq \mathbf{0} \text{ ganzzahlig} \\ M = M_0 + \mathbf{N} \cdot X \\ \mathbf{A} \cdot M \geq b \end{cases}$$

wobei $\mathbf{A} \cdot M \geq b$ die in Matrixschreibweise kodierte Teilmarkierung M_{par} darstellt (siehe dazu Abschnitt 3.4.1 auf Seite 68).

(2) Bezeichne $\mathcal{L}_i$ das Ungleichungssystem, das im i-ten Iterationsschritt erzeugt wurde. Falls das lineare Ungleichungssystem $\mathcal{L}_i$ keine Lösung besitzt, beende den Algorithmus mit der Ausgabe, daß die Teilmarkierung M_{par} nicht erreichbar ist, andernfalls gehe zu (3).

(3) Überprüfe mit dem in Satz 4.1.2 auf der vorherigen Seite beschriebenen Verfahren, ob $M \notin \mathcal{R}(M_0)$. Falls ja, gehe zu (4), andernfalls gehe zu (5).

Abbildung 4.2 *Sicheres S/T-Netzsystem, für das die Erreichbarkeit der Teilmarkierung* $(\{s_6\},\{s_5\})$ *nachgewiesen werden soll.*



(4) Bezeichne Θ die in (3) erkannte Falle, die unter M_0 markiert, aber unter M unmarkiert ist. Berechne eine Lösung des linearen Ungleichungssystems

$$\mathcal{L}_{i+1}\colon \begin{cases} \mathcal{L}_i \\ \left(\sum_{s\in\Theta} M(s)\right) \geq 1 \end{cases}$$

und gehe zu (2).

(5) Teste, ob der Transitionsvektor $X \in \mathbb{N}^{|T|}$ in Σ schalten kann (siehe dazu Abschnitt 4.3 auf Seite 102). Falls ja, so ist die Teilmarkierung M_{par} von M_0 aus erreichbar, andernfalls kann keine Aussage bezüglich der Erreichbarkeit von M_{par} in Σ getroffen werden.

In Schritt (4) des Algorithmus' wird dem zu lösenden Ungleichungssystem jeweils eine unter M_0 markierte, jedoch unter M unmarkierte Falle als neue Schnittebene hinzugefügt. Das S/T-Netzsystem Σ kann zwar exponentiell in der Größe von Σ viele Fallen aufweisen, jedoch ergibt sich damit immer noch eine endliche Anzahl von möglichen Schnittebenen, die dem Ungleichungssystem in Schritt (4) hinzugefügt werden. Somit ist die Terminierung von TrapCheck garantiert.
Anhand des sicheren S/T-Netzsystems Σ aus Abbildung 4.2 ([Mel98]) soll nun gezeigt werden, daß mit dem Algorithmus TrapCheck die Erreichbarkeit der Teilmarkierung

$$M_{par} = (\{s_6\},\{s_5\})$$

nachgewiesen werden kann. Zunächst wird in Schritt (1) von TrapCheck eine Lösung

des linearen Ungleichungssystems

$$\mathcal{L}_0\colon \begin{cases} \text{Variablen: } M \text{ binär}, X \geq \mathbf{0} \text{ ganzzahlig} \\ M(s_1) = 1 - X(t_1) \\ M(s_2) = 1 - X(t_2) \\ M(s_3) = X(t_1) + X(t_3) - X(t_4) \\ M(s_4) = X(t_2) - X(t_4) + X(t_5) \\ M(s_5) = -X(t_3) + X(t_4) \\ M(s_6) = X(t_4) - X(t_5) \\ M(s_5) \geq 1 \\ M(s_6) \leq 0 \end{cases}$$

gesucht. Eine Lösung für $\mathcal{L}_0$ ist durch

$$M = (0,1,0,0,1,0)^t \quad \text{und} \quad X = (1,0,0,1,1)^t$$

gegeben. Da $\mathcal{L}_0$ eine Lösung besitzt, wird in Schritt (3) von TrapCheck nach Satz 4.1.2 auf Seite 86 überprüft, ob das lineare Ungleichungssystem

$$\mathcal{F}_0\colon \begin{cases} \text{Variable: } Y \geq \mathbf{0} \text{ rational} \\ 0 \leq -Y(s_1) + Y(s_3) \\ 0 \leq -Y(s_2) + Y(s_4) \\ 0 \leq Y(s_3) - Y(s_5) \\ 0 \leq -Y(s_3) - Y(s_4) + 2 \cdot Y(s_5) + 2 \cdot Y(s_6) \\ 0 \leq Y(s_4) - Y(s_6) \\ 0 < Y(s_1) + Y(s_2) \\ 0 = Y(s_2) + Y(s_5) \end{cases}$$

eine Lösung besitzt und M somit in Σ von M_0 aus unerreichbar ist. $\mathcal{F}_0$ besitzt die Lösung

$$Y = (1,0,1,1,0,1)^t,$$

d.h., nach Satz 4.1.2 auf Seite 86 bildet die Stellenmenge

$$\Theta = \{s_1, s_3, s_4, s_6\}$$

eine Falle von Σ, die wegen $M_0(s_1) = 1$ unter M_0 markiert, aber unter $M = \{s_2, s_5\}$ unmarkiert ist.
(Tatsächlich handelt es sich bei der Stellenmenge Θ wegen

$$\Theta^\bullet = \{t_1, t_4, t_5\} \subseteq \{t_1, t_2, t_4, t_5\} = {}^\bullet\Theta$$

auch nach Definition 4.1.1 auf Seite 84 um eine Falle von Σ).
Es hat sich also herausgestellt, daß $M \notin \mathcal{R}(M_0)$, da unter jeder von M_0 aus erreichbaren Markierung mindestens eine der Stellen aus Θ eine Marke aufweisen muß. Diese falsch-positive Lösung der Markierungsgleichung wird nun ausgeschlossen, indem dem Ungleichungssystem $\mathcal{L}_0$ in Schritt (4) von TrapCheck die Falle Θ als Schnittebene hinzugefügt wird. Somit ergibt sich das lineare Ungleichungssystem

$$\mathcal{L}_1 : \begin{cases} \mathcal{L}_0 \\ M(s_1) + M(s_3) + M(s_4) + M(s_6) \geq 1, \end{cases}$$

das es nun zu lösen gilt. Eine Lösung für $\mathcal{L}_1$ ist durch

$$M = (0,0,0,1,1,0)^t \quad \text{und} \quad X = (1,1,0,1,1)^t$$

gegeben. Nun wird in Schritt (3) von TrapCheck nach einer Lösung des linearen Ungleichungssystems

$$\mathcal{F}_1 : \begin{cases} \text{Variable: } Y \geq \mathbf{0} \text{ rational} & \\ 0 \leq -Y(s_1) + Y(s_3) & \text{(a)} \\ 0 \leq -Y(s_2) + Y(s_4) & \text{(b)} \\ 0 \leq Y(s_3) - Y(s_5) & \text{(c)} \\ 0 \leq -Y(s_3) - Y(s_4) + 2 \cdot Y(s_5) + 2 \cdot Y(s_6) & \text{(d)} \\ 0 \leq Y(s_4) - Y(s_6) & \text{(e)} \\ 0 < Y(s_1) + Y(s_2) & \text{(f)} \\ 0 = Y(s_4) + Y(s_5) & \text{(g)} \end{cases} \tag{4.1.4}$$

gesucht. Das Ungleichungssystem $\mathcal{F}_1$ ist jedoch unlösbar, d.h., es gibt in Σ keine weitere unter M_0 markierte, aber unter M unmarkierte Falle.
(Aus 4.1.4 (g) folgt unmittelbar $Y(s_4) = Y(s_5) = 0$. Daraus folgt jedoch mit 4.1.4 (b) und (e) unmittelbar $Y(s_2) = Y(s_6) = 0$. Mit 4.1.4 (f) gilt dann $Y(s_1) > 0$. Aus 4.1.4 (a) folgt dann $Y(s_3) > 0$, was jedoch einen Widerspruch zu Ungleichung 4.1.4 (d) ergibt, für die $Y(s_3) = 0$ gelten müßte).
Die Markierung $M = \{s_4, s_5\}$ kann also mittels Satz 4.1.2 auf Seite 86 nicht als unerreichbar ausgeschlossen werden, weshalb in Schritt (5) von TrapCheck nun noch getestet werden muß, ob der Schaltvektor

$$X = (1,1,0,1,1)^t$$

in Σ ausgeführt werden kann und M somit tatsächlich von M_0 aus erreichbar ist (siehe dazu Abschnitt 4.3 auf Seite 102). Dieses entspricht der Suche nach einer Schaltfolge von Σ, welche die Anfangsmarkierung M_0 in die Markierung M überführt. Durch die Vorgabe des Lösungsvektors X wird die Suche allerdings eingeschränkt, da lediglich Schaltfolgen bestehend aus denjenigen Transitionen betrachtet werden müssen, für die

der Lösungsvektor X einen Wert größer als Null ergeben hat. Tatsächlich gibt es in dem S/T-Netzsystem Σ aus Abbildung 4.2 auf Seite 88 den Ablauf

$$M_0 = \{s_1, s_2\} \xrightarrow{t_1} \{s_2, s_3\} \xrightarrow{t_2} \{s_3, s_4\} \xrightarrow{t_4} \{s_5, s_6\} \xrightarrow{t_5} \{s_4, s_5\} = M$$

Aus der Erreichbarkeit der Markierung M folgt dann auch unmittelbar die Erreichbarkeit der Teilmarkierung $M_{par} = (\{s_6\}, \{s_5\})$.
Allerdings stellt TrapCheck kein exaktes Verfahren für den Nachweis der Erreichbarkeit von Teilmarkierungen in sicheren S/T-Netzsystemen dar, wie anhand des sicheren S/T-Netzsystems Σ aus Abbildung 4.1 auf Seite 85 und der Teilmarkierung

$$M_{par} = (\emptyset, \{s_2, s_4\})$$

gezeigt werden kann. Das lineare Ungleichungssystem

$$\mathcal{L}_0 : \begin{cases} \text{Variablen: } M \text{ binär}, X \geq \mathbf{0} \text{ ganzzahlig} \\ M(s_1) = 1 - X(t_2) + X(t_4) \\ M(s_2) = X(t_1) - X(t_2) \\ M(s_3) = X(t_2) - X(t_4) \\ M(s_4) = X(t_3) - X(t_4) \\ M(s_5) = 1 - X(t_1) + X(t_3) \\ M(s_6) = X(t_1) - X(t_3) \\ M(s_2) \geq 1 \\ M(s_4) \geq 1 \end{cases}$$

besitzt die Lösung

$$M = (1, 1, 0, 1, 1, 0)^t \quad \text{und} \quad X = (1, 0, 1, 0)^t.$$

Da die Anfangsmarkierung M_0 des S/T-Netzsystems Σ aus Abbildung 4.1 auf Seite 85 eine Teilmenge der Markierung M bildet, d.h.,

$$M_0 = \{s_1, s_5\} \subset \{s_1, s_2, s_4, s_5\} = M,$$

kann es in Σ keine Falle geben, die unter M_0 markiert, aber unter M unmarkiert ist. Folglich ist das lineare Ungleichungssystem

$$\mathcal{F}_0 : \begin{cases} \text{Variable: } Y \geq \mathbf{0} \text{ rational} & \\ 0 \leq Y(s_1) + 2 \cdot Y(s_2) - Y(s_5) + 2 \cdot Y(s_6) & \text{(a)} \\ 0 \leq -Y(s_1) - Y(s_2) + 2 \cdot Y(s_3) & \text{(b)} \\ 0 \leq Y(s_3) + 2 \cdot Y(s_4) + 2 \cdot Y(s_5) - Y(s_6) & \text{(c)} \\ 0 \leq 2 \cdot Y(s_1) - Y(s_3) - Y(s_4) & \text{(d)} \\ 0 < Y(s_1) + Y(s_5) & \text{(e)} \\ 0 = Y(s_1) + Y(s_2) + Y(s_4) + Y(s_5) & \text{(f)} \end{cases} \tag{4.1.5}$$

unlösbar. (Die Unlösbarkeit von $\mathcal{F}_0$ ergibt sich unmittelbar aus den Zeilen 4.1.5 (e) und (f)). Mittels Satz 4.1.2 auf Seite 86 kann die Markierung M nicht als von M_0 aus unerreichbar ausgeschlossen werden. Ein Blick auf das S/T-Netzsystem Σ (Abbildung 4.1 auf Seite 85) zeigt jedoch, daß alle Abläufe von Σ die Form

$$M_0 \xrightarrow{(t_1 t_2 t_3 t_4)^*}$$

aufweisen. Daraus wird deutlich, daß die durch den Transitionsvektor

$$X = (1, 0, 1, 0)^t$$

induzierten Schaltfolgen $t_1 t_3$ bzw. $t_3 t_1$ nicht in Σ ausgeführt werden können. Leider kann daraus nicht abgeleitet werden, daß die Markierung M von M_0 aus unerreichbar ist, da die Markierungsgleichung bzw. das Ungleichungssystem $\mathcal{L}_0$ durchaus noch weitere Lösungen aufweisen könnten. Der Algorithmus TrapCheck kann also in diesem Fall nicht entscheiden, ob die Teilmarkierung

$$M_{par} = (\emptyset, \{s_2, s_4\})$$

in dem S/T-Netzsystem Σ aus Abbildung 4.1 auf Seite 85 von der Anfangsmarkierung M_0 aus erreichbar ist.
Deshalb wird in Abschnitt 4.2 das Verfahren TrapSchaltCheck vorgestellt, welches qualitativ besser als TrapCheck in der Hinsicht ist, daß mittels TrapSchaltCheck falsch-positive Lösungen der Markierungsgleichung eines sicheren S/T-Netzsystems ausgeschlossen werden können, über deren Gültigkeit mittels des Verfahrens TrapCheck keine Aussage getroffen werden kann, ohne den Schaltvektortest durchführen zu müssen.

4.2 Das Konzept der Schaltnetze

Als weiteres notwendiges Kriterium für den Nachweis der Erreichbarkeit von Markierungen in sicheren S/T-Netzsystemen spielen in der Literatur [Mur89] beispielsweise sogenannte *Schaltnetze* eine Rolle, durch deren strukturelle Analyse falsch-positive Lösungen der Markierungsgleichung entdeckt und eliminiert werden können.
Ein Schaltnetz $\Sigma_{\vec{\sigma}}$ eines sicheren S/T-Netzsystems Σ wird durch den Parikhvektor $\vec{\sigma}$ der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot \vec{\sigma}$$

von Σ derart induziert, daß es lediglich die Transitionen enthält, die laut $\vec{\sigma}$ schalten müssen, um die Markierung M von M_0 aus zu erreichen. Der Vorteil einer solchen Vorgehensweise liegt darin, daß irrelevante Netzteile von Σ ausgeblendet werden und mit $\Sigma_{\vec{\sigma}}$ in vielen Fällen ein recht kleines Netz vorliegt, das dann auf sehr effiziente Weise strukturell analysiert werden kann.

Definition 4.2.1 (Schaltnetze)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine Lösung $X \in \mathbb{N}^{|T|}$ der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot X$$

von Σ. Das durch den Schaltvektor X generierte *Schaltnetzsystem* Σ_X von Σ ist durch das Tupel (N_X, M_{0_X}) mit $N_X = (S_X, T_X, F_X)$ gegeben, wobei

$$\begin{aligned} S_X &= \{s \in S \mid \exists t \in T : X(t) > 0 \wedge s \in {}^\bullet t \cup t^\bullet\} \\ T_X &= \{t \in T \mid X(t) > 0\} \\ F_X &= F \cap ((S_X \times T_X) \cup (T_X \times S_X)) \\ M_{0_X} &= M_0|_{S_X} \end{aligned}$$

In diesem Fall wird N_X auch als *Schaltnetz* von Σ bezeichnet.

Mit dem Konzept der Schaltnetzsysteme ergibt sich nun die Möglichkeit, ein weiteres notwendiges Kriterium für erreichbare Markierungen sicherer S/T-Netzsysteme festzulegen.

Satz 4.2.1 (Notwendiges Kriterium für Erreichbarkeit [Mur89])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Falls eine Markierung M von M_0 aus erreichbar ist, d.h., $M \in \mathcal{R}(M_0)$, dann gibt es eine Lösung $X \in \mathbb{N}^{|T|}$ der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot X$$

von Σ, sodaß in dem durch den Schaltvektor X generierten Schaltnetzsystem $\Sigma_X = (N_X, M_{0_X})$ mit $N_X = (S_X, T_X, F_X)$ keine nichtleere, unter $M|_{S_X}$ unmarkierte Falle existiert.

In Abschnitt 4.2.1 werden nun unter Berücksichtigung des Ergebnisses von Satz 4.2.1 Schnittebenen hergeleitet, mit denen falsch-positive Lösungen der Markierungsgleichung eines S/T-Netzsystems ausgeschlossen werden können.

4.2.1 Ein iterativer Algorithmus

Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Satz 4.2.1 besagt, daß es für jede erreichbare Markierung $M \in \mathcal{R}(M_0)$ eine Lösung $X \in \mathbb{N}^{|T|}$ der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot X$$

von Σ gibt, sodaß in dem durch den Schaltvektor X generierten Schaltnetzsystem $\Sigma_X = (N_X, M_{0_X})$ mit $N_X = (S_X, T_X, F_X)$ keine nichtleere, unter $M|_{S_X}$ unmarkierte Falle existiert.

Falls es in Σ_X nun doch eine nichtleere, unter $M|_{S_X}$ unmarkierte Falle Θ gibt, muß diese falsch-positive Lösung der Markierungsgleichung ausgeschlossen werden, und da bei dem Nachweis der Erreichbarkeit von Teilmarkierungen sowohl M als auch X als Variablenvektoren der Markierungsgleichung angesehen werden, gibt es mehrere Möglichkeiten für die Festlegung von Schnittebenen.[2]

(i) $\left(\sum_{s\in\Theta} M_{0_X}(s)\right) > 0$ und $\left(\sum_{s\in\Theta} M|_{S_X}(s)\right) = 0$
Da Θ eine Falle von Σ_X darstellt, gilt

$$\Theta^{\bullet}_{\Sigma_X} \subseteq {}^{\bullet}\Theta_{\Sigma_X}$$

Da jedoch Σ_X ein Teilnetz von Σ darstellt, könnte

$$\Theta^{\bullet}_{\Sigma} \setminus {}^{\bullet}\Theta_{\Sigma} \neq \emptyset$$

gelten, weshalb Θ dann keine Falle von Σ ergeben würde. Somit müssen also erneut zwei Fälle betrachtet werden:

(a) $\Theta^{\bullet}_{\Sigma} \subseteq {}^{\bullet}\Theta_{\Sigma}$, d.h., Θ stellt eine Falle von Σ dar.
Dann folgt $M \notin \mathcal{R}(M_0)$, da ansonsten die Falle Θ nach Proposition 4.1.1 auf Seite 84 auch unter der Markierung M markiert sein müßte. Deshalb wird die Markierungsgleichung um die Schnittebene

$$\left(\sum_{s\in\Theta} M(s)\right) > 0 \tag{4.2.1}$$

erweitert.

(b) $\Theta^{\bullet}_{\Sigma} \setminus {}^{\bullet}\Theta_{\Sigma} \neq \emptyset$, d.h., Θ stellt keine Falle von Σ dar.
Dann müssen die beiden Fälle unterschieden werden, daß entweder mindestens eine oder keine Transition aus $\Theta^{\bullet}_{\Sigma} \setminus {}^{\bullet}\Theta_{\Sigma}$ schaltet.

(1) Keine Transition aus $\Theta^{\bullet}_{\Sigma} \setminus {}^{\bullet}\Theta_{\Sigma}$ schaltet.
Da die Stellenmenge Θ unter M_0 markiert ist, muß sie auch unter M markiert bleiben. Deshalb wird die Markierungsgleichung um die Schnittebenen

$$\left(\sum_{s\in\Theta} M(s)\right) > 0 \tag{4.2.2}$$

$$\left(\sum_{t\in\Theta^{\bullet}_{\Sigma}\setminus{}^{\bullet}\Theta_{\Sigma}} X(t)\right) = 0 \tag{4.2.3}$$

erweitert.

[2] Zur Unterscheidung, ob ${}^{\bullet}\Theta$ (bzw. $\Theta^{\bullet}$) auf Σ oder Σ_X bezogen werden, werden im folgenden die Notationen ${}^{\bullet}\Theta_{\Sigma}$ (bzw. $\Theta^{\bullet}_{\Sigma}$) und ${}^{\bullet}\Theta_{\Sigma_X}$ (bzw. $\Theta^{\bullet}_{\Sigma_X}$) verwendet.

(2) Mindestens eine Transition aus $\Theta_\Sigma^\bullet \setminus {}^\bullet\Theta_\Sigma$ schaltet.
In diesem Fall wird die Markierungsgleichung um die Schnittebene

$$\Big(\sum_{t\in\Theta_\Sigma^\bullet\setminus{}^\bullet\Theta_\Sigma} X(t)\Big) > 0 \tag{4.2.4}$$

erweitert.

Die beiden Schnitteben 4.2.2 und 4.2.3 auf der vorherigen Seite induzieren Lösungen der Markierungsgleichung, bei denen

* *mindestens eine* Stelle von Θ markiert ist und *keine* Transition aus $\Theta_\Sigma^\bullet \setminus {}^\bullet\Theta_\Sigma$ schaltet.

Die Schnittebe 4.2.4 induziert Lösungen der Markierungsgleichung, bei denen

* *keine* Stelle aus Θ markiert ist und *mindestens eine* Transition aus $\Theta_\Sigma^\bullet \setminus {}^\bullet\Theta_\Sigma$ schaltet, oder
* *mindestens eine* Stelle aus Θ markiert ist und *mindestens eine* Transition aus $\Theta_\Sigma^\bullet \setminus {}^\bullet\Theta_\Sigma$ schaltet.

Genau diese Lösungen werden aber auch durch die Schnittebene

$$\Big(\sum_{s\in\Theta} M(s) + \sum_{t\in\Theta_\Sigma^\bullet\setminus{}^\bullet\Theta_\Sigma} X(t)\Big) > 0 \tag{4.2.5}$$

induziert, weshalb die Schnitteben 4.2.2 und 4.2.3 auf der vorherigen Seite sowie 4.2.4 zu dieser einen Schnittebene zusammengefaßt werden können, wodurch sich zusätzlich die Anzahl zu lösender Ungleichungssysteme verringert.

Darüberhinaus bleibt festzuhalten, daß die Schnittebenen 4.2.1 auf der vorherigen Seite (Fall (a)) und 4.2.5 (Fall (b)) im Falle $\Theta_\Sigma^\bullet \subseteq {}^\bullet\Theta_\Sigma$ identisch sind, sodaß auch hier keine Unterscheidung getroffen zu werden braucht und die Markierungsgleichung im Fall (i) einfach um die Schnittebene 4.2.5 ergänzt wird.

(ii) $\Big(\sum_{s\in\Theta} M_{0_X}(s)\Big) = 0$ und $\Big(\sum_{s\in\Theta} M|_{S_X}(s)\Big) = 0$
Die Tatsache, daß die Stellenmenge Θ sowohl unter M_{0_X} als auch unter $M|_{S_X}$ unmarkiert ist, schließt jedoch nicht aus, daß Θ in Σ „zwischenzeitlich“ hätte markiert sein können. Aus diesem Grund müssen zwei Fälle betrachtet werden:

(a) $\exists M_1 \in \mathcal{R}(M_0)\colon \Big(M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M \wedge \Big(\sum_{s\in\Theta} M_1(s)\Big) > 0\Big)$
Dann muß es jedoch eine Zerlegung von σ_1 in $\sigma_1' t \sigma_1''$ mit $t \in {}^\bullet\Theta_\Sigma \setminus \Theta_\Sigma^\bullet$ geben. Die Markierungsgleichung wird also zunächst um die Schnittebene

$$\Big(\sum_{t\in{}^\bullet\Theta_\Sigma\setminus\Theta_\Sigma^\bullet} X(t)\Big) > 0$$

erweitert. Desweiteren liegt jetzt wieder Fall (i) vor, d.h., die Stellenmenge Θ beschreibt eine Falle von Σ_X, die unter $M_1|_{S_X}$ markiert, aber unter $M|_{S_X}$ unmarkiert ist. Also wird die Markierungsgleichung noch um die Schnittebene 4.2.5 auf der vorherigen Seite erweitert.

(b) Zwischen den Markierungen M_0 und M kommt keine wie in Fall (a) beschriebene Markierung M_1 vor.
Daraus folgt dann aber unmittelbar, daß in der Schaltfolge $\sigma_1\sigma_2$ keine Transition aus ${}^\bullet\Theta_\Sigma \cup \Theta_\Sigma^\bullet$ vorkommen kann, da die Stellen aus Θ anfänglich unmarkiert sind und während der Ausführung der Schaltfolge $\sigma_1\sigma_2$ auch nicht markiert werden. In diesem Fall wird die Markierungsgleichung um die Schnittebene

$$\Big(\sum_{s\in\Theta} M(s) + \sum_{t\in{}^\bullet\Theta_\Sigma\cup\Theta_\Sigma^\bullet} X(t)\Big) = 0$$

erweitert.

Unter Verwendung dieser Erkenntnisse kann nun ein iteratives Halbentscheidungsverfahren für die Erreichbarkeit von Teilmarkierungen in sicheren S/T-Netzsystemen mittels der Schaltnetztheorie angegeben werden.

Iterativer Algorithmus TrapSchaltCheck
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine Teilmarkierung $M_{par} = (S^0, S^1)$ mit $S^0, S^1 \subseteq S$, die auf Erreichbarkeit überprüft werden soll. Desweiteren bezeichne $\mathcal{Q}$ eine anfänglich leere Liste, welche die zu lösenden Ungleichungssysteme verwaltet.

(1) Berechne eine Lösung des linearen Ungleichungssystems

$$\mathcal{L}_0 : \begin{cases} \text{Variablen: } M \text{ binär}, X \geq \mathbf{0} \text{ ganzzahlig} \\ M = M_0 + \mathbf{N}\cdot X \\ \mathbf{A}\cdot M \geq b \end{cases}$$

wobei $\mathbf{A}\cdot M \geq b$ die in Matrixschreibweise kodierte Teilmarkierung M_{par} darstellt (siehe dazu Abschnitt 3.4.1 auf Seite 68).

(2) Bezeichne $\mathcal{L}_i$ das im i-ten Iterationsschritt erzeugte lineare Ungleichungssystem. Falls $\mathcal{L}_i$ keine Lösung besitzt, gehe zu (6), andernfalls gehe zu (3).

(3) Generiere das durch den Schaltvektor X induzierte Schaltnetzsystem Σ_X.

(4) Überprüfe, ob es in Σ_X eine nichtleere, unter der Markierung $M|_{S_X}$ unmarkierte Falle Θ gibt. (Dazu kann die Methode aus Satz 4.1.2 auf Seite 86 für $Y \neq \mathbf{0}$ und ohne die Ungleichung 4.1.2 auf Seite 86 verwendet werden). Falls es eine solche Falle Θ gibt, gehe zu (5), andernfalls gehe zu (7).

(5) Bezeichne Θ die in Schritt (4) erkannte Falle, die unter $M|_{S_X}$ unmarkiert ist.

– Falls Θ unter M_{0_X} markiert ist, füge der Liste $\mathcal{Q}$ das lineare Ungleichungssystem

$$\mathcal{L}_{i+1}^{0}\colon \begin{cases} \mathcal{L}_i \\ \left(\sum_{s\in\Theta} M(s) + \sum_{t\in\Theta_\Sigma^\bullet\setminus{}^\bullet\Theta_\Sigma} X(t)\right) > 0 \end{cases}$$

hinzu und gehe zu (6).

– andernfalls (falls Θ unter M_{0_X} unmarkiert ist) füge der Liste $\mathcal{Q}$ die linearen Ungleichungssysteme

$$\mathcal{L}_{i+1}^{1}\colon \begin{cases} \mathcal{L}_i \\ \left(\sum_{t\in{}^\bullet\Theta_\Sigma\setminus\Theta_\Sigma^\bullet} X(t)\right) > 0 & \text{falls } {}^\bullet\Theta_\Sigma \setminus \Theta_\Sigma^\bullet \neq \emptyset \\ \left(\sum_{s\in\Theta} M(s) + \sum_{t\in\Theta_\Sigma^\bullet\setminus{}^\bullet\Theta_\Sigma} X(t)\right) > 0 \end{cases}$$

und

$$\mathcal{L}_{i+1}^{2}\colon \begin{cases} \mathcal{L}_i \\ \left(\sum_{s\in\Theta} M(s) + \sum_{t\in{}^\bullet\Theta_\Sigma\cup\Theta_\Sigma^\bullet} X(t)\right) = 0 \end{cases}$$

hinzu und gehe zu (6).

(6) Falls die Liste $\mathcal{Q}$ von zu lösenden linearen Ungleichungssystemen nicht leer ist, entferne ein Ungleichungssystem $\mathcal{L}$ aus $\mathcal{Q}$, berechne eine Lösung von $\mathcal{L}$ und gehe zu (2), andernfalls gehe zu (8).

(7) Überprüfe, ob der Schaltvektor X in dem S/T-Netzsystem Σ ausgeführt werden kann (siehe dazu Abschnitt 4.3 auf Seite 102). Falls ja, dann beende den Algorithmus mit der Ausgabe $M_{par} \in \mathcal{R}(M_0)$, andernfalls gehe zu (6).

(8) Falls für ein Ungleichungssystem in Schritt (7) gezeigt wurde, daß dessen Schaltvektor X in Σ nicht ausgeführt werden konnte, kann keine Aussage über die Erreichbarkeit der Teilmarkierung M_{par} in Σ getroffen werden, andernfalls (falls für kein Ungleichungssystem Schritt (7) ausgeführt wurde) gilt $M_{par} \notin \mathcal{R}(M_0)$.

Die Terminierung von TrapSchaltCheck kann unter Berücksichtigung der folgenden Aspekte garantiert werden: Wird im Iterationsschritt $i+1$ von TrapSchaltCheck ein Ungleichungssystem vom Typ $\mathcal{L}_{i+1}^{0}$ erzeugt, so bedeutet dieses, daß der Markierungsgleichung entweder eine unter M_0 markierte, aber unter M unmarkierte Teilmenge der Stellen von Σ als Schnittebene hinzugefügt wird, oder eine aus der Stellenmenge Θ ausgehende Transition schalten muß, was in einem anderen Schaltnetz resultiert. Wird ein Ungleichungssystem vom Typ $\mathcal{L}_{i+1}^{1}$ erzeugt, so wird dem Ungleichungssystem eine Schnittebene derart hinzugefügt, daß eine in die Stellenmenge Θ eingehende Transition schalten muß und zusätzlich eine Stelle aus Θ markiert sein muß oder eine aus der

Stellenmenge Θ ausgehende Transition schalten muß, was erneut ein anderes Schaltnetz ergibt. Wird ein Ungleichungssystem vom Typ $\mathcal{L}^2_{i+1}$ generiert, so eliminiert die hinzugefügte Schnittebene alle Stellen aus Θ sowie deren Eingangs- und Ausgangstransitionen aus dem Lösungsraum. Auch diese Vorgehensweise resultiert in einem anderen Schaltnetz. In jedem Iterationsschritt von TrapSchaltCheck wird also ein anderes Schaltnetz von Σ betrachtet. Da Σ höchstens exponentiell viele verschiedene Schaltnetze aufweist, kann folglich die Terminierung von TrapSchaltCheck garantiert werden.
Anhand des sicheren S/T-Netzsystems Σ aus Abbildung 4.1 auf Seite 85 soll nun überprüft werden, ob mittels des Algorithmus' TrapSchaltCheck die Unerreichbarkeit der Teilmarkierung

$$M_{par} = (\emptyset, \{s_2, s_4\})$$

nachgewiesen werden kann. Das lineare Ungleichungssystem

$$\mathcal{L}_0 : \begin{cases} \text{Variablen: } M \text{ binär}, X \geq \mathbf{0} \text{ ganzzahlig} & \\ M(s_1) = 1 - X(t_2) + X(t_4) & \text{(a)} \\ M(s_2) = X(t_1) - X(t_2) & \text{(b)} \\ M(s_3) = X(t_2) - X(t_4) & \text{(c)} \\ M(s_4) = X(t_3) - X(t_4) & \text{(d)} \\ M(s_5) = 1 - X(t_1) + X(t_3) & \text{(e)} \\ M(s_6) = X(t_1) - X(t_3) & \text{(f)} \\ M(s_2) \geq 1 & \text{(g)} \\ M(s_4) \geq 1 & \text{(h)} \end{cases} \tag{4.2.6}$$

besitzt die Lösung

$$M = (1,1,0,1,1,0)^t \quad \text{und} \quad X = (1,0,1,0)^t.$$

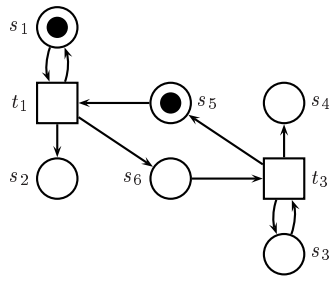
In Schritt (3) von TrapSchaltCheck wird nun das durch den Schaltvektor $X = (1,0,1,0)^t$ induzierte Schaltnetz Σ_X generiert, das in Abbildung 4.3 auf der nächsten Seite dargestellt ist. Nun wird in Schritt (4) nach einer nichtleeren, unter der Markierung $M|_{S_X}$ unmarkierten Falle von Σ_X gesucht. Dazu wird eine Lösung des linearen Ungleichungssystems

$$\mathcal{F}_0 : \begin{cases} \text{Variable: } Y \geq \mathbf{0} \text{ rational} \\ 0 \leq Y(s_1) + 2 \cdot Y(s_2) - Y(s_5) + 2 \cdot Y(s_6) \\ 0 \leq Y(s_3) + 2 \cdot Y(s_4) + 2 \cdot Y(s_5) - Y(s_6) \\ 0 = Y(s_1) + Y(s_2) + Y(s_4) + Y(s_5) \\ 0 < Y(s_1) + Y(s_2) + Y(s_3) + Y(s_4) + Y(s_5) + Y(s_6) \end{cases}$$

berechnet. Eine Lösung von $\mathcal{F}_0$ ist durch

$$Y = (0,0,1,0,0,0)^t$$

Abbildung 4.3 *Durch den Schaltvektor $X = (1,0,1,0)^t$ generiertes Schaltnetzsystem Σ_X*



gegeben, welche die unter der Markierung $M|_{S_X} = \{s_1, s_2, s_4, s_5\}$ unmarkierte Falle $\Theta = \{s_3\}$ von Σ_X induziert. Da Θ unter $M_{0_X} = \{s_1, s_5\}$ auch unmarkiert ist, werden in Schritt (5) von TrapSchaltCheck die beiden linearen Ungleichungssysteme

$$\mathcal{L}_1^1\colon \begin{cases} \mathcal{L}_0 \\ X(t_2) > 0 \\ M(s_3) + X(t_4) > 0 \end{cases} \quad \text{und} \quad \mathcal{L}_1^2\colon \begin{cases} \mathcal{L}_0 \\ M(s_3) + X(t_2) + X(t_3) + X(t_4) = 0 \quad \text{(a)} \end{cases} \tag{4.2.7}$$

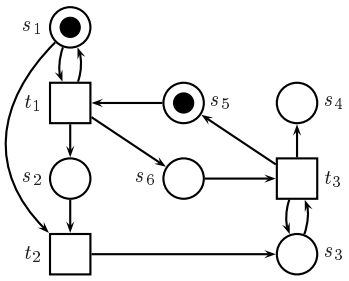
erzeugt. Hieraus wird deutlich, daß mittels TrapSchaltCheck eine falsch-positive Lösung der Markierungsgleichung ausgeschlossen werden kann, die bei Anwendung von TrapCheck nur mittels des Schaltvektortests als ungültige Lösung erkannt wurde.
Das lineare Ungleichungssystem $\mathcal{L}_1^2$ ist unlösbar.
(Aus Gleichung 4.2.7 (a) auf dieser Seite und Ungleichung 4.2.6 (h) auf der vorherigen Seite folgt unmittelbar $X(t_3) = X(t_4) = 0$ und $M(s_4) = 1$. Dieses steht jedoch im Widerspruch zu Gleichung 4.2.6 (d) auf der vorherigen Seite.)
Das lineare Ungleichungssystem $\mathcal{L}_1^1$ besitzt die Lösung

$$M = (0,1,1,1,0,1)^t \quad \text{und} \quad X = (2,1,1,0)^t.$$

Das durch diese Lösung der Markierungsgleichung generierte Schaltnetzsystem Σ_X ist in Abbildung 4.4 auf der nächsten Seite dargestellt. Als nächstes wird mittels des linearen Ungleichungssystems

$$\mathcal{F}_1^1\colon \begin{cases} \text{Variable: } Y \geq \mathbf{0} \text{ rational} \\ 0 \leq Y(s_1) + 2\cdot Y(s_2) - Y(s_5) + 2\cdot Y(s_6) & \text{(a)} \\ 0 \leq -Y(s_1) - Y(s_2) + 2\cdot Y(s_3) & \text{(b)} \\ 0 \leq Y(s_3) + 2\cdot Y(s_4) + 2\cdot Y(s_5) - Y(s_6) & \text{(c)} \\ 0 = Y(s_2) + Y(s_3) + Y(s_4) + Y(s_6) & \text{(d)} \\ 0 < Y(s_1) + Y(s_2) + Y(s_3) + Y(s_4) + Y(s_5) + Y(s_6) & \text{(e)} \end{cases} \tag{4.2.8}$$

Abbildung 4.4 *Durch den Schaltvektor* $X = (2,1,1,0)^t$ *generiertes Schaltnetzsystem* Σ_X



nach einer nichtleeren Falle von Σ_X gesucht, die unter der Markierung $M = \{s_2, s_3, s_4, s_6\}$ unmarkiert ist. Das lineare Ungleichungssystem $\mathcal{F}_1^1$ ist jedoch unlösbar, weshalb es keine solche Falle in Σ_X gibt.
(Aus Gleichung 4.2.8 (d) auf der vorherigen Seite folgt $Y(s_2) = Y(s_3) = Y(s_4) = Y(s_6) = 0$. Damit folgt aus 4.2.8 (b) unmittelbar $Y(s_1) = 0$. Aus 4.2.8 (a) kann dann auch noch $Y(s_5) = 0$ abgeleitet werden, was jedoch im Widerspruch zu Ungleichung 4.2.8 (e) steht.)
Nun muß in Schritt (7) von TrapSchaltCheck überprüft werden, ob der Schaltvektor

$$X = (2,1,1,0)^t$$

in dem S/T-Netzsystem Σ (Abbildung 4.1 auf Seite 85) ausgeführt werden kann (siehe dazu Abschnitt 4.3 auf Seite 102). Da jedoch alle Abläufe von Σ die Form

$$M_0 \xrightarrow{(t_1 t_2 t_3 t_4)^*}$$

aufweisen, ist unmittelbar klar, daß X keinen gültigen Schaltvektor von Σ darstellt. Somit kann auch leider mit dem Algorithmus TrapSchaltCheck keine Aussage bezüglich der Erreichbarkeit der Teilmarkierung

$$M_{par} = (\emptyset, \{s_2, s_4\})$$

in dem S/T-Netzsystem Σ (Abbildung 4.1 auf Seite 85) getroffen werden.

4.2.2 Diskussion der Schnittebenen

In [Mel98] wurde eine andere Strategie für die Auswahl der Schnittebenen gewählt. In diesem Abschnitt sollen die Unterschiede zwischen dem in dieser Arbeit und dem in [Mel98] gewählten Ansatz vorgestellt und diskutiert werden.
Dazu seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ sowie eine Lösung $M \in \{0,1\}^{|S|}$ und $X \in \mathbb{N}^{|T|}$ der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot X$$

von Σ gegeben.
Falls es in dem durch den Schaltvektor X erzeugten Schaltnetzsysten Σ_X eine unter der Markierung M unmarkierte, aber unter der Anfangsmarkierung M_0 markierte Falle Θ gibt, besteht in der Wahl der Schnittebenen kein Unterschied zwischen [Mel98] und dem in dieser Arbeit gewählten Ansatz.
Falls die Falle Θ jedoch auch unter der Anfangsmarkierung M_0 unmarkiert ist, dann werden in der vorliegenden Arbeit die beiden Fälle unterschieden, daß Θ entweder irgendwann oder niemals markiert war, d.h., daß irgendwann eine Transition aus ${}^\bullet\Theta_\Sigma \setminus \Theta_\Sigma^\bullet$ geschaltet hat oder nicht. Aus dieser Fallunterscheidung resultieren im nächsten Iterationsschritt die beiden Ungleichungssysteme

$$\mathcal{L}_{i+1}^1 \colon \begin{cases} \mathcal{L}_i \\ \left(\sum_{t \in {}^\bullet\Theta_\Sigma \setminus \Theta_\Sigma^\bullet} X(t)\right) > 0 \\ \left(\sum_{s \in \Theta} M(s) + \sum_{t \in \Theta_\Sigma^\bullet \setminus {}^\bullet\Theta_\Sigma} X(t)\right) > 0 \end{cases}$$

und

$$\mathcal{L}_{i+1}^2 \colon \begin{cases} \mathcal{L}_i \\ \left(\sum_{s \in \Theta} M(s) + \sum_{t \in {}^\bullet\Theta_\Sigma \cup \Theta_\Sigma^\bullet} X(t)\right) = 0 \end{cases}$$

In [Mel98] hingegen wurde die Fallunterscheidung betrachtet, ob eine „innere“ Transition der Falle Θ, d.h., eine Transition aus ${}^\bullet\Theta_\Sigma \cap \Theta_\Sigma^\bullet$, geschaltet hat oder nicht. Eine „innere“ Transition von Θ kann jedoch nur schalten, falls Θ markiert ist, und da Θ anfänglich unmarkiert ist, muß demzufolge auch eine in die Stellenmenge Θ eingehende Transition, d.h., eine Transition aus ${}^\bullet\Theta_\Sigma \setminus \Theta_\Sigma^\bullet$, schalten. Somit ergeben sich im nächsten Iterationsschritt die Ungleichungssysteme

$$\mathcal{G}_{i+1}^1 \colon \begin{cases} \mathcal{G}_i \\ \left(\sum_{t \in {}^\bullet\Theta_\Sigma \cap \Theta_\Sigma^\bullet} X(t)\right) > 0 \\ \left(\sum_{t \in {}^\bullet\Theta_\Sigma \setminus \Theta_\Sigma^\bullet} X(t)\right) > 0 \\ \left(\sum_{s \in \Theta} M(s) + \sum_{t \in \Theta_\Sigma^\bullet \setminus {}^\bullet\Theta_\Sigma} X(t)\right) > 0 \end{cases}$$

und

$$\mathcal{G}_{i+1}^2 \colon \begin{cases} \mathcal{G}_i \\ \left(\sum_{t \in {}^\bullet\Theta_\Sigma \cap \Theta_\Sigma^\bullet} X(t)\right) = 0 \end{cases}$$

Bei näherer Betrachtung der vier Ungleichungssysteme stellt man fest, daß unter bestimmten Voraussetzungen eine Beziehung zwischen den Ungleichungssystemen hergestellt werden kann: Im Falle $\mathcal{L}_i = \mathcal{G}_i$ gilt offensichtlich

$$\mathcal{G}_{i+1}^1 \Rightarrow \mathcal{L}_{i+1}^1 \quad \text{und} \quad \mathcal{L}_{i+1}^2 \Rightarrow \mathcal{G}_{i+1}^2,$$

d.h., jede Lösung von $\mathcal{G}^1_{i+1}$ stellt auch eine Lösung von $\mathcal{L}^1_{i+1}$ dar, und jede Lösung von $\mathcal{L}^2_{i+1}$ existiert auch in $\mathcal{G}^2_{i+1}$. Allerdings können daraus noch keine weiteren Rückschlüsse auf die Mächtigkeit der Schnittebenen gezogen werden.
Jedoch scheint es die natürlichere Vorgehensweise zu sein, diejenigen Fälle zu unterscheiden, bei denen die Stellenmenge Θ markiert bzw. unmarkiert ist, als das Schalten bzw. Nicht-Schalten der inneren Transitionen von Θ zu betrachten. Zudem scheinen die für $\mathcal{L}^1_{i+1}$ und $\mathcal{L}^2_{i+1}$ gewählten Schnittebenen zwei Vorteile gegenüber den Schnittebenen von $\mathcal{G}^1_{i+1}$ und $\mathcal{G}^2_{i+1}$ aufzuweisen. Falls es in dem Netzsystem Σ keine in die Stellenmenge Θ eingehenden Transitionen gibt, d.h., ${}^\bullet\Theta_\Sigma \subseteq \Theta^\bullet_\Sigma$, dann sind die Ungleichungssysteme $\mathcal{L}^1_{i+1}$ und $\mathcal{G}^1_{i+1}$ unlösbar. Da Θ anfänglich unmarkiert ist, ergibt es sich desweiteren aus der Natur der Markierungsgleichung, daß diese keine Lösung aufweisen kann, unter der eine Stelle aus Θ markiert ist oder eine aus Θ ausgehende Transtion, d.h., eine Transition aus $\Theta^\bullet_\Sigma \setminus {}^\bullet\Theta_\Sigma$, schaltet. Dieses „Wissen“ muß sich das zur Lösung der Ungleichungssysteme verwendete Programm bei der Suche nach einer Lösung für $\mathcal{G}^2_{i+1}$ erst erarbeiten, während es bei $\mathcal{L}^2_{i+1}$ bereits explizit in das Ungleichungssystem hineinkodiert wurde.
Darüberhinaus ist es wünschenswert, die Anzahl der zu lösenden Ungleichungssysteme möglichst klein zu halten. Deshalb sollten Schnittebenen verwendet werden, die den Lösungsraum möglichst stark eingrenzen, denn je stärker sich die Beschränkung des Lösungsraums auf den nächsten Iterationsschritt auswirkt, desto schneller kann vielleicht eine Unlösbarkeit des iterierten Ungleichungssystems herbeigeführt werden. Unter diesem Hintergrund erweist sich die Schnittebene von $\mathcal{L}^2_{i+1}$ in dem Sinne stärker als die Schnittebenen von $\mathcal{G}^1_{i+1}$ und $\mathcal{G}^2_{i+1}$, daß viel mehr Variablen des Ungleichungssystems auf einen exakten Wert festgelegt werden. Damit besteht die Möglichkeit, daß für Ungleichungssysteme, die während der Iterationen aus Ungleichungssystemen vom Typ $\mathcal{L}^2_{i+1}$ hervorgehen, unter Umständen weniger Iterationsschritte zur Unlösbarkeit führen als für Ungleichungssysteme, die aus den Typen $\mathcal{G}^1_{i+1}$ oder $\mathcal{G}^2_{i+1}$ abgeleitet werden. Dieses könnte wiederum dazu führen, daß mit den in dieser Arbeit vorgestellten Schnittebenen bei einigen Beispielen weniger Ungleichungssysteme gelöst werden müßten als mit dem in [Mel98] beschriebenen Ansatz.
Um diese Vermutungen zu unterstützen, wurde auch eine Version von TrapSchaltCheck implementiert, welche die für $\mathcal{G}^1_{i+1}$ und $\mathcal{G}^2_{i+1}$ hergeleiteten Schnittebenen aus [Mel98] verwendet. Ein experimenteller Vergleich der beiden Verfahren und eine ausführliche Diskussion der Ergebnisse werden in Abschnitt 4.5 auf Seite 108 vorgenommen.

4.3 Schaltvektortest

Da die Markierungsgleichung nicht azyklischer S/T-Netzsysteme lediglich ein notwendiges, aber keinesfalls hinreichendes Kriterium für Erreichbarkeit darstellt, muß der Lösungsvektor (Schaltvektor) X der Markierungsgleichung noch auf „Schaltbarkeit“ im S/T-Netzsystem überprüft werden, sofern die erhaltene Lösung nicht mit den in den Abschnitten 4.1.2 auf Seite 86 und 4.2.1 auf Seite 93 vorgestellten Verfahren TrapCheck und TrapSchaltCheck ausgeschlossen werden kann.

Die Schaltbarkeitsanalyse von X kann auf ein einfaches Erreichbarkeitsproblem in dem durch den Schaltvektor X generierten Schaltnetzsystem Σ_X reduziert werden. Dazu wird Σ_X für jede Transition t um $X(t)$ Stellen erweitert, welche das Vorkommen von t in einem Ablauf von Σ_X zählen. Mittels dieser Modifizierung ist der Schaltvektor X genau dann in Σ ausführbar, wenn in dem modifizierten Schaltnetzsystem Σ_X eine Markierung erreichbar ist, sodaß für jede Transition t von Σ_X genau die Stelle markiert ist, welche angibt, daß die Transition $X(t)$-mal geschaltet hat.
Im folgenden wird das modifizierte Schaltnetzsystem formal definiert, das im weiteren Verlauf auch als Testnetzsystem Σ_T bezeichnet wird.

Definition 4.3.1 (Testnetzsystem Σ_T)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, ein Schaltvektor $X \in \mathbb{N}^{|T|}$ von Σ sowie das durch den Schaltvektor X generierte Schaltnetzsystem $\Sigma_X = (N_X, M_{0_X})$ mit $N_X = (S_X, T_X, F_X)$. Das Tupel $\Sigma_T = (N_T, M_{0_T})$ wird als das durch den Schaltvektor X generierte *Testnetzsystem* von Σ bezeichnet, falls $N_T = (S_T, T_T, F_T)$ ein S/T-Netzsystem beschreibt, sodaß

$$
\begin{aligned}
S_T &= S_X \cup \{s_{t^i} \mid t \in T_X \wedge 0 \leq i \leq X(t)\} \cup \{s'\} \\
T_T &= \{t^i \mid t \in T_X \wedge 1 \leq i \leq X(t)\} \cup \{t'\} \\
F_T &= \{(s, t^i) \mid s \in S_X \wedge t^i \in T_T \wedge (s,t) \in F_X\} \cup \\
&\quad \{(t^i, s) \mid s \in S_X \wedge t^i \in T_T \wedge (t,s) \in F_X\} \cup \\
&\quad \{(s_{t^i}, t^{i+1}) \mid s_{t^i} \in S_T \wedge t^{i+1} \in T_T \wedge 0 \leq i < X(t)\} \cup \\
&\quad \{(t^i, s_{t^i}) \mid s_{t^i} \in S_T \wedge t^i \in T_T \wedge 1 \leq i \leq X(t)\} \cup \\
&\quad \{(s_{t^i}, t') \mid s_{t^i} \in S_T \wedge i = X(t)\} \cup \\
&\quad \{(t', s')\}
\end{aligned}
$$

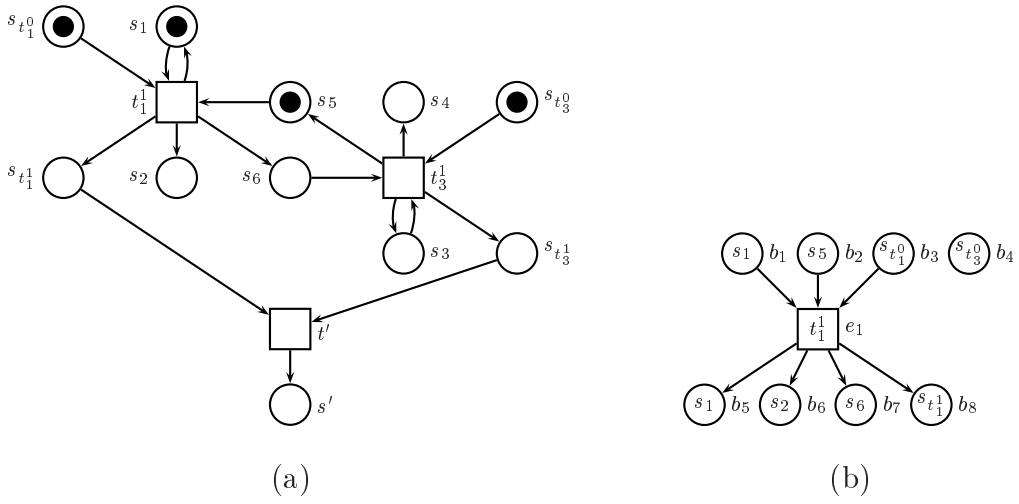
und M_{0_T} eine Markierung von N_T beschreibt, wobei für jedes $s \in S_T$ gilt:

$$
M_{0_T}(s) = \begin{cases} 1 & \text{falls } s \in \{s_{t^0} \mid t \in T_X\} \\ 0 & \text{falls } s \in \{s_{t^i} \mid t \in T_X \wedge i \geq 1\} \cup \{s'\} \\ M_{0_X}(s) & \text{sonst} \end{cases}
$$

Als direkte Folgerung aus Definition 4.3.1 von Σ_T ergibt sich Satz 4.3.1.

Satz 4.3.1 (Schaltbarkeit eines Transitionsvektors)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, ein Schaltvektor $X \in \mathbb{N}^{|T|}$ von Σ sowie das durch den Schaltvektor X generierte Testnetzsystem $\Sigma_T = (N_T, M_{0_T})$ mit $N_T = (S_T, T_T, F_T)$. Der Schaltvektor X ist genau dann in Σ ausführbar, wenn die Teilmarkierung $M_{par} = (\emptyset, \{s'\})$ in dem Testnetzsystem Σ_T von der Anfangsmarkierung M_{0_T} aus erreichbar ist, d.h., $M_{par} \in \mathcal{R}(M_{0_T})$.

Abbildung 4.5 *Durch den Transitionsvektor $X = (1,0,1,0)^t$ generiertes Testnetzsystem Σ_T (a) und dessen endliches, vollständiges Präfix (b)*



In Abschnitt 4.1.2 auf Seite 86 wurde gezeigt, daß die Markierungsgleichung des sicheren S/T-Netzsystems Σ aus Abbildung 4.1 auf Seite 85 die Lösung

$$M = (1,1,0,1,1,0)^t \quad \text{und} \quad X = (1,0,1,0)^t$$

aufweist, die mit dem Verfahren TrapCheck nicht ausgeschlossen werden kann, da Σ keine unter $M_0 = \{s_1, s_5\}$ $(\subset M)$ markierte, aber unter M unmarkierte Falle aufweist. Da die Markierungsgleichung lediglich ein notwendiges, jedoch keinesfalls hinreichendes Kriterium für Erreichbarkeit darstellt, muß in Schritt (5) des Algorithmus' TrapCheck noch überprüft werden, ob der Schaltvektor

$$X = (1,0,1,0)^t$$

auch tatsächlich in Σ schalten kann.
Dazu wird das durch den Schaltvektor X induzierte Testnetzsystem Σ_T erzeugt, das in Abbildung 4.5 (a) auf dieser Seite dargestellt ist. Nach Satz 4.3.1 auf der vorherigen Seite ist der Schaltvektor X genau dann in dem S/T-Netzsystem Σ (Abbildung 4.1 auf Seite 85) ausführbar, wenn die Teilmarkierung

$$M_{par} = (\emptyset, \{s'\})$$

in Σ_T von M_{0_T} aus erreichbar ist. In Kapitel 3 wurden bereits einige exakte, effiziente Verfahren zur Erreichbarkeitsanalyse von Teilmarkierungen in sicheren S/T-Netzsystemen vorgestellt. Da hier nur jeweils eine Teilmarkierung pro Testnetzsystem betrachtet wird, bietet es sich an, das in Abschnitt 3.4.3 auf Seite 76 vorgestellte Verfahren OnTheFly zum Nachweis oder Ausschluß der Erreichbarkeit von M_{par} einzusetzen. Dabei wird

das Testnetzsystem Σ_T entfaltet, bis entweder ein Repräsentant der Transition t' in die Entfaltung eingefügt werden kann, d.h., $M_{par} \in \mathcal{R}(M_{0_T})$, oder das endliche, vollständige Präfix von Σ_T erzeugt wurde, d.h., $M_{par} \notin \mathcal{R}(M_{0_T})$. Im Falle des Testnetzsystems Σ_T aus Abbildung 4.5 (a) auf der vorherigen Seite erhält man das endliche, vollständige Präfix aus Abbildung 4.5 (b), welches keinen Repräsentanten für t' aufweist. Daraus folgt, daß die Teilmarkierung M_{par} in Σ_T von der Anfangsmarkierung M_{0_T} aus unerreichbar ist und der Schaltvektor $X = (1, 0, 1, 0)^t$ in dem S/T-Netzsystem Σ aus Abbildung 4.1 auf Seite 85 nicht geschaltet werden kann.
Das in diesem Abschnitt vorgestellte Verfahren zum Schaltvektortest erweist sich als sehr effizient, da in vielen Fällen nur wenige Transitionen schalten müssen, um die Erreichbarkeit einer Teilmarkierung nachzuweisen. Die durch die Schaltvektoren generierten Testnetzsysteme sind also oftmals sehr viel kleiner als deren ursprünglichen Netzsysteme und lassen sich folglich viel schneller entfalten.

4.4 Implementierungsaspekte

In diesem Abschnitt sollen einige Optimierungen und Heuristiken bezüglich der Implementierung der in den Abschnitten 4.1.2 auf Seite 86 und 4.2.1 auf Seite 93 vorgestellten Verfahren TrapCheck und TrapSchaltCheck diskutiert werden.
Zu Beginn überprüfen beide Algorithmen, ob eine zu testende Teilmarkierung M_{par} von der Anfangsmarkierung M_0 des S/T-Netzsystems überdeckt wird. In diesem Fall werden die Algorithmen unmittelbar mit dem Ergebnis beendet, daß die Teilmarkierung M_{par} bereits anfänglich vorliegt.
Andernfalls weisen beide Methoden zunächst einmal die Gemeinsamkeit auf, daß für ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ Lösungen $M \in \{0, 1\}^{|S|}$ und $X \in \mathbb{N}^{|T|}$ der Markierungsgleichung

$$M = M_0 + \mathbf{N} \cdot X$$

von Σ berechnet werden, die dann ggf. im Zuge weiterer Iterationen um zusätzliche Schnittebenen ergänzt wird. Da Interesse darin besteht, sowohl die Schaltnetzsysteme, welche nach unter der Markierung M unmarkierten Fallen durchsucht werden, als auch die Testnetzsysteme, anhand derer der Schaltvektortest durchgeführt wird, so klein wie möglich zu halten, und man zudem auch die kürzeste Schaltfolge ermitteln möchte, die zu einer erreichbaren Markierung führt, wird die Markierungsgleichung um die Zielfunktion

$$\textit{Minimiere} \sum_{t \in T} X(t)$$

erweitert.
Im Zuge weiterer Iterationsschritte von TrapCheck und TrapSchaltCheck werden der Markierungsgleichung von Σ Schnittebenen der Art

$$\Big(\sum_{s \in \Theta} M(s)\Big) > 0 \quad \text{und} \quad \Big(\sum_{t \in \Theta_\Sigma^\bullet \setminus {}^\bullet\Theta_\Sigma} X(t)\Big) > 0$$

zum Ausschluß falsch positiver Lösungen hinzugefügt. Je kleiner $|\Theta|$ und $|\Theta^{\bullet}_{\Sigma} \setminus {}^{\bullet}\Theta_{\Sigma}|$ sind, desto stärker grenzen die Schnittebenen den Lösungsraum der Markierungsgleichung ein. Deshalb soll nach Möglichkeit nach minimalen Fallen in dem Sinne gesucht werden, daß die Menge der die Falle bildenden Stellen minimale Kardinalität aufweist, d.h.,

$$\textit{Minimiere } |\{s \in S \mid Y(s) > 0\}| \tag{4.4.1}$$

Würde man in Satz 4.1.2 auf Seite 86 lediglich ein binäres Ungleichungssystem betrachten, ließe sich Zeile 4.4.1 zwar durch die Zielfunktion

$$\textit{Minimiere } \sum_{s \in S} Y(s) \tag{4.4.2}$$

ausdrücken, aber man würde eventuell Lösungen verlieren. Für ein rationales Ungleichungssystem kann Zeile 4.4.1 jedoch nicht mehr durch eine lineare Funktion ausgedrückt werden, sodaß in diesem Fall trotzdem die Zielfunktion 4.4.2 als gute Näherung verwendet wird, wohlweislich, daß die Lösung nicht optimal in dem Sinne sein muß, daß sie eine minimale Falle liefert.
In Schritt (5) von TrapSchaltCheck (Abschnitt 4.2.1 auf Seite 93) wird nach einer Falle gesucht, die in dem vom Schaltvektor X generierten Schaltnetzsystem Σ_X unter der Markierung $M|_{S_X}$ unmarkiert ist. Danach werden dann im nächsten Iterationsschritt von TrapSchaltCheck jeweils ein bzw. zwei neue Ungleichungssysteme betrachtet, je nachdem, ob die Falle anfänglich (also unter M_{0_X}) markiert oder unmarkiert ist. Natürlich soll die Anzahl der zu lösenden Ungleichungssysteme möglichst klein gehalten werden, und deshalb scheint es sich als gute Strategie zu erweisen, zunächst, sofern das durch den Schaltvektor X generierte Schaltnetzsystem Σ_X eine unter der Anfangsmarkierung M_{0_X} markierte Stelle enthält, mit dem in Satz 4.1.2 auf Seite 86 vorgestellten Verfahren nach einer unter M_{0_X} markierten und unter $M|_{S_X}$ unmarkierten Falle von Σ_X zu suchen, da im Falle der Existenz einer solchen Falle im nächsten Iterationsschritt von TrapSchaltCheck lediglich ein neues Ungleichungssystem betrachtet werden muß. Falls Σ_X keine anfänglich markierte Stelle aufweist oder keine unter M_{0_X} markierte und unter $M|_{S_X}$ unmarkierte Falle gefunden werden konnte, wird nach einer nicht leeren Falle von Σ_X gesucht, die sowohl unter M_{0_X} als auch unter $M|_{S_X}$ unmarkiert ist. Dazu wird die Ungleichung 4.1.2 aus dem in Satz 4.1.2 auf Seite 86 eingeführten Ungleichungssystem entfernt und durch die Schnittebenen

$$\begin{aligned} Y^t \cdot M_{0_X} &= 0 \\ \Big(\sum_{s \in S_X \setminus M_{0_X}} Y(s) \Big) &> 0 \end{aligned}$$

ersetzt.

4.4.1 Wie geht es nach erfolglosem Schaltvektortest weiter?

Eine offene Fragestellung besteht darin, wie mit den Algorithmen TrapCheck und TrapSchaltCheck fortgefahren werden könnte, falls der Schaltvektortest ergibt, daß

der Schaltvektor (Lösungsvektor) X_i einer Markierungsgleichung in dem der Markierungsgleichung zugrundeliegenden S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ nicht schalten kann. Da diese Lösung nun nicht mehr mittels des Fallenkonzeptes ausgeschlossen werden kann, müßte nach einer anderen Ausschlußmöglichkeit für X_i gesucht werden.
Leider besteht in linearen Ungleichungssystemen nicht die Möglichkeit, die Markierungsgleichung im nächsten Iterationsschritt um $|T|$ Schnittebenen der Art

$$X_{i+1}(t_j) \neq X_i(t_j)$$

für $1 \leq j \leq |T|$ zu erweitern. Auch über die Wahl der Zielfunktion gibt es keine Möglichkeit, die Lösung X als Ganzes auszuschließen, sofern man den Bereich der linearen Programmierung nicht verlassen möchte.
Man könnte sich überlegen, eine ausführliche Fallunterscheidung derart vorzunehmen, daß im nächsten Iterationsschritt für jede Transition $t_k \in T$ zwei Ungleichungssysteme der Form

$$\mathcal{L}_{i+1}^1 = \begin{cases} \mathcal{L}_i \\ X_{i+1}(t_j) = X_i(t_j) \quad \text{für } 1 \leq j < k \\ X_{i+1}(t_k) > X_i(t_k) \end{cases}$$

und

$$\mathcal{L}_{i+1}^2 = \begin{cases} \mathcal{L}_i \\ X_{i+1}(t_j) = X_i(t_j) \quad \text{für } 1 \leq j < k \\ X_{i+1}(t_k) < X_i(t_k) \end{cases}$$

betrachtet werden. Diese Vorgehensweise entspräche jedoch einer exponentiellen Suche, und die Anzahl der zu lösenden Ungleichungssysteme könnte bereits im nächsten Iterationsschritt explodieren. Deshalb scheint sich eine solche Fallunterscheidung als wenig sinnvoll zu erweisen.
Eine weniger kostenintensive Vorgehensweise könnte darin bestehen, eine Fallunterscheidung derart vorzunehmen, daß zufällig eine beliebige Teilmenge $T' \subseteq T$ bestimmt wird, für die dann im nächsten Iterationsschritt zwei Ungleichungssysteme der Form

$$\mathcal{L}_{i+1}^1 = \begin{cases} \mathcal{L}_i \\ \left(\sum_{t \in T'} X_{i+1}(t)\right) > \left(\sum_{t \in T'} X_i(t)\right) \end{cases}$$

und

$$\mathcal{L}_{i+1}^1 = \begin{cases} \mathcal{L}_i \\ \left(\sum_{t \in T'} X_{i+1}(t)\right) < \left(\sum_{t \in T'} X_i(t)\right) \end{cases}$$

betrachtet werden. Dadurch könnte sichergestellt werden, daß sich die Anzahl der zu lösenden Ungleichungssysteme mit jedem Iterationsschritt höchstens verdoppelt. Allerdings würde dieser Ansatz das Problem aufweisen, daß ganze Ebenen des Suchraums ausgeblendet werden. Das hätte wiederum zur Folge, daß die Erreichbarkeit einer Teilmarkierung nun nicht mehr exakt ausgeschlossen werden kann, da in den ausgeblendeten Suchraumebenen eine gültige Lösung liegen könnte. Von daher wäre es auch nicht sinnvoll, den Verifikationsprozeß für eine Markierung, für die Erreichbarkeit ausgeschlossen werden soll, an diesem Punkt unnötig durch weitere Iterationen zu verlängern, wohlweislich, daß man das gewünschte Ergebnis nicht mehr nachweisen kann. Deshalb wird auch dieser Ansatz nicht weiter verfolgt.
Eine Lösung des Problems scheint mit dem Release der Version 5.5 des unter LGPL lizensierten MIP-Solvers LpSolve vom 17. Mai 2005 vorzuliegen, welcher nun die Möglichkeit bietet, alle Lösungen des linearen Ungleichungssystems zu berechnen, die denselben optimalen Wert für die Zielfunktion liefern. Somit könnten zunächst alle neben X liegenden Lösungen derselben „Länge“ untersucht werden, und falls keine der Lösungen zum gewünschten Ergebnis führen würde, könnte die Markierungsgleichung im nächsten Iterationsschritt um die Schnittebene

$$\Big(\sum_{t\in T} X_{i+1}(t)\Big) > \Big(\sum_{t\in T} X_i(t)\Big)$$

erweitert werden. Auf diesem Wege würde man keine Suchraumebenen ausblenden und könnte auch weiterhin die Erreichbarkeit einer Teilmarkierung exakt ausschließen.
Da jedoch der zeitliche Rahmen zwischen dem Release der Version 5.5 von LpSolve und der Fertigstellung dieser Arbeit eng bemessen war, konnte zwar einerseits die neuste LpSolve-Version noch in die Implementierungen von TrapCheck und TrapSchaltCheck integriert und in den Testergebnissen von Abschnitt 4.5 berücksichtigt werden, andererseits aber die obige Strategie nicht mehr implementiert werden.
Die experimentellen Ergebnisse zeigen jedoch deutlich, daß die beiden Verfahren TrapCheck und TrapSchaltCheck auch ohne diese Heuristik bereits sehr hohe Erfolgsquoten bezüglich des Nachweises oder Ausschlusses der Erreichbarkeit von Teilmarkierungen aufweisen, sodaß eine Berücksichtigung der Heuristik nicht notwendigerweise zu einer qualitativen Verbesserung der beiden Methoden führen müßte.

4.5 Experimentelle Ergebnisse

In diesem Abschnitt werden Testergebnisse zur Erreichbarkeitsanalyse von Teilmarkierungen in sicheren S/T-Netzsystemen mittels der in den vorherigen Abschnitten beschriebenen Verfahren TrapCheck und TrapSchaltCheck präsentiert. Dabei wurden sowohl die Beispiele von Corbett [Cor94] als auch Modellierungen anderen Ursprungs in die Tests einbezogen. Eine Beschreibung aller betrachteten Systeme kann Anhang A auf Seite 223 entnommen werden.
Die zu testenden Teilmarkierungen wurden wiederum mit einem Zufallsgenerator erzeugt, der für jede der ausgewählten Stellen erneut im Verhältnis 4:1 festgelegt hat, ob

$M_{par}(s) = 1$ oder $M_{par}(s) = 0$ getestet werden sollte. Es wurden jeweils 25 Teilmarkierungen der Größe 2, 4, 6 und 8 generiert, sodaß für jedes System jeweils 100 Teilmarkierungen auf Erreichbarkeit überprüft wurden.
Alle Testreihen wurden auf einem Linux PC durchgeführt, der mit einem 2,4 GHz Intel(R) Xeon(TM) Prozessor und 4 GByte Hauptspeicher ausgestattet war.
Zum Lösen der linearen Ungleichungssysteme wurde LpSolve in der Version 5.5 vom 17. Mai 2005 verwendet. LpSolve wurde in allen Testreihen mit seinen Standardeinstellungen aufgerufen, einzig die maximale Suchtiefe für den zur Lösung ganzzahliger Ungleichungssysteme verwendeten Branch-and-Bound-Algorithmus mußte auf eine unbegrenzte Tiefe erweitert werden, da die voreingestellte maximale Suchtiefe in einigen Testfällen sehr schnell erreicht wurde. Eine ausführliche Diskussion bezüglich der Wahl geschickter Heuristiken für die Auswahl des Pivotelements beim dualen Simplex als auch der Wahl einer Branching-Strategie, die in erheblichem Maße die Größe des Branching-Baums bestimmt, wurde in [Mel98] geführt. Darüberhinaus haben sich die Standardeinstellungen von LpSolve bewährt, also die Verwendung einer Kombination aus dualem und primalen Simplex sowie die Wahl der nicht-ganzzahligen Variablen mit dem kleinsten Index als Branching-Strategie, sodaß von weiteren Untersuchungen diesbezüglich abgesehen wurde.
Tabelle 4.1 auf der nächsten Seite zeigt die experimentellen Ergebnisse der Erreichbarkeitsanalyse mittels TrapCheck und TrapSchaltCheck, wobei TrapSchaltCheck einmal die in Abschnitt 4.2.2 auf Seite 100 diskutierten Ungleichungssysteme vom Typ $\mathcal{L}_{i+1}$ ($\mathcal{L}$-Version) und das andere Mal die Ungleichungssysteme vom Typ $\mathcal{G}_{i+1}$ ($\mathcal{G}$-Version) verwendete. Die mit $|S|$ und $|T|$ bezeichneten Spalten geben die Anzahl von Stellen und Transitionen der Netzsysteme an. In den „?“-Spalten ist die Anzahl von Teilmarkierungen notiert, für die keine Aussage (keine „Ja/Nein“-Antwort) bezüglich der Erreichbarkeit abgeleitet werden konnte. In den mit „GL“ bezeichneten Spalten steht die maximale Anzahl von Ungleichungssystemen, die im Zuge der Iterationen erzeugt wurden. Die in Sekunden gemessenen Verifikationszeiten sind in den mit „t“ bezeichneten Spalten angegeben.
Um die Tabelle übersichtlicher zu gestalten, wurden für TrapSchaltCheck nur in den Fällen Einträge für die $\mathcal{G}$-Version vorgenommen, in denen ein Unterschied zu den Ergebnissen der $\mathcal{L}$-Version bestand.
Zunächst ist auffällig, daß die Verifikationszeiten der Algorithmen für die meisten Systeme bei unter einer Sekunde liegen und zudem in vielen Fällen eine hundertprozentige Aussage über den Nachweis oder den Ausschluß der Erreichbarkeit von Teilmarkierungen getroffen werden kann, wobei TrapSchaltCheck eine deutliche höhere Erfolgsquote als TrapCheck aufweist. Mit Ausnahme des Systems Dme(9), bei dem die Erfolgsquoten der beiden Algorithmen bei lediglich 22 bzw. 44 Prozent liegen, kann für die übrigen Systeme im Falle von TrapCheck eine Erfolgsquote von über 72 Prozent und im Falle von TrapSchaltCheck eine Erfolgsquote von über 92 Prozent festgehalten werden. Dieses Ergebnis ist nicht unbedingt zu erwarten gewesen, da die Markierungsgleichung allgemeiner Petrinetzsysteme lediglich ein notwendiges, aber keinesfalls hinreichendes Kriterium für Erreichbarkeit darstellt. Allerdings bestätigen die hohen Erfolgsquoten ei-

Tabelle 4.1 *Experimentelle Ergebnisse zur Erreichbarkeitsanalyse für* TrapCheck *und* TrapSchaltCheck.

			TrapCheck			TrapSchaltCheck $\mathcal{L}$-Version			TrapSchaltCheck $\mathcal{G}$-Version		
	$\|S\|$	$\|T\|$	?	GL	t	?	GL	t	?	GL	t
Abp	61	95	7	5	0.04	0	27	0.05			
Bds	79	59	2	3	0.01	0	3	0.01			
Buffer(50)	100	51	0	1	0.02	0	1	0.02			
Cyclic(12)	131	71	0	9	0.03	0	26	0.05			
Dac(9)	82	43	0	1	0.01	0	1	0.01			
Dijkstra(2)	68	86	4	10	0.03	1	59	0.03			
Dme(9)	202	147	78	3	0.03	56	830	2.42	56	1575	3.92
Dp(12)	120	48	0	1	0.01	0	1	0.01			
Dpd(7)	98	63	1	6	0.02	0	6	0.01			
Dpfm(5)	37	41	0	1	0.01	0	1	0.01			
Dph(6)	94	92	0	1	0.03	0	1	0.01			
Eisenbahn	44	44	12	2	0.01	0	9	0.01			
Elevator(3)	369	782	2	3	0.35	0	18	0.32			
Furnace(3)	83	99	28	3	0.01	2	8	0.02	1	8	0.02
Gasnq(3)	181	223	0	1	0.06	0	1	0.05			
Gasq(3)	316	475	0	1	0.16	0	1	0.16			
Hartstone(25)	153	52	0	1	0.02	0	1	0.03			
Key(3)	180	133	3	4	0.99	1	18	0.17			
Mmgt(2)	104	114	0	1	0.02	0	1	0.02			
Over(5)	141	95	8	3	0.17	6	53	0.18	6	53	0.19
Prodcell(5)	231	202	13	7	5.27	0	114	1.98	0	117	2.00
Ring(9)	162	99	1	8	18.87	0	13	0.09			
Rw(12)	111	313	0	1	0.05	0	1	0.05			
Rw(2)(1)	72	88	12	8	0.13	8	87	0.29	9	89	0.28
Sentest(50)	287	77	0	1	0.03	0	1	0.03			
Slotring(9)	60	60	27	1	0.01	5	4	0.01			
Speed	43	31	7	3	0.01	0	8	0.01	0	11	0.01
Telephon(3)	232	672	4	8	84.34	4	44	4.17			

nerseits und die durchweg geringen Verifikationszeiten andererseits, daß beide Verfahren eine Relevanz für praktische Belange besitzen.
Darüberhinaus scheint sich TrapSchaltCheck in dem Sinne qualitativ besser als TrapCheck zu erweisen, daß für viele Systeme die Erfolgsquote der „Ja/Nein"-Antworten von TrapSchaltCheck über der Erfolgsquote von TrapCheck liegt. Allerdings zeigen die Ergebnisse auch, daß die Anzahl der zu lösenden Ungleichungssysteme bei TrapSchaltCheck schnell explodieren kann, wie beispielsweise anhand von Dme(9), Prodcell(5) oder auch Rw(2)(1) zu sehen ist.
Die $\mathcal{L}$- und die $\mathcal{G}$-Version von TrapSchaltCheck haben bei lediglich sechs Systemen unterschiedliche Ergebnisse aufgewiesen. Dabei hat die $\mathcal{L}$-Version in 59 Fällen teilweise erheblich weniger Ungleichungssysteme als die $\mathcal{G}$-Version erzeugt, während der umge-

kehrte Effekt bei lediglich vier Markierungen in sehr schwacher Form aufgetreten ist. Beispielsweise terminierte bei dem DME(9)-System die $\mathcal{L}$-Version von TRAPSCHALTCHECK nach der Berechnung von 589 Ungleichungssystemen innerhalb von 10 Sekunden, wohingegen die $\mathcal{G}$-Version 32 Sekunden benötigte, um 1575 Ungleichungssysteme abzuarbeiten.
Mit Ausnahme eines Falles, bei dem die beiden Versionen von TRAPSCHALTCHECK mit einer „Ja“- bzw. „?“-Antwort terminierten, sind alle Aussagen bezüglich (Un-)Erreichbarkeit identisch. Daraus läßt sich die Vermutung ableiten, daß die gewählten Schnittebenen bei den in dieser Arbeit verwendeten Beispielen eine ähnliche Mächtigkeit aufweisen.
Abschließend soll noch erwähnt werden, daß sowohl TRAPCHECK als auch TRAPSCHALTCHECK in drei bzw. fünf Fällen innerhalb von drei Stunden Rechenzeit kein Ergebnis lieferten. Da diese Fälle bei verschiedenen Systemen auftraten, wurden sie aus den Ergebnislisten gestrichen, da sie den Gesamteindruck verzerren würden. In 34 Fällen beendete LPSOLVE den Verifikationsvorgang aufgrund numerischer Instabilitäten oder anderer Fehlermeldungen. Bei insgesamt 2800 durchgeführten Testfällen liegt ein solches Ergebnis jedoch noch im Toleranzbereich.

Kapitel 5

Reduktionstechniken für temporallogische Eigenschaften

Während in den Kapiteln 3 und 4 effiziente Algorithmen für die Verifikation von Sicherheitseigenschaften betrachtet wurden, die sich auf Erreichbarkeitsfragen zurückführen lassen, werden in diesem Kapitel ein Verfahren und verschiedene Reduktionstechniken für den Nachweis von Sicherheits- und Lebendigkeitseigenschaften in Form von temporallogischen Aussagen behandelt.
Bei der Verifikation von Erreichbarkeitseigenschaften genügt es, lediglich endliche Anfangsstücke aller Systemabläufe zum Nachweis einer solchen Eigenschaft zu betrachten. Bei der Verifikation von Lebendigkeitseigenschaften geht es jedoch vielmehr um die Fragestellung, ob ein System beispielsweise immer wieder eine bestimmte Aktion ausführen oder immer wieder ein spezieller Systemzustand eintreten wird. Zu diesem Zweck muß für die Verifikation von Lebendigkeitseigenschaften über alle unendlichen Systemabläufe argumentiert werden. Übertragen auf Petrinetze sind beispielsweise Fragestellungen interessant, ob eine Stelle s in jedem Systemablauf immer wieder irgendwann markiert sein wird, oder ob s irgendwann eine Marke erhalten und diese dann nie mehr verlieren wird. Derartige Aussagen lassen sich nicht mehr auf einfache Erreichbarkeitseigenschaften zurückführen.
Deshalb werden zur Spezifikation solcher Aussagen im allgemeinen Linearzeit-Temporallogiken (LTL) verwendet, die beispielsweise in [Pnu77, Eme90] beschrieben worden sind. Ein grundlegender Ansatz für die Verifikation von LTL-Eigenschaften ist von Vardi und Wolper [Var96, VW86a] vorgeschlagen worden. Dieser basiert auf automatentheoretischen Überlegungen und besteht im wesentlichen aus den folgenden drei Schritten:
Gegeben seien ein Automat $\mathcal{A}_S$, der das System beschreibt, und eine LTL-Eigenschaft φ.

1. Konstruiere einen Büchiautomaten $\mathcal{A}_{\neg\varphi}$, der genau diejenigen Abläufe akzeptiert,

welche die Eigenschaft φ verletzen.

2. Konstruiere aus den Automaten $\mathcal{A}_S$ und $\mathcal{A}_{\neg\varphi}$ einen Produktautomaten $\mathcal{A}_{S\times\neg\varphi}$, der genau diejenigen Abläufe akzeptiert, welche sowohl von $\mathcal{A}_S$ als auch von $\mathcal{A}_{\neg\varphi}$ akzeptiert werden.

3. Überprüfe den Produktautomaten $\mathcal{A}_{S\times\neg\varphi}$ auf Leerheit, d.h., ob $L(\mathcal{A}_{S\times\neg\varphi}) = \emptyset$.

Das System $\mathcal{A}_S$ erfüllt genau dann die LTL-Eigenschaft φ, wenn $L(\mathcal{A}_{S\times\neg\varphi}) = \emptyset$, d.h., wenn es keinen Systemablauf gibt, der φ verletzt.
Esparza und Heljanko haben diesen automatentheoretischen Ansatz auf die Petrinetzebene übertragen und in [EH00a, EH01, Hel02] einen entfaltungsbasierten Ansatz für die Verifikation von LTL-X-Eigenschaften in nebenläufigen Systemen beschrieben. Dabei wird ähnlich wie in dem automatentheoretischen Ansatz von Vardi und Wolper ein Produktnetzsystem aus dem ursprünglichen Netzsystem und einem Büchiautomaten konstruiert, der genau diejenigen Abläufe akzeptiert, welche die zu verifizierende LTL-X-Eigenschaft verletzen. Der Nachweis der LTL-X-Eigenschaft wird dann auf zwei einfache Probleme zurückgeführt, welche die unendlichen Schaltfolgen des Produktnetzsystems betreffen. Zur Lösung dieser Probleme werden Entfaltungstechniken eingesetzt, um im Netzsystem vorhandene Nebenläufigkeiten auszunutzen und somit eine kompakte Repräsentation des Zustandsraums zu erhalten.
Das Verifikationsverfahren von Esparza und Heljanko basiert auf sicheren S/T-Netzsystemen. Diese sind jedoch für Modellierungszwecke relativ ungeeignet, da es bereits für Systeme mit mehreren hundert Stellen und Transitionen eine äußerst zeitaufwendige und fehleranfällige Angelegenheit darstellt, alle Stellen, Transitionen und Kanten manuell zu erzeugen. Vielmehr möchte man Systeme in komfortableren, programmiersprachenähnlichen Formalismen spezifizieren. Eine solche Möglichkeit bietet beispielsweise das Model-Checking Kit [SSE03], welches eine Sammlung von Programmen darstellt, die es einem erlauben, Systeme mit endlichem Zustandsraum in verschiedenen Formalismen zu modellieren und anschließend mit unterschiedlichen Algorithmen zu verifizieren. Darunter befinden sich Algorithmen für den Nachweis von Verklemmungsfreiheit und Erreichbarkeitsfragen sowie Verfahren für die Verifikation von temporallogischen Eigenschaften. Das interessanteste Merkmal des Kits besteht darin, daß verschiedene Verifikationsalgorithmen auf dasselbe System angewendet werden können, unabhängig von dem Formalismus, in dem das System ursprünglich modelliert wurde. Erhält das Kit ein Modell und eine nachzuweisende Eigenschaft als Eingabe, die in einem vom Kit unterstützten Formalismus beschrieben sind, so

1. übersetzt es diese in ein sicheres S/T-Netzsystem[1] und eine entsprechene Eigenschaft über dem Netzsystem,

[1] Als Formalismus wurden sichere S/T-Netzsysteme ausgewählt, da sie eine einfache Struktur aufweisen und eine einfache Definition von unabhängigen Aktionen erlauben, welches sie zu einer geeigneten „Assemblersprache“ für nebenläufige Systeme macht.

2. überführt das sichere S/T-Netzsystem und die Eigenschaft in das Eingabeformat des ausgewählten Model-Checkers und

3. interpretiert die Ausgabe des Model-Checkers auf der Ebene des Formalismus', in dem das System ursprünglich modelliert wurde.

Durch die automatische Übersetzung der Systemmodelle in sichere S/T-Netzsysteme können unter Umständen Redundanzen entstehen, die zu unnötig großen Netzsystemen führen. Aus diesem Grund werden in diesem Kapitel Techniken beschrieben, mit deren Hilfe ein Großteil solcher Redundanzen wieder entfernt werden kann, bevor der eigentliche Verifikationsvorgang beginnt. Dabei wird unter anderem auf bekannte Petrinetztechniken zurückgegriffen, wie zum Beispiel Invarianten [DE95, DNR96, Des98, Rei85], implizite Stellen [CS90] oder lokale Netzreduktionen [Ber85, PPP00]. Mittels Invarianten und impliziten Stellen können Stellen und Transitionen entfernt werden, die das Verhalten eines Netzsystems nicht beeinflussen. Lokale Netzreduktionen verkleinern ein Netzsystem unter der Garantie, daß Verhaltensanalysen des reduzierten und des ursprünglichen Netzsystems dieselben Ergebnisse liefern.
Da in diesem Kapitel speziell Reduktionstechniken für das von Esparza und Heljanko beschriebene Verfahren zum Nachweis von LTL-X-Eigenschaften [EH00a, EH01, Hel02] betrachtet werden, können die Bedingungen für die Anwendbarkeit einiger in [DE95, Ber85, PPP00] vorgeschlagenen Reduktionen abgeschwächt werden. Folglich können die Reduktionen häufiger angewendet werden, was zu kleineren Netzsystemen führt. Das Kapitel wird abgerundet durch die Präsentation experimenteller Ergebnisse, die Aufschluß darüber geben, daß sich die hier betrachteten Reduktionstechniken in der Praxis gut bewähren.

5.1 LTL Model-Checking von S/T-Netzsystemen

Für das Verständnis des von Esparza und Heljanko beschriebenen Verfahrens zum Nachweis von LTL-X-Eigenschaften [EH00a, EH01, Hel02] ist es zunächst notwendig, Syntax und Semantik von LTL sowie das Konzept des Büchiautomaten einzuführen.

5.1.1 Linearzeit-Temporallogik (LTL)

Definition 5.1.1 beschreibt die Linearzeit-Temporallogik (LTL) nach Pnueli [Pnu77].

Definition 5.1.1 (Syntax von LTL)
Gegeben sei eine Menge Π von atomaren Propositionen. *LTL-Formeln* über Π sind wie folgt induktiv definiert:

- **wahr** beschreibt eine LTL-Formel.

- $\pi \in \Pi$ beschreibt eine LTL-Formel.

- Beschreiben φ und ψ LTL-Formeln, dann stellen auch

$$\neg\varphi, \quad \varphi \wedge \psi, \quad \mathcal{X}\varphi \quad \text{und} \quad \varphi \,\mathcal{U}\, \psi$$

  LTL-Formeln dar.

Dabei werden $\mathcal{X}$ und $\mathcal{U}$ im allgemeinen als *Next-* und *Until*-Operatoren bezeichnet. Darüberhinaus werden die folgenden Vereinfachungen verwendet:

$$\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi) \qquad \Diamond\varphi = \textbf{wahr}\ \mathcal{U}\ \varphi \qquad \Box\varphi = \neg\Diamond\neg\varphi$$

Der Operator $\Diamond\varphi$ (*eventually* (oder *finally*) φ) bedeutet, daß φ irgendwann an einem Punkt in der Zukunft gelten wird, wohingegen der Operator $\Box\varphi$ (*always* (oder *generally*) φ) aussagt, daß φ von nun an in jedem Punkt gilt. Das $\mathcal{X}$-freie Fragment von LTL wird als LTL-X bezeichnet.

Definition 5.1.2 (Semantik von LTL)
Gegeben sei eine LTL-Formel φ über der Menge Π von atomaren Propositionen. Desweiteren bezeichne $\Pi(\varphi)$ die Menge von atomaren Propositionen, welche in φ vorkommen. Die LTL-Formel φ definiert eine Sprache $L(\varphi)$ von ω-Wörtern über dem Alphabet $2^{\Pi(\varphi)}$, die φ erfüllen. Die Erfüllbarkeitsrelation $\models_{LTL}$ ist wie folgt induktiv über die Struktur von φ definiert (dabei bezeichne $\xi \models_{LTL} \varphi$, daß das ω-Wort ξ über dem Alphabet $2^{\Pi(\varphi)}$ die Formel φ erfüllt):

$$\begin{aligned}
&\xi \models_{LTL} \textbf{wahr} \\
&\xi \models_{LTL} \pi \in \Pi \quad &&\Leftrightarrow \quad \pi \in \xi(0) \\
&\xi \models_{LTL} \neg\varphi \quad &&\Leftrightarrow \quad \xi \not\models_{LTL} \varphi \\
&\xi \models_{LTL} \varphi \wedge \psi \quad &&\Leftrightarrow \quad \xi \models_{LTL} \varphi \text{ und } \xi \models_{LTL} \psi \\
&\xi \models_{LTL} \mathcal{X}\varphi \quad &&\Leftrightarrow \quad \xi^1 \models_{LTL} \varphi \\
&\xi \models_{LTL} \varphi \,\mathcal{U}\, \psi \quad &&\Leftrightarrow \quad \exists i\colon (\xi^i \models_{LTL} \psi) \wedge (\forall j < i\colon \xi^j \models_{LTL} \varphi)
\end{aligned}$$

5.1.2 LTL auf sicheren S/T-Netzsystemen

LTL kann auf sicheren S/T-Netzsystemen interpretiert werden, und zwar in einer *aktionsbasierten* oder *zustandsbasierten* Variante.
In der aktionsbasierten Variante wird für ein S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ die Menge Π der atomaren Propositionen mit der Transitionenmenge T assoziiert, d.h., für jede Transition $t \in T$ gibt es eine atomare Proposition $\pi_t \in \Pi$. Um zu entscheiden, ob eine unendliche Schaltfolge

$$\sigma = t_1 t_2 t_3 \ldots$$

von Σ die Formel φ erfüllt, wird diese zunächst in ein ω-Wort über dem Alphabet $2^{\Pi(\varphi)}$ transformiert. Dieses geschieht mittels einer Abbildung $\nu\colon T \to 2^{\Pi(\varphi)}$, die folgendermaßen definiert ist:

$$\nu(t) = \begin{cases} \{\pi_t\} & \text{falls } \pi_t \in \Pi(\varphi) \\ \emptyset & \text{sonst} \end{cases}$$

Die Abbildung ν ordnet also jedem Zeitpunkt der Schaltfolge eine Menge von Propositionen zu, die zu diesem Zeitpunkt gelten. Im allgemeinen kann ν auf ω-Wörter erweitert werden, d.h., $\nu\colon T^\omega \to (2^{\Pi(\varphi)})^\omega$ mit

$$\nu(\sigma) = \nu(t_1)\nu(t_2)\nu(t_3)\ldots$$

für eine unendliche Schaltfolge $\sigma = t_1 t_2 t_3 \ldots$. Damit gilt nun:

$$\sigma \models^{\nu}_{LTL} \varphi \Leftrightarrow \nu(\sigma) \in L(\varphi),$$

wobei $\sigma \models^{\nu}_{LTL} \varphi$ bezeichnet, daß σ die Formel φ unter der Abbildung ν erfüllt. Das S/T-Netzsystem Σ erfüllt genau dann die LTL-Formel φ (was mit $\Sigma \models^{\nu}_{LTL} \varphi$ bezeichnet wird), wenn jede unendliche Schaltfolge σ von Σ die Formel φ unter der Abbildung ν erfüllt.
Bei der zustandsbasierten Variante wird die Menge Π der atomaren Propositionen mit der Stellenmenge S assoziiert, d.h., für jede Stelle $s \in S$ gibt es eine atomare Proposition $\pi_s \in \Pi$.[2] Für eine unendliche Schaltfolge $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \ldots$ von Σ wird zunächst die unendliche Zustandsfolge

$$\xi = M_0 M_1 M_2 \ldots$$

in ein entsprechendes ω-Wort über dem Alphabet $2^{\Pi(\varphi)}$ überführt. Dieses geschieht mittels einer Abbildung $\nu\colon \{0,1\}^S \to 2^{\Pi(\varphi)}$, sodaß für eine Markierung $M \in \mathcal{R}(M_0)$ gilt:

$$\nu(M) = \{\pi_s \in \Pi(\varphi) \mid M(s) = 1\},$$

d.h., jedem Zustand (jeder Markierung) der Zustandsfolge ξ wird die Menge der atomaren Propositionen aus $\Pi(\varphi)$ zugewiesen, die in diesem Zustand gelten. Auch hier wird ν wieder auf ω-Wörter erweitert, d.h., $\nu\colon (\{0,1\}^S)^\omega \to (2^{\Pi(\varphi)})^\omega$ mit

$$\nu(\xi) = \nu(M_1)\nu(M_2)\nu(M_3)\ldots$$

für eine unendliche Zustandsfolge $\xi = M_1 M_2 M_3 \ldots$. Damit gilt nun:

$$\xi \models^{\nu}_{LTL} \varphi \Leftrightarrow \nu(\xi) \in L(\varphi),$$

wobei $\xi \models^{\nu}_{LTL} \varphi$ bezeichnet, daß ξ die Formel φ unter der Abbildung ν erfüllt. Das S/T-Netzsystem Σ erfüllt genau dann die LTL-Formel φ (was mit $\Sigma \models^{\nu}_{LTL} \varphi$ bezeichnet wird), wenn jede unendliche Zustandsfolge ξ von Σ die Formel φ unter der Abbildung ν erfüllt.
Im folgenden wird die zustandsbasierte Variante von LTL betrachtet, da sie sich für praktische Belange als interessanter erweist.

[2] Alternativ dazu werden im weiteren Verlauf der Arbeit auch die Stellennamen als atomare Propositionen verwendet.

5.1.3 Büchiautomaten und -netze

Büchiautomaten spielen eine zentrale Rolle bei der Verifikation von LTL-Eigenschaften. Sie „arbeiten“ ganz ähnlich wie herkömmliche endliche Automaten, jedoch werden die Akzeptanzbedingungen für Büchiautomaten über ω-Wörtern definiert.

Definition 5.1.3 (Büchiautomat [Büc62])
Ein *Büchiautomat* wird definiert als ein 5-Tupel $\mathcal{A} = (Q, \Gamma, \delta, q_0, Q_E)$, wobei

- Q eine endliche Menge von Zuständen,
- Γ ein endliches Alphabet,
- $\delta \subseteq Q \times \Gamma \times Q$ eine Übergangsrelation,
- $q_0 \in Q$ den Anfangszustand und
- $Q_E \subseteq Q$ eine Menge von Endzuständen

beschreiben.
Der Büchiautomat $\mathcal{A}$ *akzeptiert* ein unendliches Wort $a_1 a_2 a_3 \ldots \in \Gamma^\omega$, falls in $\mathcal{A}$ ein unendlicher Ablauf $\sigma = q_0 a_1 q_1 a_2 q_2 a_3 q_3 \ldots$ mit $(q_i, a_{i+1}, q_{i+1}) \in \delta$ für jedes $i \geq 0$ existiert, sodaß unendlich viele Zustände aus Q_E (Endzustände) in σ vorkommen. Anstatt $(q_i, a_{i+1}, q_{i+1}) \in \delta$ wird auch die Notation $q_i \xrightarrow{a_{i+1}} q_{i+1}$ verwendet.
Die Sprache $L(\mathcal{A})$ enthält alle unendlichen Wörter aus Γ^ω, die von $\mathcal{A}$ akzeptiert werden. Die Sprache $L(\mathcal{A}, q)$ für $q \in Q$ enthält alle unendlichen Wörter aus Γ^ω, die $\mathcal{A}$ ausgehend von q akzeptiert.

Satz 5.1.1 beschreibt einen Zusammenhang zwischen Büchiautomaten und LTL-Formeln, der für die Verifikation linearzeit-temporallogischer Aussagen von großer Bedeutung ist.

Satz 5.1.1 (Büchiautomaten und LTL [VW86b, GPVW96])
Gegeben sei eine LTL-Formel φ über einer Menge Π von atomaren Propositionen. Dann gibt es einen Büchiautomaten $\mathcal{A} = (Q, 2^{\Pi(\varphi)}, \delta, q_0, Q_E)$ der Größe $\mathcal{O}(2^{|\varphi|})$ mit $L(\mathcal{A}) = L(\varphi)$.

Für die Übertragung des automatentheoretischen Ansatzes zur Verifikation von LTL-Eigenschaften auf die Petrinetzebene ist es notwendig, das Konzept der Büchiautomaten in Form sogenannter *Büchinetze* in die Petrinetztheorie zu übernehmen.

Definition 5.1.4 (S/T-Büchinetz)
Ein 4-Tupel $N_\mathcal{B} = (S, T, F, S_E)$ wird als *S/T-Büchinetz* bezeichnet, falls

- (S, T, F) ein S/T-Netz und
- $S_E \subseteq S$ eine Menge von Endstellen

beschreiben.
Ein Tupel $\Sigma_{\mathcal{B}} = (N_{\mathcal{B}}, M_0)$ wird *S/T-Büchinetzsystem mit der Anfangsmarkierung* M_0 genannt, falls $N_{\mathcal{B}}$ ein S/T-Büchinetz und M_0 eine Markierung von $N_{\mathcal{B}}$ darstellen.
Ein S/T-Büchinetzsystem $\Sigma_{\mathcal{B}}$ *akzeptiert* eine unendliche Schaltfolge $\sigma = t_1 t_2 t_3 \ldots$, falls es in $\Sigma_{\mathcal{B}}$ einen unendlichen Ablauf

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_3} M_3 \ldots$$

gibt, der unendlich viele Markierungen aufweist, die Stellen aus S_E (Endstellen) markieren. Die Sprache $L(\Sigma_{\mathcal{B}})$ enthält alle unendlichen Schaltfolgen, die von $\Sigma_{\mathcal{B}}$ akzeptiert werden.

5.1.4 Der Ansatz von Esparza und Heljanko

Die Reduktionstechniken, welche in den nachfolgenden Abschnitten noch behandelt werden, wurden gezielt auf das Verfahren zur Verifikation von LTL-X-Eigenschaften nach Esparza und Heljanko [EH00a, EH01, Hel02] abgestimmt. Deshalb wird im folgenden auf diese Methode näher eingegangen.
Esparzas und Heljankos Ansatz basiert auf dem automatentheoretischen Verfahren zum Nachweis linearzeit-temporallogischer Eigenschaften von Vardi und Wolper [VW86a, Var96], der jedoch auf die Petrinetzebene übertragen wurde. Dabei wird ein Netzsystem konstruiert, das sich aus dem Produkt des Systemnetzes und des Büchiautomaten ergibt, der genau diejenigen Abläufe akzeptiert, welche die zu verifizierende Eigenschaft verletzen. Das Model-Checking Problem wird dann auf die beiden Probleme reduziert, sogenannte *illegale ω-Traces* und *illegale Livelocks* in dem Produktnetzsystem zu finden. Beide Probleme werden unter Zuhilfenahme des Produktnetzpräfixes gelöst. Der Hauptvorteil dieses Ansatzes im Vergleich zu Wallners [Wal98] liegt in dessen Einfachheit. Wallner erzeugt zunächst ein endliches, vollständiges Präfix des Produktnetzsystems und anschließend einen Graphen, dessen Definition jedoch nicht trivial ist und der exponentiell in der Größe des Präfixes wachsen kann. Der Ansatz von Esparza und Heljanko vermeidet zwar die Konstruktion eines solchen Graphen, kann jedoch zur Erzeugung eines größeren Präfixes führen.
Im folgenden werden nun die wichtigsten Definitionen und Ergebnisse aus [EH00a, EH01, Hel02] vorgestellt. Während die Konstruktion des Produktnetzsystems in [EH01, Hel02] eher informell beschrieben wurde, soll in dieser Arbeit eine rein formale Beschreibung des Produktnetzsystems angegeben werden, die dann später anhand eines Beispiels näher erläutert wird. Darüberhinaus werden zusätzlich zwei Teilnetze des Produktnetzes definiert, die später bei den Korrektheitsbeweisen der Reduktionsregeln Anwendung finden.
Zur Vereinfachung der Synchronisation von Netzsystem und Büchiautomat wird dieser zunächst in eine geeignetere Darstellung überführt. Dabei kann ein Element aus $2^{\Pi(\varphi)}$, mit dem eine Kante des Automats beschriftet ist, als Konjunktion aller Propositionen aus $\Pi(\varphi)$ verstanden werden. Eine solche Interpretation der Kantenbeschriftungen kann mittels der Abbildung ζ aus Definition 5.1.5 auf der nächsten Seite vorgenommen werden.

Definition 5.1.5 (Abbildung ζ)
Gegeben seien eine LTL-X-Formel φ und die Menge $\mathcal{B}_{exp}(\Pi(\varphi))$ von booleschen Ausdrücken über den atomaren Propositionen von φ. Die Abbildung $\zeta\colon 2^{\Pi(\varphi)} \to \mathcal{B}_{exp}(\Pi(\varphi))|_{\{\wedge\}}$ ist definiert als:

$$\zeta(x) = \bigwedge_{\pi_i \in x} \pi_i \bigwedge_{\pi_i \in \Pi(\varphi)\setminus x} \neg\pi_i$$

Der Einfachheit halber wird $\zeta(x)$ im folgenden mit der Menge

$$\zeta(x) = \{\pi_i \mid \pi_i \in x\} \cup \{\neg\pi_i \mid \pi_i \in \Pi(\varphi) \setminus x\}$$

assoziiert.

Definition 5.1.6 (Produktnetzsystem $\Sigma_{S\times\neg\varphi}$)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma_S = (N_S, M_{0_S})$ mit $N_S = (S_S, T_S, F_S)$ und ein Büchiautomat $\mathcal{A}_{\neg\varphi} = (Q, 2^{\Pi(\varphi)}, \delta, q_0, Q_E)$ mit $L(\mathcal{A}_{\neg\varphi}) = L(\neg\varphi)$. Das Tupel $\Sigma_{S\times\neg\varphi} = (N_\mathcal{B}, M_0)$ heißt *Produktnetzsystem*, falls $N_\mathcal{B} = (S, T, F, S_E)$ ein S/T-Büchinetz beschreibt, wobei

- $S = S_V \cup S_H \cup S_B \cup \{s_s, s_f\}$, wobei

  $$\begin{aligned} S_V &= \{s \in S_S \mid s \in \Pi(\varphi)\} \cup \{\bar{s} \mid \exists (q, x, q') \in \delta\colon \neg s \in \zeta(x)\} \\ S_H &= S_S \setminus S_V \\ S_B &= \{s_q \mid q \in Q\} \end{aligned}$$

  S_V und S_H bezeichnen die Mengen der *sichtbaren* („*visible*“) bzw. *unsichtbaren* („*hidden*“) *Stellen*. $\bar{s}$ bezeichnet das Komplement einer Stelle s, siehe hierzu auch Abschnitt 3.3.2 auf Seite 58. S_B beschreibt die Menge der *Büchistellen* und s_s, s_f modellieren einen Scheduler, der gewährleistet, daß der Büchiautomat die einzelnen Systemschritte überwachen kann.
  Die Menge $S_E \subseteq S_B$ von *Endstellen* ist durch

  $$S_E = \{s_q \mid q \in Q_E\}$$

  gegeben.

- $T = T_V \cup T_H \cup T_B \cup T_L$, wobei

  $$\begin{aligned} T_V &= \{t \in T_S \mid \exists s \in S_V\colon s \in {}^\bullet t \cup t^\bullet\} \\ T_H &= T_S \setminus T_V \\ T_B &= \{t_{(q,\zeta(x),q')} \mid (q, x, q') \in \delta\} \\ T_L &= \{t_{(q,M)} \mid q \in Q \wedge M \in \mathcal{R}(M_0)|_{\Sigma_{S\times\neg\varphi}\setminus T_L} \wedge M(s_q) = 1 \wedge M(s_f) = 1 \wedge \\ &\qquad (\nu(M))^\omega \in L(\mathcal{A}_{\neg\varphi}, q)\} \end{aligned}$$

  T_V und T_H bezeichnen die Mengen der *sichtbaren* bzw. *unsichtbaren Transitionen*. T_B beschreibt die Menge der *Büchitransitionen*. Dabei bedeutet $t_{(q,\zeta(x),q')}$, daß die

Transition eine Marke von der Stelle s_q abzieht und auf die Stelle $s_{q'}$ legt, falls die Stellen aus $\zeta(x)$ markiert sind. Die Menge T_L enthält sogenannte *Livelock-Wächter*[3], auf die im weiteren Verlauf noch näher eingegangen wird.
Ferner wird eine Menge $T_I \subseteq T_B$ von *unendlichen Trace-Wächtern* definiert, die alle Büchitransitionen enthält, deren Schalten eine Marke auf eine Endstelle legt, d.h.,

$$T_I = \{t_{(q,\zeta(x),q')} \mid q' \in Q_E\}$$

- $F = F_S \cup F_{\bar{V}} \cup F_B \cup F_V \cup F_{s_s} \cup F_{s_f} \cup F_L$, wobei

$$\begin{aligned}
F_{\bar{V}} &= \{(\bar{s}, t) \mid \bar{s} \in S_V \wedge (t,s) \in F_S \wedge (s,t) \notin F_S\} \cup \\
&\quad \{(t, \bar{s}) \mid \bar{s} \in S_V \wedge (s,t) \in F_S \wedge (t,s) \notin F_S\} \\
F_B &= \{(s_q, t_{(q,\zeta(x),q')}) \mid s_q \in S_B \wedge t_{(q,\zeta(x),q')} \in T_B\} \cup \\
&\quad \{(t_{(q,\zeta(x),q')}, s_{q'}) \mid s_{q'} \in S_B \wedge t_{(q,\zeta(x),q')} \in T_B\} \\
F_V &= \{(s, t_{(q,\zeta(x),q')}) \mid s \in S_V \wedge t_{(q,\zeta(x),q')} \in T_B \wedge s \in \zeta(x)\} \cup \\
&\quad \{(t_{(q,\zeta(x),q')}, s) \mid s \in S_V \wedge t_{(q,\zeta(x),q')} \in T_B \wedge s \in \zeta(x)\} \cup \\
&\quad \{(\bar{s}, t_{(q,\zeta(x),q')}) \mid \bar{s} \in S_V \wedge t_{(q,\zeta(x),q')} \in T_B \wedge \neg s \in \zeta(x)\} \cup \\
&\quad \{(t_{(q,\zeta(x),q')}, \bar{s}) \mid \bar{s} \in S_V \wedge t_{(q,\zeta(x),q')} \in T_B \wedge \neg s \in \zeta(x)\} \\
F_{s_s} &= \{(s_s, t) \mid t \in T_V\} \cup \{(t_{(q,\zeta(x),q')}, s_s) \mid t_{(q,\zeta(x),q')} \in T_B\} \\
F_{s_f} &= \{(s_f, t_{(q,\zeta(x),q')}) \mid t_{(q,\zeta(x),q')} \in T_B\} \cup \{(t, s_f) \mid t \in T_V\} \\
F_L &= \{(s, t_{(q,M)}) \mid s \in S \wedge t_{(q,M)} \in T_L \wedge M(s) = 1\} \cup \\
&\quad \{(t_{(q,M)}, s) \mid s \in (S_V \cup S_H) \wedge t_{(q,M)} \in T_L \wedge M(s) = 1\}
\end{aligned}$$

Mittels $F_{\bar{V}}$ werden die notwendigen Komplemente in das Netzsystem eingebunden. Durch F_B wird die Struktur des Teilnetzes festgelegt, das den Büchiautomaten repräsentiert. Die Synchronisation der sichtbaren Stellen mit den Büchitransitionen geschieht durch F_V. Mittels F_{s_s} und F_{s_f} werden die Schedulerstellen in das Netzsystem integriert. Zuletzt werden durch F_L noch die Livelock-Wächter mit dem Netzsystem verbunden.

- Die Anfangsmarkierung M_0 ist durch

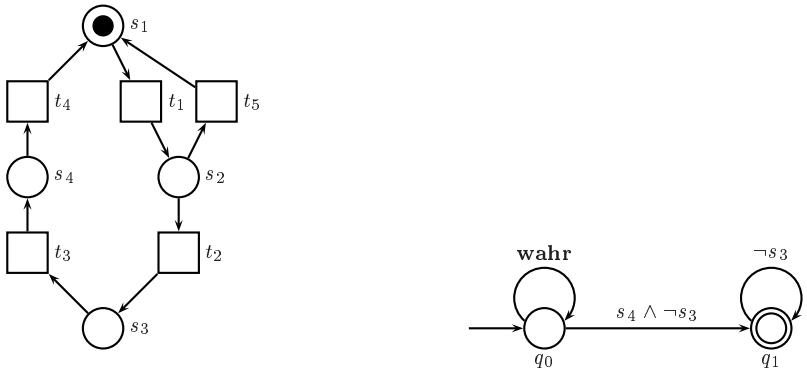
$$M_0(s) = \begin{cases} 1 & \text{falls } M_{0_S}(s) = 1 \vee (s = \overline{s'} \wedge M_{0_S}(s') = 0) \vee s = s_{q_0} \vee s = s_f \\ 0 & \text{sonst} \end{cases}$$

gegeben. Die Stelle s_f muß anfänglich markiert sein, damit das Büchinetz die Anfangsmarkierung des Netzsystems überwachen kann.

Proposition 5.1.1 auf der nächsten Seite ergibt sich als direkte Folgerung aus der Definition 5.1.6 auf der vorherigen Seite von $\Sigma_{S \times \neg\varphi}$.

[3]Da es für jeden Zustand $q \in Q$ exponentiell in der Größe von Σ_S viele Transitionen $t_{(q,M)}$ geben kann, werden diese nicht explizit, sondern „On-the-Fly“ während der Präfixbildung erzeugt.

Abbildung 5.1 *Sicheres S/T-Netzsystem (a) und ein Büchiautomat, der die Formel* $\neg\varphi = \neg\Box(s_4 \Rightarrow \Diamond s_3)$ *akzeptiert (b)*



(a) (b)

Proposition 5.1.1 (Büchitransitionen und sichtbare Transitionen [ES01a])
Gegeben sei ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$*. Es gilt:*

$$\forall t \in T_V \cup T_B : |{}^\bullet t| \geq 2 \wedge |t^\bullet| \geq 2$$
$$\forall t \in T_L : t^\bullet \subset {}^\bullet t$$

Definition 5.1.7 (Illegale ω-Traces und illegale Livelocks [EH01, Hel02])
Gegeben sei ein Produktnetzsystem $\Sigma_{S\times\neg\varphi} = (N_\mathcal{B}, M_0)$ mit $N_\mathcal{B} = (S, T, F, S_E)$.

- Eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$ heißt *illegaler ω-Trace* von $\Sigma_{S\times\neg\varphi}$, falls σ unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) enthält.
- Eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma t} M \xrightarrow{\sigma_1}$ heißt *illegaler Livelock* von $\Sigma_{S\times\neg\varphi}$, falls $t \in T_L$ und $\sigma_1 \in T_H^\omega$.

Mittels der soeben eingeführten Begriffe und Definitionen kann nun das Hauptergebnis aus [EH01, Hel02] in Form von Satz 5.1.2 präsentiert werden.

Satz 5.1.2 (LTL-X Model-Checking [EH01, Hel02])
Gegeben seien ein sicheres S/T-Netzsystem Σ_S *und eine LTL-X-Formel* φ*.* Σ_S *erfüllt genau dann* φ*, wenn in* $\Sigma_{S\times\neg\varphi}$ *weder illegale* ω*-Traces noch illegale Livelocks vorkommen.*

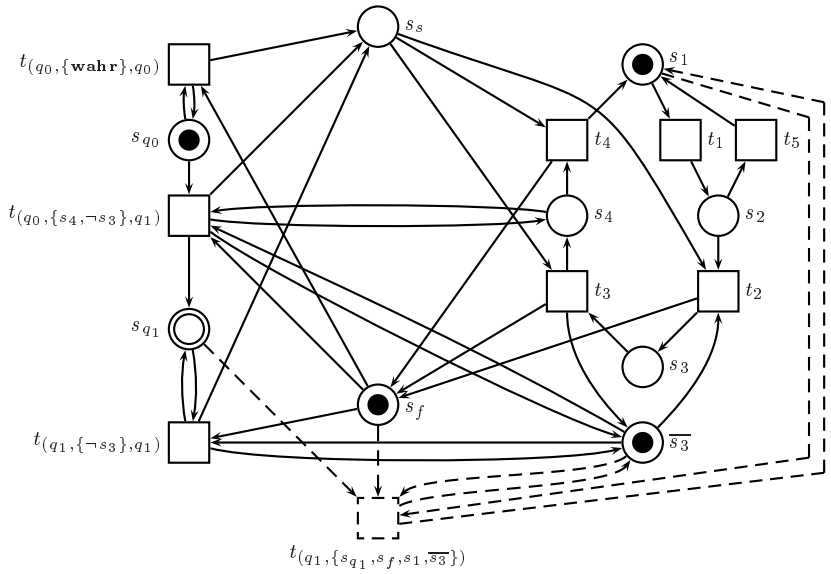
Satz 5.1.2 und die vorherigen Definitionen sollen nun anhand des Netzsystems aus Abbildung 5.1 (a) auf dieser Seite näher erläutert werden. Es soll gezeigt werden, daß das Netzsystem die LTL-X-Eigenschaft

$$\varphi = \Box(s_4 \Rightarrow \Diamond s_3)$$

(„*immer, wenn die Stelle* s_4 *markiert ist, wird irgendwann auch die Stelle* s_3 *markiert sein*“) nicht erfüllt, was durch Ausführung der unendlichen Schaltfolge

$$t_1 t_2 t_3 t_4 (t_1 t_5)^\omega$$

Abbildung 5.2 *Produktnetzsystem für das Netzsystem und den Büchiautomaten aus Abbildung 5.1 auf der vorherigen Seite*



leicht nachgewiesen werden kann. Dazu muß zunächst ein Büchiautomat $\mathcal{A}_{\neg\varphi}$ erzeugt werden, der genau diejenigen Abläufe akzeptiert, welche φ verletzen, d.h., $L(\mathcal{A}_{\neg\varphi}) = L(\neg\varphi)$. Ein solcher Büchiautomat ist in Abbildung 5.1 (b) auf der vorherigen Seite dargestellt, wobei anzumerken ist, daß die Kanten mit den booleschen Ausdrücken nach Anwendung der Abbildung ζ beschriftet sind und der Automat anschließend noch optimiert wurde.[4] Abbildung 5.2 zeigt das Produktnetzsystem $\Sigma_{S\times\neg\varphi}$. Die Stellen $s_3, \overline{s_3}, s_4$ und die Transitionen t_2, t_3, t_4 werden als sichtbar bezeichnet, wohingegen die Stellen s_1, s_2 sowie die Transitionen t_1, t_5 als unsichtbar gelten. Die Stelle $s_{q_1} \in S_E$ stellt die einzige Endstelle des Büchinetzes dar. Die Menge T_I der unendlichen Trace-Wächter ergibt sich zu

$$T_I = \{t_{(q_0,\{s_4,\neg s_3\},q_1)}, t_{(q_1,\{\neg s_3\},q_1)}\},$$

und die Menge T_L von Livelock-Wächtern, die erst während der Präfixbildung erzeugt werden, wird lediglich durch die gestrichelte Transition

$$t_{(q_1,\{s_{q_1},s_f,s_1,\overline{s_3}\})}$$

angedeutet, da diese für das Verständnis des Beispiels von Bedeutung ist. Die Synchronisation zwischen dem Ausgangsnetzsystem und dem den Büchiautomaten repräsentierenden Büchinetzsystem geschieht lediglich über die sichtbaren Stellen und Transitionen. Dabei garantieren die beiden Schedulerstellen s_s (Schalten einer sichtbaren Transition)

[4] Es existieren viele Tools zur automatischen Generierung und Optimierung von Büchiautomaten zu einer gegebenen LTL-Formel, siehe beispielsweise [GPVW96, Hol03, GO01, DGV99, EH00b].

und s_f (Schalten einer Büchitransition), daß das Büchinetz jede sichtbare Aktion des Ausgangsnetzes beobachten kann, indem zwischen dem Schalten zweier sichtbarer Transitionen mindestens eine Büchitransition schalten muß, und umgekehrt.
Um nun zu überprüfen, ob ein System Σ_S eine LTL-X-Eigenschaft φ erfüllt, muß das Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ nach Satz 5.1.2 auf Seite 122 auf illegale ω-Traces und illegale Livelocks untersucht werden. Besäße $\Sigma_{S\times\neg\varphi}$ einen illegalen ω-Trace, dann müßte es eine unendliche Schaltfolge σ geben, in der die T_I-Transitionen

$$t_{(q_0,\{s_4,\neg s_3\},q_1)} \quad \text{und/oder} \quad t_{(q_1,\{\neg s_3\},q_1)}$$

unendlich oft vorkommen. Dazu müßte die Stelle s_f unendlich oft eine Marke erhalten, was jedoch nur dadurch erreicht werden könnte, daß mindestens eine der sichtbaren Transitionen t_2, t_3 oder t_4 unendlich oft schaltet. Alle möglichen Schaltfolgen σ, in der immer wieder sichtbare Transitionen vorkommen, beispielsweise

$$\sigma = t_1 t_{(q_0,\{\mathbf{wahr}\},q_0)} t_2 t_{(q_0,\{\mathbf{wahr}\},q_0)} t_3 t_{(q_0,\{s_4,\neg s_3\},q_1)} t_4 t_{(q_1,\{\neg s_3\},q_1)} t_1 t_2,$$

führen jedoch zu der Verklemmung

$$M = \{s_{q_1}, s_f, s_3\}.$$

Daraus folgt unmittelbar, daß es in $\Sigma_{S\times\neg\varphi}$ keinen illegalen ω-Trace gibt. Nun bleibt zu prüfen, ob $\Sigma_{S\times\neg\varphi}$ einen illegalen Livelock enthält, d.h., eine unendliche Schaltfolge σ, in der ein Livelock-Wächter (eine Transition aus T_L) vorkommt, welchem dann ausschließlich nur noch unsichtbare Transitionen folgen. Zunächst soll jedoch gezeigt werden, daß die Transition

$$t_{(q_1,\{s_{q_1},s_f,s_1,\overline{s_3}\})}$$

die Kriterien eines Livelock-Wächters erfüllt. Durch das Ausführen der Schaltfolge

$$\sigma = t_1 t_{(q_0,\{\mathbf{wahr}\},q_0)} t_2 t_{(q_0,\{\mathbf{wahr}\},q_0)} t_3 t_{(q_0,\{s_4,\neg s_3\},q_1)} t_4$$

erhält man den Systemablauf

$$M_0 = \{s_{q_0}, s_f, s_1, \overline{s_3}\} \xrightarrow{\sigma} \{s_{q_1}, s_f, s_1, \overline{s_3}\} = M,$$

wobei die Markierung M von M_0 aus ohne das Schalten eines Livelock-Wächters erreichbar ist und $M(s_f) = 1$ gilt. Desweiteren gilt

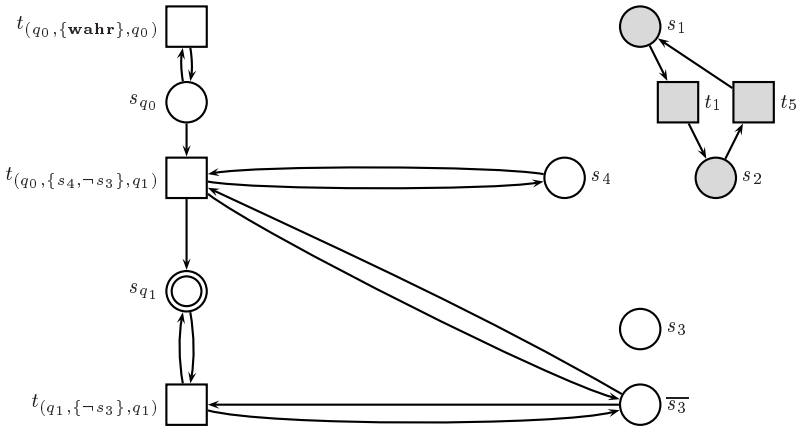
$$(\nu(M))^\omega = \emptyset^\omega \quad \text{und} \quad \emptyset^\omega \in L(\mathcal{A}_{\neg\varphi}, q_1),$$

und somit sind alle Kriterien erfüllt, die ein Livelock-Wächter aufweisen muß. Es ist offensichtlich, daß die unendliche Schaltfolge

$$\sigma = t_1 t_{(q_0,\{\mathbf{wahr}\},q_0)} t_2 t_{(q_0,\{\mathbf{wahr}\},q_0)} t_3 t_{(q_0,\{s_4,\neg s_3\},q_1)} t_4 t_{(q_1,\{s_{q_1},s_f,s_1,\overline{s_3}\})} (t_1 t_5)^\omega$$

einen illegalen Livelock darstellt, woraus unmittelbar folgt, daß das Netzsystem Σ_S die LTL-X-Eigenschaft φ nicht erfüllt.

Abbildung 5.3 *Teilnetze $N_{\neg\varphi}$ (weiß) und N_h (grau) des Produktnetzsystems aus Abbildung 5.2 auf Seite 123*



Um die Korrektheitsbeweise der in dieser Arbeit vorgeschlagenen Netzreduktionen zu vereinfachen, wird das Ergebnis von Satz 5.1.2 auf Seite 122 in einfacher nachprüfbare Systemeigenschaften übersetzt, und bei den anschließenden Reduktionen muß dann lediglich auf die Erhaltung dieser Eigenschaften geachtet werden. Dazu werden zunächst die beiden Teilnetze $N_{\neg\varphi}$ und N_h des Produktnetzsystems $\Sigma_{S\times\neg\varphi}$ definiert.

Definition 5.1.8 (Teilnetze $N_{\neg\varphi}$ und N_h)
Gegeben sei ein Produktnetzsystem $\Sigma_{S\times\neg\varphi} = (N_{\mathcal{B}}, M_0)$ mit $N_{\mathcal{B}} = (S, T, F, S_E)$ nach Definition 5.1.6 auf Seite 120.

- $N_{\neg\varphi} = (S_{\neg\varphi}, T_{\neg\varphi}, F_{\neg\varphi}, S_{\neg\varphi_E})$ beschreibt ein Büchinetz, wobei

$$\begin{aligned} S_{\neg\varphi} &= S_V \cup S_B \\ T_{\neg\varphi} &= T_B \\ F_{\neg\varphi} &= F \cap ((S_{\neg\varphi} \times T_{\neg\varphi}) \cup (T_{\neg\varphi} \times S_{\neg\varphi})) \\ S_{\neg\varphi_E} &= S_E \end{aligned}$$

- $N_h = (S_h, T_h, F_h)$ beschreibt ein Netz, wobei

$$\begin{aligned} S_h &= S_H \\ T_h &= T_H \\ F_h &= F \cap ((S_h \times T_h) \cup (T_h \times S_h)) \end{aligned}$$

Abbildung 5.3 zeigt die beiden Teilnetze $N_{\neg\varphi}$ (weiß) und N_h (grau) für das Produknetzsystem $\Sigma_{S\times\neg\varphi}$ aus Abbildung 5.2 auf Seite 123. Dabei repräsentiert das Büchinetz $N_{\neg\varphi}$ den Büchiautomaten $\mathcal{A}_{\neg\varphi}$, und N_h beschreibt das Teilnetz, in dem ausschließlich unsichtbare Transitionen schalten können.

Satz 5.1.3 (LTL-X Model-Checking [ES01a])
Gegeben seien ein sicheres S/T-Netzsystem Σ_S und eine LTL-X-Formel φ. Σ_S verletzt genau dann φ, wenn mindestens eine der folgenden Bedingungen in dem Produktnetzsystem $\Sigma_{S\times\neg\varphi} = (N_{\mathcal{B}}, M_0)$ gilt:

(i) Es existiert eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen, oder

(ii) es existiert eine Schaltfolge $M_0 \xrightarrow{\sigma} M$, sodaß

(a) $L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})) \neq \emptyset$ und
(b) $L^\omega((N_h, M|_{S_h})) \neq \emptyset$.

Beweis: Mit dem Ergebnis von Satz 5.1.2 auf Seite 122 genügt es, die folgenden Behauptungen zu beweisen:

(1) $\Sigma_{S\times\neg\varphi}$ besitzt einen illegalen ω-Trace $\Leftrightarrow$ Bedingung (i) von Satz 5.1.3 gilt in $\Sigma_{S\times\neg\varphi}$.

($\Rightarrow$) Angenommen, $\Sigma_{S\times\neg\varphi}$ besitzt einen illegalen ω-Trace. Aus Definition 5.1.7 auf Seite 122 folgt die Existenz einer unendlichen Schaltfolge $M_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter vorkommen. Dieses entspricht Bedingung (i) von Satz 5.1.3.

($\Leftarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es gibt eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter vorkommen. Dieses entspricht aber genau Definition 5.1.7 auf Seite 122 eines illegalen ω-Traces.

(2) $\Sigma_{S\times\neg\varphi}$ besitzt einen illegalen Livelock $\Leftrightarrow$ Bedingung (ii) von Satz 5.1.3 gilt in $\Sigma_{S\times\neg\varphi}$.

($\Rightarrow$) Angenommen, $\Sigma_{S\times\neg\varphi}$ besitzt einen illegalen Livelock. Aus Definition 5.1.7 auf Seite 122 folgt die Existenz einer unendlichen Schaltfolge

$$M_0 \xrightarrow{\sigma t} M \xrightarrow{\sigma_1},$$

wobei $t \in T_L$ und $\sigma_1 \in T_H^\omega$. Aus $t \in T_L$ und der Definition 5.1.6 auf Seite 120 von Livelock-Wächtern folgt die Existenz einer unendlichen Schaltfolge

$$M_0 \xrightarrow{\sigma} M_1 \xrightarrow{t} M \xrightarrow{\sigma_1},$$

wobei $M_1(s_f) = 1$ und weiterhin gilt:

$$\exists q \in Q\colon M_1(s_q) = 1 \wedge (\nu(M_1))^\omega \in L(\mathcal{A}_{\neg\varphi}, q)$$

Daraus folgt unmittelbar die Existenz von Zuständen $q_1, q_2, q_3, \ldots \in Q$, sodaß

$$q \xrightarrow{\nu(M_1)} q_1 \xrightarrow{\nu(M_1)} q_2 \xrightarrow{\nu(M_1)} q_3 \ldots$$

einen unendlichen Ablauf von $\mathcal{A}_{\neg\varphi}$ darstellt, in dem unendlich viele Zustände aus Q_E vorkommen. Zunächst soll lediglich das Anfangsstück

$$q \xrightarrow{\nu(M_1)} q_1$$

des Ablaufs betrachtet werden. Aus $(q, \nu(M_1), q_1) \in \delta$ und Definition 5.1.8 auf Seite 125 folgt, daß in $N_{\neg\varphi}$ eine Transition

$$t_{(q,\zeta(\nu(M_1)),q_1)} \in T_{\neg\varphi}$$

existiert, sodaß

$$\begin{aligned} {}^\bullet t_{(q,\zeta(\nu(M_1)),q_1)} = {} & \{s_q\} \cup \\ & \{s \mid M_1(s) = 1 \wedge s \in \Pi(\varphi)\} \cup \\ & \{\bar{s} \mid M_1(s) = 0 \wedge s \in \Pi(\varphi)\} \end{aligned}$$

und

$$\begin{aligned} t^\bullet_{(q,\zeta(\nu(M_1)),q_1)} = {} & \{s_{q_1}\} \cup \\ & \{s \mid M_1(s) = 1 \wedge s \in \Pi(\varphi)\} \cup \\ & \{\bar{s} \mid M_1(s) = 0 \wedge s \in \Pi(\varphi)\}. \end{aligned}$$

Daraus folgt unmittelbar die Existenz einer erreichbaren Markierung $M_2|_{S_{\neg\varphi}}$, sodaß

$$M_1|_{S_{\neg\varphi}} \xrightarrow{t_{(q,\zeta(\nu(M_1)),q_1)}} M_2|_{S_{\neg\varphi}},$$

wobei

$$M_2|_{S_{\neg\varphi}}(s) = \begin{cases} 0 & \text{falls } s = s_q \wedge s_q \neq s_{q_1} \\ 1 & \text{falls } s = s_{q_1} \\ M_1|_{S_{\neg\varphi}}(s) & \text{sonst} \end{cases}$$

Mit der gleichen Argumentationskette wie eben folgt die Existenz einer erreichbaren Markierung $M_3|_{S_{\neg\varphi}}$, sodaß

$$M_2|_{S_{\neg\varphi}} \xrightarrow{t_{(q_1,\zeta(\nu(M_1)),q_2)}} M_3|_{S_{\neg\varphi}}.$$

Schließlich liefert beliebiges Wiederholen obiger Argumentationskette das Ergebnis

$$t_{(q,\zeta(\nu(M_1)),q_1)} t_{(q_1,\zeta(\nu(M_1)),q_2)} t_{(q_2,\zeta(\nu(M_1)),q_3)} \ldots \in L((N_{\neg\varphi}, M_1|_{S_{\neg\varphi}})).$$

Das bedeutet, daß es in $\Sigma_{S\times\neg\varphi}$ die Schaltfolge

$$M_0 \xrightarrow{\sigma} M_1 \quad \text{mit} \quad L((N_{\neg\varphi}, M_1|_{S_{\neg\varphi}})) \neq \emptyset$$

gibt, welches Bedingung (ii)(a) von Satz 5.1.3 auf Seite 126 entspricht. Mit der Beziehung

$$\forall s \in S_H \colon M_1(s) = M(s)$$

folgt aus $M \xrightarrow{\sigma_1}$ und $\sigma_1 \in T_H^\omega$ unmittelbar

$$\sigma_1 \in L^\omega((N_h, M_1|_{S_h})),$$

womit auch Bedingung (ii)(b) von Satz 5.1.3 auf Seite 126 erfüllt ist.

($\Leftarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$. Dann existiert eine Schaltfolge $M_0 \xrightarrow{\sigma} M$, sodaß

$$L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})) \neq \emptyset \quad \text{und} \quad L^\omega((N_h, M|_{S_h})) \neq \emptyset.$$

Ohne Beschränkung der Allgemeinheit kann

$$M(s_f) = 1$$

angenommen werden.
(Andernfalls gibt es erreichbare Markierungen $M_1, M_2 \in \mathcal{R}(M_0)$, sodaß eine Zerlegung von σ in

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t_{(q_i,x,q_j)}} M_2 \xrightarrow{\sigma_2} M$$

mit $t_{(q_i,x,q_j)} \in T_B$ und $\sigma_2 \in T_H^*$ gewählt werden kann. Aus $s_f \in {}^\bullet t_{(q_i,x,q_j)}$ folgt

$$M_1(s_f) = 1.$$

Wegen $\sigma_2 \in T_H^*$ gilt

$$\forall s \in S_{\neg\varphi} \colon M_2(s) = M(s)$$

und somit

$$t_{(q_i,x,q_j)}\sigma_3 \in L((N_{\neg\varphi}, M_1|_{S_{\neg\varphi}})) \quad \text{für} \quad \sigma_3 \in L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})).$$

Wegen $t_{(q_i,x,q_j)} \in T_B$ gilt

$$\forall s \in S_h \colon M_1(s) = M_2(s)$$

und somit

$$\sigma_2\sigma_4 \in L^\omega((N_h, M_1|_{S_h})) \quad \text{für} \quad \sigma_4 \in L^\omega((N_h, M|_{S_h})).$$

)
Aus $L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})) \neq \emptyset$ folgt die Existenz einer unendlichen Schaltfolge

$$t_{(q,x_1,q_1)}t_{(q_1,x_2,q_2)}t_{(q_2,x_3,q_3)}\ldots,$$

sodaß

$$M|_{S_{\neg\varphi}} \xrightarrow{t_{(q,x_1,q_1)}} M_1|_{S_{\neg\varphi}} \xrightarrow{t_{(q_1,x_2,q_2)}} M_2|_{S_{\neg\varphi}} \xrightarrow{t_{(q_2,x_3,q_3)}} M_3|_{S_{\neg\varphi}} \ldots,$$

wobei unendlich viele Markierungen vorkommen, die Stellen aus $S_{\neg\varphi_E}$ markieren. Aufgrund der Konstruktionsweise von $N_{\neg\varphi}$ ist sichergestellt, daß ein Schalten der Transitionen keine Änderung der Markenanzahl von Stellen aus S_V hervorruft. Daher gilt

$$\forall s \in S_V \colon M|_{S_{\neg\varphi}}(s) = M_1|_{S_{\neg\varphi}}(s) = M_2|_{S_{\neg\varphi}}(s) = M_3|_{S_{\neg\varphi}}(s) = \ldots$$

Daraus kann mit

$$\forall i \; \forall s \in S_V \colon s \in x_i \vee \bar{s} \in x_i \tag{5.1.1}$$

unmittelbar

$$x_1 = x_2 = x_3 = \ldots$$

abgeleitet werden. Aus Definition 5.1.8 auf Seite 125 von $N_{\neg\varphi}$ und Definition 5.1.6 auf Seite 120 von $\Sigma_{S\times\neg\varphi}$ folgt

$$(q, \zeta^{-1}(x_1), q_1) \in \delta, \quad (q_1, \zeta^{-1}(x_1), q_2) \in \delta, \quad (q_2, \zeta^{-1}(x_1), q_3) \in \delta, \quad \ldots$$

und somit

$$(\zeta^{-1}(x_1))^\omega \in L(\mathcal{A}_{\neg\varphi}, q).$$

Aus Gleichung 5.1.1 folgt

$$\forall s \in \Pi(\varphi) \colon M|_{S_{\neg\varphi}}(s) = 1 \Leftrightarrow s \in x_1.$$

(Andernfalls könnte die Transition $t_{(q,x_1,q_1)}$ nicht schalten). Somit gilt

$$\begin{aligned}\zeta^{-1}(x_1) &= \{s \mid s \in x_1 \wedge s \in \Pi(\varphi)\} \\ &= \{s \mid M|_{S_{\neg\varphi}}(s) = 1 \wedge s \in \Pi(\varphi)\} \\ &= \{s \mid M(s) = 1 \wedge s \in \Pi(\varphi)\} \\ &= \nu(M),\end{aligned}$$

woraus

$$(\nu(M))^\omega \in L(\mathcal{A}_{\neg\varphi}, q)$$

folgt. Dann gibt es nach Definition 5.1.6 auf Seite 120 von $\Sigma_{S\times\neg\varphi}$ eine Transition $t_{(q,M)} \in T_L$, sodaß

$$M_0 \xrightarrow{\sigma} M \xrightarrow{t_{(q,M)}} M',$$

wobei

$$\forall s \in S_V \cup S_H \colon M'(s) = M(s).$$

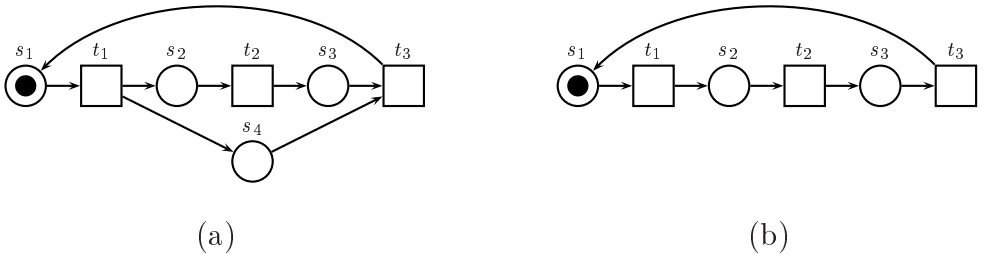
Mit $\sigma_1 \in L^\omega((N_h, M|_{S_h}))$ und somit $\sigma_1 \in T_H^\omega$ folgt unmittelbar

$$M_0 \xrightarrow{\sigma} M \xrightarrow{t_{(q,M)}} M' \xrightarrow{\sigma_1},$$

welches Definition 5.1.7 auf Seite 122 eines illegalen Livelocks entspricht.

■

Abbildung 5.4 *Sicheres S/T-Netzsystem mit (a) und ohne (b) implizite Stelle s_4*



(a) (b)

5.2 Reduktionsregeln

In diesem Abschnitt werden verschiedene Reduktionsregeln vorgestellt, die vor der Verifikation einer LTL-X-Eigenschaft auf das Produktnetzsystem unter Erhaltung der in Satz 5.1.3 auf Seite 126 beschriebenen Merkmale angewendet werden können. Dabei werden einerseits Methoden beschrieben, die auf Techniken der linearen Programmierung zurückgreifen und sehr effektiv sind, da sie das Netzsystem global untersuchen. Andererseits werden auch lokale Reduktionen betrachtet, die einfach anzuwenden sind, aber lediglich kleine Teilnetze inspizieren.
Den Ausgangspunkt für die in den folgenden Abschnitten betrachteten Reduktionsregeln bilden Techniken, die in [DE95, Rei85, Ber85, CS90] diskutiert wurden. Darunter befinden sich Methoden zur Erkennung impliziter Stellen sowie Regeln zur Abstraktion, Pre- und Post-Agglomeration von Netzknoten. Diese werden auf den in [EH01, Hel02] vorgestellten Ansatz zur Verifikation von LTL-X-Eigenschaften übertragen und können in diesem Zusammenhang teilweise verallgemeinert werden. Aufgrund dieser Generalisierungen besteht jedoch die Notwendigkeit, weitere Regeln zu betrachten. So führt beispielsweise die Betrachtung einer verallgemeinerten Regel zur Abstraktion von Netzknoten zu der Notwendigkeit, spezielle Situationen zu erkennen, in denen Transitionen tot sind. Solche speziellen Situationen können anhand der beiden Regeln $TT2$ und $TT3$ zur Erkennung toter Transitionen herausgefiltert werden. Auch eine verallgemeinerte Form der Pre-Agglomeration führt zu der Notwendigkeit, Transitionen zu erkennen, die in dem Kontext der Verifikation einer temporallogischen Eigenschaft als bedeutungslos erscheinen. Solche Transitionen können mit der Regel zur Erkennung bedeutungsloser Transitionen eliminiert werden.

5.2.1 Implizite Stellen

In diesem Abschnitt wird eine Möglichkeit zur Erkennung sogenannter *impliziter Stellen* vorgestellt. Im allgemeinen wird eine Stelle als implizit bezeichnet, falls sie niemals die einzige Ursache dafür darstellt, daß eine ihrer Ausgangstransitionen nicht schalten kann. In Abbildung 5.4 (a) auf dieser Seite ist ein sicheres S/T-Netzssystem dargestellt, das die implizite Stelle s_4 aufweist. Diese kann als Linearkombination der Stellen s_2 und s_3

verstanden werden, und es gilt:

$$\begin{aligned} & \forall M \in \mathcal{R}(M_0)\colon M(s_4) = M(s_2) + M(s_3) \\ \Rightarrow \quad & \forall M \in \mathcal{R}(M_0)\colon M(s_4) \geq M(s_3) \\ \Rightarrow \quad & \forall M \in \mathcal{R}(M_0)\colon M(s_3) \Rightarrow M(s_4) \end{aligned}$$

Somit ist klar, daß die Abwesenheit einer Marke auf der Stelle s_4 niemals ein Schalten der Transition t_3 verhindern kann, und somit kann s_4 aus dem Netzsystem entfernt werden (Abbildung 5.4 (b) auf der vorherigen Seite), ohne dessen Schaltfolgen zu verändern.
Eine Methode zur Erkennung impliziter Stellen in beschränkten S/T-Netzsystemen unter Verwendung von Techniken der linearen Programmierung wurde in [CS90] vorgeschlagen, und das im folgenden betrachtete Verfahren basiert auf diesem Ansatz.

Definition 5.2.1 (Implizite Stelle [CS90])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Stelle $s \in S$ sowie das S/T-Netzsystem Σ', welches aus Σ durch Entfernen von s erhalten wurde. Die Stelle s heißt *implizit*, falls $L(\Sigma) = L(\Sigma')$.

Eine implizite Stelle stellt also ein spracherhaltenes Element eines Netzsystems dar. Daraus leitet sich unmittelbar die folgende Charakterisierung einer impliziten Stelle ab.

Proposition 5.2.1 (Charakterisierung einer impliziten Stelle)
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Stelle $s \in S$ verhält sich genau dann implizit, wenn

$$\forall M \in \mathcal{R}(M_0)\ \forall t \in T\colon (\forall s' \in {}^\bullet t \setminus \{s\}\colon M(s') = 1) \Rightarrow M(s) \geq F((s,t))$$

Wie bereits erwähnt wurde, stellt s_4 in dem Netzsystem aus Abbildung 5.4 (a) auf der vorherigen Seite eine implizite Stelle dar. Würde man jedoch

$$M_0 = (0, 1, 0, 0)^t$$

als Anfangsmarkierung wählen, so wäre die Stelle s_4 nicht mehr implizit, da sie ein Schalten der Transition t_3 verhindern würde. Die einzige Schaltfolge des Netzsystems wäre t_2. In diesem Fall dürfte die Stelle s_4 nicht aus dem Netzsystem entfernt werden, da ansonsten die unendliche Schaltfolge $(t_2 t_3 t_1)^\omega$ ausführbar wäre. Es wird deutlich, daß es von der Anfangsmarkierung eines Netzsystems abhängen kann, ob sich eine Stelle implizit verhält oder nicht.
Allerdings ist es in dem obigen Beispiel möglich, für jede beliebige Anfangsmarkierung $M_0|_{S\setminus\{s_4\}}$ eine Belegung für $M_0(s_4)$ anzugeben, sodaß sich die Stelle s_4 implizit verhält. Dazu wähle man

$$M_0(s_4) = M_0(s_2) + M_0(s_3).$$

In diesem Fall wird s_4 als *strukturell implizite Stelle* bezeichnet.

Definition 5.2.2 (Strukturell implizite Stelle [CS90])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Stelle $s \in S$ heißt genau dann *strukturell implizit*, wenn es für jede beliebige Anfangsmarkierung $M_0|_{S\setminus\{s\}}$ eine Belegung für $M_0(s)$ gibt, sodaß sich s implizit verhält.

Proposition 5.2.2 gibt Aufschluß darüber, wie strukturell implizite Stellen mittels Techniken der linearen Programmierung erkannt werden können.

Proposition 5.2.2 (Strukturell implizite Stelle [CS90])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Stelle $s \in S$ verhält sich genau dann strukturell implizit, wenn das folgende Ungleichungssystem eine Lösung für den Variablenvektor $Y \geq \mathbf{0}$ besitzt:

$$\begin{aligned} Y^t \cdot \mathbf{N} &\leq l_s \\ Y(s) &= 0 \end{aligned}$$

Dabei bezeichnen $\mathbf{N}$ die Inzidenzmatrix von Σ und l_s den Inzidenzvektor von s.

Es soll nun gezeigt werden, wie die strukturell implizite Stelle s_4 in dem Netzsystem aus Abbildung 5.4 (a) auf Seite 130 mit Proposition 5.2.2 erkannt werden kann. Gemäß Proposition 5.2.2 ergibt sich für s_4 das folgende lineare Ungleichungssystem:

$$\begin{aligned} -Y(s_1) + Y(s_2) + Y(s_4) &\leq 1 & \qquad Y(s_1) &\geq 0 \\ -Y(s_2) + Y(s_3) &\leq 0 & \qquad Y(s_2) &\geq 0 \\ Y(s_1) - Y(s_3) - Y(s_4) &\leq -1 & \qquad Y(s_3) &\geq 0 \\ Y(s_4) &= 0 \end{aligned}$$

Der Vektor

$$Y = (0, 1, 1, 0)^t$$

stellt eine Lösung des linearen Ungleichungssystems dar, und somit wurde s_4 korrekterweise als strukturell implizite Stelle erkannt.
Wie bereits erwähnt wurde, kann für jede strukturell implizite Stelle s zu jeder beliebigen Anfangsmarkierung $M_0|_{S\setminus\{s\}}$ immer eine Belegung für $M_0(s)$ gefunden werden, sodaß sich s implizit verhält. Proposition 5.2.3 liefert eine Antwort auf die Fragestellung, wie eine solche Anfangsbelegung für s gefunden werden kann.

Proposition 5.2.3 (Minimale Anfangsmarkierung [CS90])
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine strukturell implizite Stelle $s \in S$. Eine obere Schranke für die minimale Anfangsmarkierung $M_0(s)$, unter der sich s implizit verhält, ist durch

$$M_0(s) = \mathit{Maximum}\{0, v\}$$

gegeben, wobei

$$v = \mathit{Minimum}\ \ Y^t \cdot M_0 + \mu,$$

sodaß

$$\begin{aligned} Y^t \cdot \mathbf{N} &\leq l_s \\ Y(s) &= 0 \\ \forall t \in s^\bullet : \sum_{s' \in {}^\bullet t} Y(s') + \mu &\geq 1 \end{aligned}$$

Dabei bezeichnen $Y \geq \mathbf{0}$ und μ Variablen.

Für die strukturell implizite Stelle s_4 des Netzsystems aus Abbildung 5.4 (a) auf Seite 130 soll nun entschieden werden, wie $M_0(s_4)$ gewählt werden muß, damit sich s_4 unter der Anfangsmarkierung

$$M_0|_{S\setminus\{s_4\}} = (0,1,0)^t$$

implizit verhält. Mit Proposition 5.2.3 auf der vorherigen Seite erhält man das folgende lineare Ungleichungssystem:

$$\text{Minimiere } Y(s_2) + \mu, \text{ sodaß}$$

$$\begin{aligned} -Y(s_1) + Y(s_2) + Y(s_4) &\leq 1 & \qquad Y(s_1) &\geq 0 \\ -Y(s_2) + Y(s_3) &\leq 0 & \qquad Y(s_2) &\geq 0 \\ Y(s_1) - Y(s_3) - Y(s_4) &\leq -1 & \qquad Y(s_3) &\geq 0 \\ Y(s_3) + Y(s_4) + \mu &\geq 1 & \qquad Y(s_4) &= 0 \end{aligned}$$

Eine optimale Lösung dieses linearen Ungleichungssystems ist durch

$$Y = (0,1,1,0)^t \quad \text{und} \quad \mu = 0$$

gegeben, woraus nach Proposition 5.2.3 auf der vorherigen Seite folgt, daß

$$M_0(s_4) = \mathit{Maximum}\{0, Y(s_2) + \mu\} = 1$$

gewählt werden sollte, damit sich die Stelle s_4 für $M_0|_{S\setminus\{s_4\}} = (0,1,0)^t$ garantiert implizit verhält.
Unter Einbeziehung der Ergebnisse von Proposition 5.2.2 auf der vorherigen Seite und Proposition 5.2.3 auf der vorherigen Seite kann nun unmittelbar eine Regel angegeben werden, welche die Erkennung von impliziten Stellen eines Netzsystems ermöglicht.

Definition 5.2.3 (Regel zur Erkennung impliziter Stellen [ES01a])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Stelle $s \in S$ genügt genau dann der Regel zur Erkennung impliziter Stellen, wenn das folgende lineare Ungleichungssystem eine Lösung für den Variablenvektor $Y \geq \mathbf{0}$ und die Variable μ besitzt:

$$\text{Minimiere } Y^t \cdot M_0 + \mu, \text{ sodaß}$$

$$\begin{aligned} Y^t \cdot \mathbf{N} &\leq l_s \\ Y(s) &= 0 \\ \forall t \in s^\bullet \colon \sum_{s' \in {}^\bullet t} Y(s') + \mu &\geq 1 \\ Y^t \cdot M_0 + \mu &\leq M_0(s) \end{aligned}$$

Das durch Entfernen der impliziten Stelle erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r = (S^r, T^r, F^r)$ ist definiert als:

$$\begin{aligned} S^r &= S \setminus \{s\} \\ T^r &= T \\ F^r &= F \cap ((S^r \times T^r) \cup (T^r \times S^r)) \\ M_0^r &= M_0|_{S^r} \end{aligned}$$

Definition 5.2.3 auf der vorherigen Seite stellt ein hinreichendes, aber keinesfalls notwendiges Kriterium für die Erkennung von impliziten Stellen dar, d.h., es gibt sowohl strukturell implizite Stellen, die sich auch unter einer kleineren als der in Proposition 5.2.3 auf Seite 132 beschriebenen Anfangsmarkierung implizit verhalten, als auch implizite Stellen, die nicht strukturell implizit sind (siehe beispielsweise [CS90]).

Satz 5.2.1 (Implizite Stellen bewahren LTL-X-Eigenschaften [ES01a])
Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ und das nach Definition 5.2.3 auf der vorherigen Seite erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi}$. Die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 gelten genau dann in $\Sigma_{S\times\neg\varphi}$, wenn sie in $\Sigma^r_{S\times\neg\varphi}$ gelten.

Beweis: Mit den Ergebnissen der Propositionen 5.2.2 und 5.2.3 auf Seite 132 folgt unmittelbar, daß sich die aus dem Netzsystem entfernten Stellen gemäß Definition 5.2.1 auf Seite 131 implizit verhalten. Nach Definition 5.2.1 stellen implizite Stellen spracherhaltene Elemente dar, d.h., die Schaltfolgen eines Netzsystems werden durch das Entfernen bzw. Hinzufügen solcher Stellen nicht verändert, weshalb die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 unberührt bleiben. ■

5.2.2 Erkennen toter Transitionen mittels linearer Programmierung

In vielen Fällen werden die zu verifizierenden Systeme zunächst in Formalismen spezifiziert, die Ähnlichkeiten zu höheren Programmiersprachen aufweisen, wie beispielsweise B(PN)^2 [BH93], die eine Modellierungssprache für das Pep-Tool [GRT+95] und das Model-Checking-Kit [SSE03] darstellt. Anschließend werden die Systeme automatisch in sichere S/T-Netzsysteme überführt. Als Nebenwirkung dieser automatischen Übersetzung kann es jedoch vorkommen, daß Redundanzen in die Netzsysteme eingefügt

werden, wie zum Beispiel Stellen, die niemals markiert sein können. Folglich können auch deren Eingangs- und Ausgangstransitionen niemals schalten. Sofern solche Stellen nicht als atomare Propositionen innerhalb der zu verifizierenden LTL-X-Formel verwendet werden, sind sie sowie deren Eingangs- und Ausgangstransitionen überflüssig und können aus dem Netzsystem entfernt werden, da sie keine notwendigen Informationen für die Verifikation der LTL-X-Eigenschaft liefern. Das Entfernen solcher Stellen und Transitionen beeinflußt nicht die Größe des Präfixes, da dieses sowieso nur die erreichbaren Markierungen kodiert, aber es kann in erheblichem Maße die Zeit verringern, welche während des Verifikationsvorgangs für die Erzeugung des Präfixes benötigt wird.

Definition 5.2.4 (Regel TT1 zur Erkennung toter Transitionen [ES01a])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Die Stellen einer Menge $S_d \subseteq S$ und die Transitionen einer Menge $T_d \subseteq T$ genügen der Regel TT1, falls die folgenden Bedingungen erfüllt sind:

(i) Das Ungleichungssystem

$$Y^t \cdot \mathbf{N} \leq \mathbf{0}$$
$$Y^t \cdot M_0 = 0$$

besitzt eine Lösung für den Variablenvektor $Y \geq \mathbf{0}$, wobei $\mathbf{N}$ die Inzidenzmatrix von Σ bezeichne.

(ii) Für die Mengen $S_d \subseteq S$ und $T_d \subseteq T$ gilt:

$$\forall s \in S\colon s \in S_d \Leftrightarrow Y(s) > 0$$
$$\forall t \in T\colon t \in T_d \Leftrightarrow \exists s \in S_d\colon s \in {}^\bullet t \cup t^\bullet$$

Das durch Entfernen der Mengen S_d und T_d erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r = (S^r, T^r, F^r)$ ist definiert als:

$$S^r = S \setminus S_d$$
$$T^r = T \setminus T_d$$
$$F^r = F \cap ((S^r \times T^r) \cup (T^r \times S^r))$$
$$M_0^r = M_0|_{S^r}$$

Intuitiv bedeutet Definition 5.2.4, daß eine Lösung der Gleichung

$$Y^t \cdot M_0 = 0$$

für $Y \neq \mathbf{0}$ eine Menge von anfänglich unmarkierten Stellen liefert und eine Lösung des Ungleichungssystems

$$Y^t \cdot \mathbf{N} \leq \mathbf{0}$$

für $Y \neq \mathbf{0}$ eine Menge von Stellen liefert, sodaß das Schalten einer beliebigen Transition die Anzahl der Marken auf diesen Stellen nicht erhöht. Folglich liefert eine Lösung $Y \neq \mathbf{0}$

Abbildung 5.5 *Sicheres S/T-Netzsystem, für das die toten Transitionen t_1 und t_2 nach Regel TT1 (Definition 5.2.4 auf der vorherigen Seite) erkannt werden (a), und sicheres S/T-Netzsystem, für das die toten Transitionen t_1 und t_2 nicht nach Regel TT1 erkannt werden (b)*

(a) (b)

für das ganze Ungleichungssystem eine Menge von Stellen, die anfänglich unmarkiert sind und auch für keine Schaltfolge des Netzsystems jemals Marken erhalten werden. Als Konsequenz daraus können auch die Eingangs- und Ausgangstransitionen dieser Stellen unter keiner erreichbaren Markierung aktiviert sein.
Dieses soll anhand des Netzsystems Σ aus Abbildung 5.5 (a) näher erläutert werden. Gemäß Definition 5.2.4 auf der vorherigen Seite wird das folgende Ungleichungssystem aufgestellt:

$$\begin{aligned} -Y(s_1) + Y(s_2) - Y(s_3) &\leq 0 & Y(s_1) &\geq 0 \\ -Y(s_2) + Y(s_3) &\leq 0 & Y(s_2) &\geq 0 \\ Y(s_1) &= 0 & Y(s_3) &\geq 0 \end{aligned}$$

Daraus folgt unmittelbar, daß der Vektor

$$Y = (0, \iota, \iota)^t \qquad (\iota > 0 \text{ beliebig})$$

beliebig viele Lösungen ungleich Null für das Ungleichungssystem liefert. Das bedeutet, daß die Stellen s_2 und s_3 für jede Schaltfolge von Σ immer unmarkiert bleiben werden und folglich die Transitionen $t_1 \in s_3^\bullet$ und $t_2 \in s_2^\bullet$ niemals werden schalten können.
Das in Definition 5.2.4 auf der vorherigen Seite erhaltene Ungleichungssystem liefert ein hinreichendes Kriterium zur Erkennung toter Transitionen, stellt aber leider kein notwendiges Kriterium dar, wie anhand des Netzsystems aus Abbildung 5.5 (b) gezeigt werden kann. Das Ungleichungssystem sieht nun folgendermaßen aus:

$$\begin{aligned} -Y(s_1) + Y(s_2) - Y(s_3) &\leq 0 & Y(s_1) &\geq 0 \\ Y(s_3) &\leq 0 & Y(s_2) &\geq 0 \\ Y(s_1) &= 0 & Y(s_3) &\geq 0 \end{aligned}$$

Daraus folgt unmittelbar, daß der Vektor

$$Y = (0, 0, 0)^t$$

die einzige Lösung für das Ungleichungssystem liefert und somit keine Lösung $Y \neq \mathbf{0}$ existiert. Die Transitionen t_1 und t_2 sind zwar tot, werden aber in diesem Fall nicht nach Regel TT1 (Definition 5.2.4 auf Seite 135) erkannt.

Proposition 5.2.4 (TT1-Transitionen sind tot)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine der Regel TT1 (Definition 5.2.4 auf Seite 135) genügende Transition $t \in T_d$. Es gilt:

$$\nexists M \in \mathcal{R}(M_0) \colon M_0 \xrightarrow{\sigma t} M$$

Beweis: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine der Regel TT1 (Definition 5.2.4 auf Seite 135) genügende Transition $t \in T_d$. Daraus folgt die Existenz einer Stelle $s \in {}^\bullet t \cup t^\bullet$ mit $Y(s) > 0$, wobei $Y \neq \mathbf{0}$ den Lösungsvektor für das in Definition 5.2.4 auf Seite 135 gegebene Ungleichungssystem bezeichnet.
Angenommen, es gibt eine erreichbare Markierung $M \in \mathcal{R}(M_0)$ mit

$$M_0 \xrightarrow{\sigma t} M,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß t nicht in σ vorkommt. Daraus folgt unmittelbar die Existenz einer erreichbaren Markierung $M_3 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma} M_3 \xrightarrow{t} M,$$

wobei

$$\forall s' \in {}^\bullet t \colon M_3(s') = 1.$$

(i) Sei $s \in {}^\bullet t$, d.h., $M_3(s) = 1$.
Wegen $Y(s) > 0$ und $Y^t \cdot M_0 = 0$ gilt jedoch $M_0(s) = 0$, und deshalb muß es eine Transition $t' \in {}^\bullet s \setminus s^\bullet$, eine Zerlegung von σ in $\sigma_1 t' \sigma_2$ und erreichbare Markierungen $M_1, M_2 \in \mathcal{R}(M_0)$ geben, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t'} M_2 \xrightarrow{\sigma_2} M_3 \xrightarrow{t} M,$$

wobei

$$\forall s' \in {}^\bullet t' \colon M_1(s') = 1.$$

Desweiteren gilt

$$\begin{aligned} & Y^t \cdot \mathbf{N} \leq \mathbf{0} \\ \Rightarrow \quad & \sum_{s' \in t'^\bullet} Y(s') - \sum_{s' \in {}^\bullet t'} Y(s') \leq 0 \\ \Leftrightarrow \quad & \sum_{s' \in t'^\bullet} Y(s') \leq \sum_{s' \in {}^\bullet t'} Y(s') \end{aligned}$$

Aus $s \in t'^{\bullet} \setminus {}^{\bullet}t'$ und $Y(s) > 0$ folgt daher unmittelbar

$$\exists s' \in {}^{\bullet}t' : Y(s') > 0.$$

Wegen $Y^t \cdot M_0 = 0$ gilt $M_0(s') = 0$, aber wegen $M_1(s') = 1$ muß in σ_1 eine Transition $t'' \in {}^{\bullet}s' \setminus s'^{\bullet}$ vorkommen. Es folgt dieselbe Argumentationskette wie zuvor, die schließlich nach höchstens $|\sigma_1|$-maliger Anwendung zu dem Widerspruch

$$\exists s'' \in S : Y(s'') > 0 \quad \text{und} \quad M_0(s'') = 1$$

führt.

(ii) Sei $s \in t^{\bullet} \setminus {}^{\bullet}t$, d.h., $M(s) = 1$.
Dieser Fall ist identisch mit der Argumentationskette, daß in Fall (i) die Transition $t' \in {}^{\bullet}s \setminus s^{\bullet}$ niemals schalten kann.

■

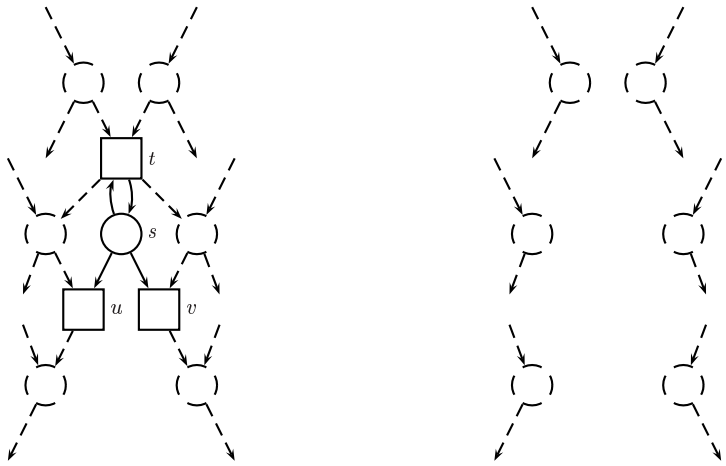
Satz 5.2.2 (TT1-Stellen/Transitionen bewahren LTL-X-Eigenschaften)
Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ und das nach Definition 5.2.4 auf Seite 135 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi}$. Die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 gelten genau dann in $\Sigma_{S\times\neg\varphi}$, wenn sie in $\Sigma^r_{S\times\neg\varphi}$ gelten.

Beweis: Nach Proposition 5.2.4 auf der vorherigen Seite sind die Transitionen der nach Regel TT1 (Definition 5.2.4 auf Seite 135) erkannten Menge T_d tot, da die Stellen aus S_d nie markiert sein können. Trivialerweise verändert das Entfernen bzw. Hinzufügen solcher Stellen und Transitionen nicht die Schaltfolgen eines Netzsystems, weshalb die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 unberührt bleiben. ■

5.2.3 Erkennen toter Transitionen mittels lokaler Teilnetze

In Abschnitt 5.2.2 auf Seite 134 wurde ein Verfahren besprochen, mit dem tote Transitionen mittels Techniken der linearen Programmierung erkannt werden können. Da dieses Verfahren jedoch lediglich ein hinreichendes, aber keinesfalls notwendiges Kriterium für das Erkennen toter Transitionen liefert, werden in diesem Abschnitt zwei strukturelle, lokale Netzeigenschaften betrachtet, anhand derer tote Transitionen entdeckt werden können.
Die erste strukturelle Eigenschaft ist in dem Teilnetz aus Abbildung 5.6 (a) auf der nächsten Seite dargestellt. Die Idee besteht darin, daß die Transition t nur schalten kann, falls die Stelle s eine Marke enthält, aber s eine Marke nur durch das Schalten von t bekommen kann. Falls s also anfänglich unmarkiert ist, wird t unter keiner erreichbaren Markierung schalten können. Somit sind die Transitionen t, u und v tot und können mitsamt der Stelle s aus dem Netz entfernt werden (Abbildung 5.6 (b) auf der nächsten Seite).

Abbildung 5.6 *Teilnetz eines sicheren S/T-Netzsystems mit (a) und ohne (b) die nach der Regel TT2 erkannten toten Transitionen t, u und v*



(a) (b)

Definition 5.2.5 (Regel TT2 zur Erkennung toter Transitionen)
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Transition $t \in T$ genügt der Regel TT2, falls es eine Stelle $s \in {}^\bullet t \cap t^\bullet$ gibt, sodaß gilt:

$$\begin{aligned} {}^\bullet s &= \{t\} \\ M_0(s) &= 0 \end{aligned}$$

Das durch Anwendung der Regel TT2 erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r = (S^r, T^r, F^r)$ ist definiert als:

$$\begin{aligned} S^r &= S \setminus \{s\} \\ T^r &= T \setminus s^\bullet \\ F^r &= F \cap ((S^r \times T^r) \cup (T^r \times S^r)) \\ M_0^r &= M_0|_{S^r} \end{aligned}$$

Proposition 5.2.5 (TT2-Transitionen sind tot)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine der Regel TT2 (Definition 5.2.5) genügende Transition $t \in T$. Es gilt:

$$\nexists M \in \mathcal{R}(M_0) \colon M_0 \xrightarrow{\sigma t} M$$

Beweis: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine der Regel TT2 (Definition 5.2.5) genügende Transition $t \in T$. Desweiteren

bezeichne $s \in {}^\bullet t \cap t^\bullet$ diejenige Stelle, welche die Bedingungen von Definition 5.2.5 erfüllt. Angenommen, es gibt eine erreichbare Markierung $M \in \mathcal{R}(M_0)$ mit

$$M_0 \xrightarrow{\sigma t} M,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß t nicht in σ vorkommt. Daraus folgt jedoch die Existenz einer Markierung $M' \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma} M' \xrightarrow{t} M.$$

Aus $s \in {}^\bullet t$ folgt unmittelbar $M'(s) = 1$. Daraus folgt mit $M_0(s) = 0$, daß es eine Zerlegung von σ in $\sigma_1 t' \sigma_2$ geben muß, sodaß

$$t' \in {}^\bullet s \setminus s^\bullet.$$

Wegen ${}^\bullet s = \{t\}$ und $t \in s^\bullet$ folgt unmittelbar ${}^\bullet s \setminus s^\bullet = \emptyset$, welches jedoch im Widerspruch zu $t' \in {}^\bullet s \setminus s^\bullet$ steht. ■

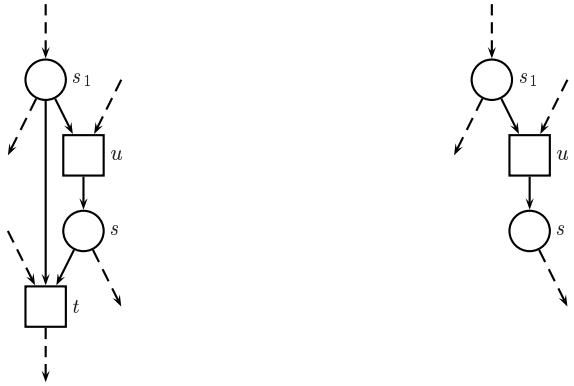
Der Beweis von Proposition 5.2.5 auf der vorherigen Seite zeigt, daß eine der Regel TT2 genügende Transition t unter keiner erreichbaren Markierung von Σ aktiviert ist, da die Stelle $s \in {}^\bullet t \cap t^\bullet$, welche die Kriterien von Definition 5.2.5 auf der vorherigen Seite erfüllt, nie eine Marke erhalten kann und deshalb unter jeder erreichbaren Markierung unmarkiert bleibt. Folglich können auch die Transitionen aus $s^\bullet$ niemals schalten und mitsamt s aus dem Netzsystem entfernt werden, ohne dessen dynamisches Verhalten zu verändern.
In Abbildung 5.7 (a) auf der nächsten Seite ist eine weitere lokale Netzsituation dargestellt, die eine tote Transition beinhaltet. Die Idee besteht darin, daß die Transition t nur dann aktiviert sein kann, wenn die Stellen s und s_1 jeweils eine Marke enthalten. Die Stelle s kann eine Marke nur durch das Schalten der Transition u erhalten. Diese entfernt jedoch ihrerseits eine Marke von s_1, weshalb s und s_1 in einem sicheren Netzsystem niemals gleichzeitig markiert sein können. Folglich kann die Transition t nie schalten und aus dem Netz entfernt werden (Abbildung 5.7 (b) auf der nächsten Seite).

Definition 5.2.6 (Regel TT3 zur Erkennung toter Transitionen)
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Transition $t \in T$ genügt der Regel TT3, falls es eine Stelle $s \in {}^\bullet t$ gibt, sodaß gilt:

$$\begin{aligned} |{}^\bullet s| &= 1 \\ ({}^\bullet s)^\bullet &= \{s\} \\ {}^\bullet t \cap {}^\bullet({}^\bullet s) &\neq \emptyset \\ M_0(s) &= 0 \end{aligned}$$

Das durch Entfernen der Transition t erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r =$

Abbildung 5.7 *Teilnetz eines sicheren S/T-Netzsystems mit (a) und ohne (b) die nach der Regel TT3 erkannte tote Transition t*



(a) (b)

(S^r, T^r, F^r) ist definiert als:

$$\begin{aligned} S^r &= S \\ T^r &= T \setminus \{t\} \\ F^r &= F \cap ((S^r \times T^r) \cup (T^r \times S^r)) \\ M_0^r &= M_0|_{S^r} \end{aligned}$$

Proposition 5.2.6 (TT3-Transitionen sind tot)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine der Regel TT3 (Definition 5.2.6 auf der vorherigen Seite) genügende Transition $t \in T$. Es gilt:

$$\nexists M \in \mathcal{R}(M_0) \colon M_0 \xrightarrow{\sigma t} M$$

Beweis: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine der Regel TT3 (Definition 5.2.6 auf der vorherigen Seite) genügende Transition $t \in T$. Desweiteren bezeichne $s \in {}^\bullet t$ diejenige Stelle, welche die Bedingungen von Definition 5.2.6 auf der vorherigen Seite erfüllt.

(i) Sei $s \in {}^\bullet t \cap t^\bullet$.
Aus $|{}^\bullet s| = 1$ folgt ${}^\bullet s = \{t\}$ und somit genügt die Transition t der Regel TT2 (Definition 5.2.5 auf Seite 139). Aus Proposition 5.2.5 auf Seite 139 folgt, daß t tot ist.

(ii) Sei $s \in {}^\bullet t \setminus t^\bullet$.
Dann gibt es eine Transition $u \neq t$, sodaß

$${}^\bullet s = \{u\} \quad \text{und} \quad u^\bullet = \{s\}.$$

(a) Sei $s \in {}^\bullet t \cap {}^\bullet u$.
Dann gilt $s \in {}^\bullet u \cap u^\bullet$ und ${}^\bullet s = \{u\}$. Somit genügt die Transition u der Regel TT2 (Definition 5.2.5 auf Seite 139), und aus Proposition 5.2.5 auf Seite 139 folgt, daß u tot ist. Wegen ${}^\bullet s = \{u\}$ kann die Stelle s niemals eine Marke erhalten, woraus unmittelbar folgt, daß auch die Transition $t \in s^\bullet$ tot ist.

(b) Sei $s \notin {}^\bullet t \cap {}^\bullet u$.
Dann gibt es eine Stelle $s_1 \neq s$, sodaß

$$s_1 \in {}^\bullet t \cap {}^\bullet u.$$

Angenommen, es gibt eine erreichbare Markierung $M \in \mathcal{R}(M_0)$ mit

$$M_0 \xrightarrow{\sigma t} M,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß t nicht in σ vorkommt. Daraus folgt jedoch die Existenz einer Markierung $M_3 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma} M_3 \xrightarrow{t} M.$$

Aus $s, s_1 \in {}^\bullet t$ folgt unmittelbar

$$M_3(s) = M_3(s_1) = 1.$$

Daraus folgt wegen $M_0(s) = 0$, daß die Transition $u \in {}^\bullet s$ in der Schaltfolge σ vorkommen muß. Daher existiert eine Zerlegung von σ in $\sigma_1 u \sigma_2$, und ohne Einschränkung der Allgemeinheit kann diese Zerlegung derart gewählt werden, daß u nicht in σ_2 vorkommt. Folglich gibt es erreichbare Markierungen $M_1, M_2 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{u} M_2 \xrightarrow{\sigma_2} M_3 \xrightarrow{t} M.$$

Ferner gilt

$$M_1(s_1) = 1, \quad M_1(s) = 0, \quad M_2(s_1) = 0, \quad M_2(s) = 1,$$

d.h., das Ausführen der Schaltfolge σ_2 erzeugt wiederum eine Marke auf der Stelle s_1, welche zwischenzeitlich durch das Schalten der Transition u von dieser entfernt wurde. Die Marke auf der Stelle s wird für das Schalten von $M_2 \xrightarrow{\sigma_2} M_3$ nicht benötigt. (Andernfalls müßte wegen $M_2(s) = M_3(s) = 1$ auch die Transition $u \in {}^\bullet s$ in σ_2 vorkommen, welches aber einen Widerspruch zu der Annahme darstellt, daß u nicht in σ_2 vorkommt). Nun bezeichne $M_2^<$ eine Markierung, die wie folgt definiert ist:

$$M_2^<(s') = \begin{cases} 0 & \text{falls } s' = s \\ M_2(s') & \text{sonst} \end{cases}$$

Damit gilt offensichtlich, daß eine Markierung $M_3^<$ existiert, sodaß

$$M_2^< \xrightarrow{\sigma_2} M_3^<.$$

Mit der Beziehung

$$\forall s' \in S\colon M_1(s') \geq M_2^<(s')$$

folgt unmittelbar die Existenz einer erreichbaren Markierung $M_4 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4.$$

Da $M_1(s_1) = 1$ und die Schaltfolge σ_2 eine neue Marke auf der Stelle s_1 erzeugt, gilt $M_4(s_1) = 2$. Dieses stellt jedoch einen Widerspruch zu der Sicherheit des Netzsystems dar.

■

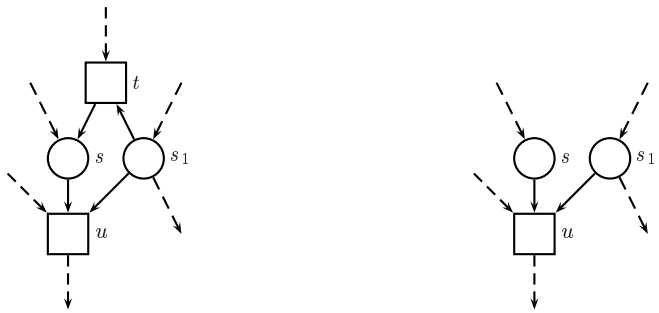
Satz 5.2.3 (TT2- und TT3-Transitionen bewahren LTL-X-Eigenschaften)
Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ und das nach den Definitionen 5.2.5 auf Seite 139 oder 5.2.6 auf Seite 140 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi}$. Die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 gelten genau dann in $\Sigma_{S\times\neg\varphi}$, wenn sie in $\Sigma^r_{S\times\neg\varphi}$ gelten.

Beweis: Nach den Propositionen 5.2.5 auf Seite 139 und 5.2.6 auf Seite 141 sind die nach den Regeln TT2 (Definition 5.2.5 auf Seite 139) und TT3 (Definition 5.2.6 auf Seite 140) erkannten Transitionen tot. Trivialerweise verändert das Entfernen bzw. Hinzufügen von toten Transitionen nicht die Schaltfolgen eines Netzsystems, weshalb die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 unberührt bleiben. ■

5.2.4 Bedeutungslose Transitionen

In diesem Abschnitt werden Transitionen betrachtet, die im Gegensatz zu toten Transitionen zwar schalten können, aber für die Verifikation von LTL-X-Eigenschaften nicht weiter von Bedeutung sind. Abbildung 5.8 (a) auf der nächsten Seite zeigt ein Teilnetz eines sicheren S/T-Netzsystems, in dem es keine Schaltfolge gibt, in der die Transition u nach der Transition t vorkommt. Das liegt darin begründet, daß die Stelle s nach dem Schalten von t zwar eine Marke enthält, aber die Stelle s_1 dann immer unmarkiert bleibt und u deshalb niemals aktiviert werden kann. Falls also die Transition t in einer Schaltfoge vorkommt, so besteht deren einzige Aktion darin, eine Marke auf die Stelle s zu legen. Diese Marke „verhungert“ dann dort, da die Transition u nie mehr aktiviert sein wird und die Marke von s nicht weggeschaltet werden kann. Folglich können alle unendlichen Schaltfolgen des Netzsystems, in denen t vorkommt, auch unabhängig vom Auftreten von t ausgeführt werden. Die Transition t ist also „bedeutungslos“ für die unendlichen Schaltfolgen des Netzsystems und kann somit aus diesem entfernt werden.

Abbildung 5.8 *Teilnetz eines sicheren S/T-Netzsystems mit (a) und ohne (b) die bedeutungslose Transition t*



(a) (b)

Definition 5.2.7 (Bedeutungslose Transitionen)
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Transition $t \in T$ heißt *bedeutungslos*, falls es eine Stelle $s \in t^\bullet \setminus {}^\bullet t$ gibt, sodaß gilt:

$$\begin{aligned} t^\bullet &= \{s\} \\ s^\bullet &\neq \emptyset \\ \forall u \in s^\bullet \colon {}^\bullet t \cap {}^\bullet u &\neq \emptyset \\ M_0(s) &= 0 \end{aligned}$$

Das durch Entfernen der bedeutungslosen Transition t erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r = (S^r, T^r, F^r)$ ist definiert als:

$$\begin{aligned} S^r &= S \\ T^r &= T \setminus \{t\} \\ F^r &= F \cap ((S^r \times T^r) \cup (T^r \times S^r)) \\ M_0^r &= M_0|_{S^r} \end{aligned}$$

Proposition 5.2.7 (Bedeutungslose Transitionen)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine nach Definition 5.2.7 bedeutungslose Transition $t \in T$. Es gilt:

$$\forall u \in (t^\bullet)^\bullet \colon \ \nexists M \in \mathcal{R}(M_0) \colon M_0 \xrightarrow{\sigma_1 t \sigma_2 u} M$$

Beweis: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine nach Definition 5.2.7 bedeutungslose Transition $t \in T$. Desweiteren bezeichne $s \in t^\bullet \setminus {}^\bullet t$ diejenige Stelle, welche den Bedingungen von Definition 5.2.7 genügt. Dann gilt

$$\forall u \in s^\bullet \colon u \neq t \wedge {}^\bullet t \cap {}^\bullet u \neq \emptyset.$$

Bezeichne u also eine Transition aus $s^\bullet$. Dann gibt es eine Stelle $s_1 \in {}^\bullet t \cap {}^\bullet u$, wobei $s_1 \neq s$. Angenommen, es gibt eine erreichbare Markierung $M \in \mathcal{R}(M_0)$ mit

$$M_0 \xrightarrow{\sigma_1 t \sigma_2 u} M,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß in σ_2 keine Transition aus $s^\bullet$ vorkommt. Daraus folgt jedoch die Existenz von erreichbaren Markierungen M_1, M_2, $M_3 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t} M_2 \xrightarrow{\sigma_2} M_3 \xrightarrow{u} M.$$

Aus $s_1 \in {}^\bullet t$ folgt

$$M_1(s_1) = 1.$$

Aus $s \in t^\bullet$ und $s_1 \in {}^\bullet t \setminus t^\bullet$ folgt

$$M_2(s) = 1 \quad \text{und} \quad M_2(s_1) = 0.$$

Aus $s, s_1 \in {}^\bullet u$ folgt

$$M_3(s) = M_3(s_1) = 1.$$

Das bedeutet, daß die Schaltfolge σ_2 eine neue Marke auf der Stelle s_1 erzeugen muß. Für das Schalten von $M_2 \xrightarrow{\sigma_2} M_3$ wird die Marke der Stelle s nicht benötigt. (Andernfalls müßte mindestens eine Transition aus $s^\bullet$ in der Schaltfolge σ_2 vorkommen, welches jedoch einen Widerspruch zu der Annahme darstellt, daß in σ_2 keine Transition aus $s^\bullet$ vorkommt). Nun bezeichne $M_2^<$ eine Markierung, die wie folgt definiert ist:

$$M_2^<(s') = \begin{cases} 0 & \text{falls } s' = s \\ M_2(s') & \text{sonst} \end{cases}$$

Damit gilt offensichtlich, daß eine Markierung $M_3^<$ existiert, sodaß

$$M_2^< \xrightarrow{\sigma_2} M_3^<.$$

Mit der Beziehung

$$\forall s' \in S\colon M_1(s') \geq M_2^<(s')$$

folgt unmittelbar die Existenz einer erreichbaren Markierung $M_4 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4.$$

Da $M_1(s_1) = 1$ und die Schaltfolge σ_2 eine neue Marke auf der Stelle s_1 erzeugt, gilt $M_4(s_1) = 2$. Dieses stellt jedoch einen Widerspruch zu der Sicherheit des Netzsystems dar. ■

Proposition 5.2.8 (Bedeutungslose Transitionen)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine nach Definition 5.2.7 auf der vorherigen Seite bedeutungslose Transition $t \in T$. Es gilt:

$$M_0 \xrightarrow{\sigma_1 t \sigma_2} \quad \Rightarrow \quad M_0 \xrightarrow{\sigma_1 \sigma_2}$$

BEWEIS: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$ und eine nach Definition 5.2.7 auf Seite 144 bedeutungslose Transition $t \in T$. Desweiteren bezeichne $s \in t^\bullet \setminus {}^\bullet t$ diejenige Stelle, welche den Bedingungen von Definition 5.2.7 auf Seite 144 genügt. Angenommen, es gibt eine unendliche Schaltfolge

$$M_0 \xrightarrow{\sigma_1 t \sigma_2}\!,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß t nicht in σ_1 vorkommt. Daraus folgt die Existenz erreichbarer Markierungen $M_1, M_2 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t} M_2 \xrightarrow{\sigma_2} .$$

Wegen $s \in t^\bullet$ gilt $M_2(s) = 1$. Die Marke auf der Stelle s wird für die Ausführung der Schaltfolge σ_2 nicht benötigt. (Andernfalls müßte eine Transition aus $s^\bullet$ in der Schaltfolge σ_2 vorkommen, was jedoch nach Proposition 5.2.7 auf Seite 144 ausgeschlossen ist). Nun bezeichne $M_2^<$ eine Markierung, die wie folgt definiert ist:

$$M_2^<(s') = \begin{cases} 0 & \text{falls } s' = s \\ M_2(s') & \text{sonst} \end{cases}$$

Damit gilt offensichtlich

$$M_2^< \xrightarrow{\sigma_2} .$$

Mit der Beziehung

$$\forall s' \in S\colon M_1(s') \geq M_2^<(s')$$

folgt unmittelbar

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} .$$

■

Satz 5.2.4 (Bedeutungslose Transitionen bewahren LTL-X-Eigenschaften)
Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ und das nach Definition 5.2.7 auf Seite 144 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi}$. Die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 gelten genau dann in $\Sigma_{S\times\neg\varphi}$, wenn sie in $\Sigma^r_{S\times\neg\varphi}$ gelten.

BEWEIS: Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi} = (N_\mathcal{B}, M_0)$ mit $N_\mathcal{B} = (S, T, F, S_E)$, eine nach Definition 5.2.7 auf Seite 144 bedeutungslose Transition $t \in T$ sowie das nach Definition 5.2.7 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi} = (N^r_\mathcal{B}, M^r_0)$ mit $N^r_\mathcal{B} = (S^r, T^r, F^r, S^r_E)$. Aus $|t^\bullet| = 1$, $t^\bullet \cap {}^\bullet t = \emptyset$ und Proposition 5.1.1 auf Seite 122 folgt $t \in T_H$.

(i) ($\Rightarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. Falls t in σ vorkommt, gibt es eine Zerlegung von σ in $\sigma_1 t \sigma_2$. Mit Proposition 5.2.8 auf der vorherigen Seite folgt unmittelbar die Existenz einer unendlichen Schaltfolge $M_0 \xrightarrow{\sigma_1\sigma_2}$, in der wegen

$t \notin T_I$ ($\subseteq T_B$) ebenfalls unendlich viele Transitionen aus T_I vorkommen. Daraus folgt jedoch unmittelbar $M_0^r \xrightarrow{\sigma_1\sigma_2}$ in $\Sigma^r_{S\times\neg\varphi}$.

($\Leftarrow$) Trivial, da jede Schaltfolge σ in $\Sigma^r_{S\times\neg\varphi}$ auch unmittelbar in $\Sigma_{S\times\neg\varphi}$ vorhanden ist.

(ii) ($\Rightarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine Schaltfolge $M_0 \xrightarrow{\sigma} M$, sodaß $L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})) \neq \emptyset$ und $L^\omega((N_h, M|_{S_h})) \neq \emptyset$. Seien also $\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M|_{S_{\neg\varphi}}))$ und $\sigma_h \in L^\omega((N_h, M|_{S_h}))$ gegeben.

(1) Sei t in σ enthalten, d.h., $\sigma = \sigma_1 t \sigma_2$. Daraus folgt die Existenz erreichbarer Markierungen $M_1, M_2 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t} M_2 \xrightarrow{\sigma_2} M,$$

und mit Proposition 5.2.8 auf Seite 145 folgt die Existenz einer erreichbaren Markierung $M_3 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_3.$$

Daraus folgt unmittelbar

$$M_0^r \xrightarrow{\sigma_1\sigma_2} M_3$$

in $\Sigma^r_{S\times\neg\varphi}$. Darüberhinaus gilt

$$\forall s \in S \setminus t^\bullet \colon M_3(s) \geq M(s). \tag{5.2.1}$$

Aus $t \in T_H$ folgt $t^\bullet \subseteq S_H$. Somit gilt

$$\forall s \in S_{\neg\varphi} \colon M_3(s) \geq M(s),$$

woraus unmittelbar

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M_3|_{S_{\neg\varphi}})) \quad \text{und} \quad \sigma_{\neg\varphi} \in L((N^r_{\neg\varphi}, M_3|_{S^r_{\neg\varphi}}))$$

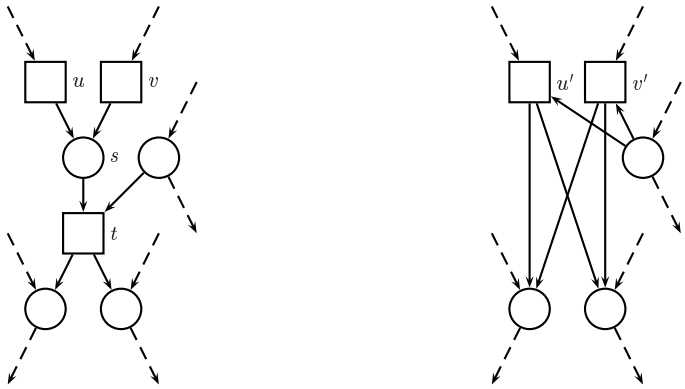
folgt. Für das Ausführen der Schaltfolge $M|_{S_h} \xrightarrow{\sigma_h}$ in N_h wird keine Marke von der Stelle $s \in t^\bullet$ verwendet. (Andernfalls müßte eine Transition $u \in s^\bullet$ in der Schaltfolge σ_h vorkommen, was aber im Widerspruch zu Proposition 5.2.7 auf Seite 144 steht, es sei denn, daß

$$\exists u \in s^\bullet \colon \forall s' \in {}^\bullet t \cap {}^\bullet u \colon s' \notin S_h.$$

Aus $s' \in {}^\bullet u$ und $s' \notin S_h$ folgt $u \notin T_h$, was jedoch im Widerspruch zu der Annahme steht, daß u in σ_h vorkommt). Mit Gleichung 5.2.1 folgt daher unmittelbar

$$\sigma_h \in L^\omega((N_h, M_3|_{S_h})) \quad \text{und} \quad \sigma_h \in L^\omega((N^r_h, M_3|_{S^r_h})).$$

Abbildung 5.9 *Teilnetz eines sicheren S/T-Netzsystems, für das die Stelle s und die Transition t der Abstraktion genügen (a), und das reduzierte Teilnetz nach durchgeführter Abstraktion (b)*



(a) (b)

(2) Sei t in $\sigma_{\neg\varphi}$ enthalten. Dann müßte $t \in T_B$ gelten, was jedoch im Widerspruch zu $t \in T_H$ steht.

(3) Sei t in σ_h enthalten, d.h., $\sigma_h = \sigma_3 t \sigma_4$. Mit Proposition 5.2.8 auf Seite 145 folgt aus $M|_{S_h} \xrightarrow{\sigma_3 t \sigma_4}$ unmittelbar die Existenz der Schaltfolge $M|_{S_h} \xrightarrow{\sigma_3 \sigma_4}$. Somit gilt

$$\sigma_3 \sigma_4 \in L^\omega((N_h^r, M|_{S_h^r})).$$

($\Leftarrow$) Trivial, da jede Schaltfolge σ in $\Sigma^r_{S\times\neg\varphi}$, $(N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}})$ oder $(N_h^r, M|_{S_h^r})$ auch unmittelbar in $\Sigma_{S\times\neg\varphi}$, $(N_{\neg\varphi}, M|_{S_{\neg\varphi}})$ oder $(N_h, M|_{S_h})$ vorhanden ist.

■

5.2.5 Abstraktion

In diesem Abschnitt wird eine Abstraktionsregel vorgestellt, die erstmals in [ES01a] erwähnt wurde. Die Idee der Abstraktion, welche in Abbildung 5.9 (a) auf dieser Seite dargestellt ist, besteht darin, daß jede Schaltfolge des Netzsystems, in der die Transition t vorkommt, in eine entsprechende Schaltfolge überführt werden kann, sodaß unmittelbar vor t eine der Eingangstransitionen von s schaltet. Die Abstraktion „versteckt“ das Schalten von t, indem t mit den Eingangstransitionen von s verschmolzen wird (Abbildung 5.9 (b) auf dieser Seite). Dabei muß allerdings sichergestellt werden, daß die Eingangstransitionen von s in dem reduzierten Netzsystem nicht aktiviert sind, bevor t im ursprünglichen Netzsystem aktiviert ist. Dieses kann dadurch erreicht werden, daß deren Vorbereiche um die Eingangsstellen von t erweitert werden. In [DE95] ist eine

Abstraktionsregel für Free-Choice-Netze[5] vorgeschlagen worden, jedoch mit der zusätzlichen Bedingung

$$^\bullet t = \{s\}.$$

Diese Einschränkung kann jedoch für die Einhaltung der Bedingungen von Satz 5.1.3 auf Seite 126 weggelassen werden, weshalb die hier betrachtete Abstraktion eine Verallgemeinerung der in [DE95] vorgestellten Abstraktion darstellt.

Definition 5.2.8 (Abstraktion [ES01a])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Stelle $s \in S$ und eine Transition $t \in s^\bullet \setminus {}^\bullet s$ genügen der Abstraktion, falls gilt:

$$\begin{aligned} s^\bullet &= \{t\} \\ {}^\bullet s &\neq \emptyset \\ ({}^\bullet s)^\bullet &= \{s\} \\ M_0(s) &= 0 \end{aligned}$$

Das durch Anwendung der Abstraktion erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r = (S^r, T^r, F^r)$ ist definiert als:

$$\begin{aligned} S^r &= S \setminus \{s\} \\ T^r &= T \setminus (\{t\} \cup \{u \in {}^\bullet s \mid {}^\bullet u \cap {}^\bullet t \neq \emptyset\}) \\ F^r &= (F \cap ((S^r \times T^r) \cup (T^r \times S^r))) \cup \\ &\quad (({}^\bullet s \cap T^r) \times (t^\bullet \cap S^r)) \cup \\ &\quad (({}^\bullet t \cap S^r) \times ({}^\bullet s \cap T^r)) \\ M_0^r &= M_0|_{S^r} \end{aligned}$$

Die Transitionen der Menge

$$\{u \in {}^\bullet s \mid {}^\bullet u \cap {}^\bullet t \neq \emptyset\}$$

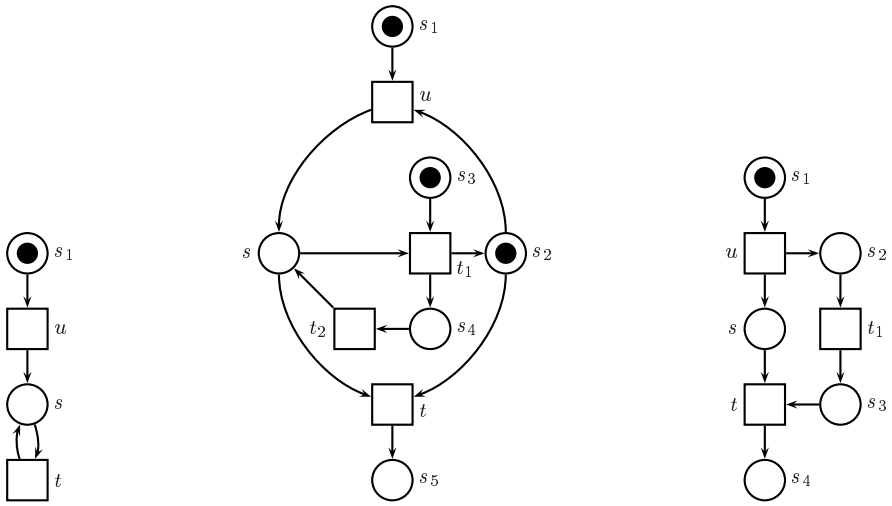
sind nach Definition 5.2.7 auf Seite 144 bedeutungslos. Daher kann es nach Proposition 5.2.7 auf Seite 144 keine Schaltfolge

$$M_0 \xrightarrow{\sigma_1 u \sigma_2 t} M$$

geben, wobei u eine bedeutungslose Transition bezeichnet. Folglich darf in diesem Fall keine Abstraktion vorgenommen werden, bevor nicht die bedeutungslosen Transitionen aus dem Netzsystem entfernt wurden.
Im folgenden soll näher auf die Notwendigkeit der einzelnen Bedingungen von Definition 5.2.8 eingegangen werden, sofern sie nicht offensichtlich ist.
Die Bedingung $t \notin {}^\bullet s$ wird gefordert, damit die in Abbildung 5.10 (a) auf der nächsten Seite dargestellte Situation nicht eintreten kann. Die Stelle s und die Transition t

[5] Ein Netz $N = (S, T, F)$ heißt *free-choice*, wenn ${}^\bullet t \times s^\bullet \subseteq F$ für jedes $s \in S$ und $t \in T$ mit $(s, t) \in F$ [DE95].

Abbildung 5.10 *Sicheres S/T-Netzsystem, für das die Stelle s und die Transition t der Abstraktion genügen würden, falls die Bedingung $t \in {}^\bullet s$ zulässig wäre (a), falls die Bedingung $|s^\bullet| > 1$ zulässig wäre (b), falls die Bedingung $|({}^\bullet s)^\bullet| > 1$ zulässig wäre (c)*



(a) (b) (c)

würden den Abstraktionsbedingungen genügen, falls $t \in {}^\bullet s$ zulässig wäre. Das Netzsystem enthält die unendliche Schaltfolge

$$M_0 \xrightarrow{ut^\omega},$$

die auch in dem reduzierten Netzsystem repräsentiert sein müßte. Eine Abstraktion nach Definition 5.2.8 auf der vorherigen Seite könnte nicht vorgenommen werden.
Die Bedingung $s^\bullet = \{t\}$ wird gefordert, damit die in Abbildung 5.10 (b) dargestellte Situation nicht eintreten kann. Die Stelle s und die Transition t würden den Abstraktionsbedingungen genügen, falls $|s^\bullet| > 1$ zulässig wäre. Das Netzsystem enthält als einzige Schaltfolge

$$M_0 = \{s_1, s_2, s_3\} \xrightarrow{ut_1t_2t} \{s_5\}.$$

In diesem Beispiel ist die Transition $u \in {}^\bullet s$ trotz ${}^\bullet u \cap {}^\bullet t \neq \emptyset$ (aber wegen ${}^\bullet u \cap {}^\bullet t_1 = \emptyset$) nicht bedeutungslos und dürfte nicht aus dem Netz entfernt werden. Die Schaltfolge ut_1t_2t kann auch nicht derart umgeordnet werden, sodaß u unmittelbar vor t schaltet. Deshalb darf in diesem Fall keine Abstraktion vorgenommen werden.
Die Bedingung $({}^\bullet s)^\bullet = \{s\}$ wird gefordert, damit die in Abbildung 5.10 (c) dargestellte Situation nicht eintreten kann. Die Stelle s und die Transition t würden den Abstraktionsbedingungen genügen, falls $|({}^\bullet s)^\bullet| > 1$ zulässig wäre. Das Netzsystem enthält als einzige Schaltfolge

$$M_0 = \{s_1\} \xrightarrow{ut_1t} \{s_4\},$$

die auch nicht derart umgeordnet werden kann, sodaß u unmittelbar vor t schaltet. Deshalb darf in diesem Fall keine Abstraktion vorgenommen werden.

Proposition 5.2.9 (Abstraktion)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Stelle $s \in S$ und eine Transition $t \in s^\bullet \setminus {}^\bullet s$, die der Abstraktion (Definition 5.2.8 auf Seite 149) genügen, sowie eine Transition $u \in {}^\bullet s$. Es gilt:

$$M_0 \xrightarrow{\sigma_1 u \sigma_2 t} M \qquad \Rightarrow \qquad M_0 \xrightarrow{\sigma_1 \sigma_2 u t} M$$

Beweis: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Stelle $s \in S$ und eine Transition $t \in s^\bullet \setminus {}^\bullet s$, die der Abstraktion (Definition 5.2.8 auf Seite 149) genügen, sowie eine Transition $u \in {}^\bullet s$. Angenommen, es gibt eine Schaltfolge

$$M_0 \xrightarrow{\sigma_1 u \sigma_2 t} M,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß t nicht in σ_2 vorkommt. Daraus folgt die Existenz von erreichbaren Markierungen $M_1, M_2, M_3 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{u} M_2 \xrightarrow{\sigma_2} M_3 \xrightarrow{t} M.$$

Aus $s \in u^\bullet$ und $s \in {}^\bullet t$ folgt unmittelbar

$$M_2(s) = M_3(s) = 1.$$

Die Marke auf der Stelle s wird für das Schalten von $M_2 \xrightarrow{\sigma_2} M_3$ nicht benötigt.
(Ansonsten müßte die Transition $t \in s^\bullet$ in der Schaltfolge σ_2 vorkommen, was jedoch einen Widerspruch zu der Annahme ergäbe, daß dieses nicht der Fall ist).
Mit der Beziehung

$$\forall s' \in S \setminus \{s\} : M_1(s') \geq M_2(s')$$

folgt also unmittelbar

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4.$$

Weiterhin gilt

$$\forall s' \in {}^\bullet u : M_1(s') = 1.$$

Für das Schalten von $M_1 \xrightarrow{\sigma_2} M_4$ werden keine Marken von den Stellen aus ${}^\bullet u$ benötigt.
(Ansonsten müßte σ_2 wegen

$$\forall s' \in {}^\bullet u : M_2(s') = 0$$

erstmal neue Marken auf den Stellen aus ${}^\bullet u$ erzeugen. Daraus folgt, daß es mindestens eine Stelle $s' \in {}^\bullet u$ und eine Transition $t' \in {}^\bullet s' \setminus s'^\bullet$ geben müßte, sodaß eine Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma' t'} M'$$

existiert, wobei ohne Einschränkung der Allgemeinheit angenommen werden könnte, daß keine Transition aus ${}^\bullet s' \cup s'^\bullet$ in σ' vorkommt. In diesem Fall würde dann $M'(s') = 2$

gelten, was jedoch einen Widerspruch zur Sicherheit des Netzsystems ergäbe).
Folglich gilt auch

$$\forall s' \in {}^\bullet u\colon M_4(s') = 1$$

und somit

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4 \xrightarrow{u} M_3.$$

Die Transition t ist unter der Markierung M_3 aktiviert, woraus sich die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4 \xrightarrow{u} M_3 \xrightarrow{t} M$$

ergibt. ■

Satz 5.2.5 (Abstraktion bewahrt LTL-X-Eigenschaften [ES01a])
Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ und das nach Definition 5.2.8 auf Seite 149 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi}$. Die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 gelten genau dann in $\Sigma_{S\times\neg\varphi}$, wenn sie in $\Sigma^r_{S\times\neg\varphi}$ gelten.

Beweis: Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi} = (N_\mathcal{B}, M_0)$ mit $N_\mathcal{B} = (S, T, F, S_E)$, eine Stelle $s \in S$ und eine Transition $t \in s^\bullet \setminus {}^\bullet s$, die der Abstraktion (Definition 5.2.8 auf Seite 149) genügen, sowie das nach Definition 5.2.8 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi} = (N^r_\mathcal{B}, M^r_0)$ mit $N^r_\mathcal{B} = (S^r, T^r, F^r, S^r_E)$. Aus $({}^\bullet s)^\bullet = \{s\}$ und Proposition 5.1.1 auf Seite 122 folgt ${}^\bullet s \subseteq T_H$. Somit gilt $s \in S_H$ und folglich $t \notin T_B$. Desweiteren gilt wegen $t^\bullet \not\subset {}^\bullet t$ und Proposition 5.1.1 auf Seite 122 auch $t \notin T_L$.

(i) ($\Rightarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. Falls t in σ vorkommt, muß wegen $s \in {}^\bullet t$ und $M_0(s) = 0$ eine Transition $u \in {}^\bullet s \setminus s^\bullet$ vor t in σ vorkommen. Deshalb gibt es eine Zerlegung von σ in

$$M_0 \xrightarrow{\sigma_1 u \sigma_2 t \sigma_3},$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß t weder in σ_1 noch in σ_2 vorkommt. Mit Proposition 5.2.9 auf der vorherigen Seite folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1 \sigma_2 u t \sigma_3}.$$

Das bedeutet, daß erreichbare Markierungen $M_1, M_2, M_3 \in \mathcal{R}(M_0)$ existieren, sodaß

$$M_0 \xrightarrow{\sigma_1 \sigma_2} M_1 \xrightarrow{u} M_2 \xrightarrow{t} M_3 \xrightarrow{\sigma_3}.$$

Aus der Definition von $N^r_\mathcal{B}$ (Definition 5.2.8 auf Seite 149) folgt die Existenz einer Transition $u_t \in T^r$ mit

$${}^\bullet u_t = {}^\bullet u \cup ({}^\bullet t \setminus \{s\}) \quad \text{und} \quad u_t^\bullet = t^\bullet.$$

Weiterhin gilt

$$\forall s' \in {}^\bullet t \colon M_2(s') = 1,$$

und wegen $u^\bullet = \{s\}$ folgt unmittelbar

$$\forall s' \in {}^\bullet t \setminus \{s\} \colon M_1(s') = 1.$$

Zusammen mit

$$\forall s' \in {}^\bullet u \colon M_1(s') = 1$$

ergibt sich

$$\forall s' \in {}^\bullet u_t \colon M_1(s') = 1.$$

Daraus folgt jedoch die Existenz der Schaltfolge

$$M_0^r \xrightarrow{\sigma_1\sigma_2} M_1|_{S\setminus\{s\}} \xrightarrow{u_t} M_3|_{S\setminus\{s\}}$$

in $\Sigma^r_{S\times\neg\varphi}$. Wegen $t \in s^\bullet \setminus {}^\bullet s$ gilt $M_3(s) = 0$, woraus unmittelbar

$$M_0^r \xrightarrow{\sigma_1\sigma_2} M_1|_{S\setminus\{s\}} \xrightarrow{u_t} M_3|_{S\setminus\{s\}} \xrightarrow{\sigma_3}$$

in $\Sigma^r_{S\times\neg\varphi}$ abgeleitet werden kann. (Sollte t erneut in σ_3 vorkommen, kann wiederum wie eben beschrieben verfahren werden). Wegen $u \in T_H$ und $t \notin T_B$ enthält die Schaltfolge auch weiterhin unendlich viele unendliche Trace-Wächter (Transitionen aus T_I).

($\Leftarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma^r_{S\times\neg\varphi}$, d.h., es existiert eine unendliche Schaltfolge $M_0^r \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. Falls eine durch Abstraktion entstandene Transition u_t in σ vorkommt, gibt es eine Zerlegung von σ in

$$M_0^r \xrightarrow{\sigma_1 u_t \sigma_2},$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß u_t nicht in σ_1 vorkommt. Das bedeutet, daß erreichbare Markierungen $M_1, M_2, \in \mathcal{R}(M_0^r)$ existieren, sodaß

$$M_0^r \xrightarrow{\sigma_1} M_1 \xrightarrow{u_t} M_2 \xrightarrow{\sigma_2} .$$

Daraus folgt nach Definition 5.2.8 auf Seite 149 die Existenz einer Stelle $s \in S$ sowie zweier Transitionen $u \in {}^\bullet s$ und $t \in s^\bullet \setminus {}^\bullet s$, sodaß

$$\begin{aligned} {}^\bullet u_t &= {}^\bullet u \cup ({}^\bullet t \setminus \{s\}) \\ u_t^\bullet &= t^\bullet \\ u^\bullet &= \{s\} \\ s^\bullet &= \{t\} \\ {}^\bullet u \cap {}^\bullet t &= \emptyset \end{aligned}$$

Daraus folgt unmittelbar

$$\forall s' \in {}^\bullet u \cup ({}^\bullet t \setminus \{s\})\colon M_1(s') = 1. \tag{5.2.2}$$

Da u_t laut Annahme nicht in σ_1 vorkommt, existiert in $\Sigma_{S\times\neg\varphi}$ die Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1',$$

wobei

$$\forall s' \in S \setminus \{s\}\colon M_1'(s') = M_1(s').$$

Wegen $s \notin {}^\bullet u$ ergibt sich dann unmittelbar mit Gleichung 5.2.2

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{u} M_3.$$

Wegen $s \in u^\bullet$ gilt $M_3(s) = 1$ und zusammen mit ${}^\bullet u \cap {}^\bullet t = \emptyset$ und Gleichung 5.2.2 folgt

$$\forall s' \in {}^\bullet t\colon M_3(s') = 1.$$

Daraus ergibt sich die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{u} M_3 \xrightarrow{t} M_2',$$

wobei

$$\forall s' \in S \setminus \{s\}\colon M_2'(s') = M_2(s').$$

Daraus folgt, daß in $\Sigma_{S\times\neg\varphi}$ die unendliche Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{u} M_3 \xrightarrow{t} M_2' \xrightarrow{\sigma_2}$$

existiert, in der wegen $u_t \notin T_I$ auch weiterhin unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. (Sollte u_t erneut in σ_2 vorkommen, kann wiederum wie eben beschrieben verfahren werden).

(ii) ($\Rightarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine Schaltfolge $M_0 \xrightarrow{\sigma} M$, sodaß $L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})) \neq \emptyset$ und $L^\omega((N_h, M|_{S_h})) \neq \emptyset$. Seien also $\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M|_{S_{\neg\varphi}}))$ und $\sigma_h \in L^\omega((N_h, M|_{S_h}))$ gegeben.

(1) Sei t in σ enthalten. Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar eine Zerlegung von σ in $\sigma_1 u \sigma_2 t \sigma_3$ mit $u \in {}^\bullet s \setminus s^\bullet$ und die Existenz einer aus der Abstraktion resultierenden Schaltfolge

$$M_0^r \xrightarrow{\sigma_1\sigma_2 u_t \sigma_3} M|_{S\setminus\{s\}}$$

in $\Sigma^r_{S\times\neg\varphi}$, wobei $u_t \in T^r$ eine Transition bezeichnet, die durch Abstraktion von s und t entstanden ist. Wegen $s \in S_H$ gilt $S_{\neg\varphi} \subseteq (S \setminus \{s\})$, und mit $S_{\neg\varphi} = S^r_{\neg\varphi}$ folgt unmittelbar

$$\sigma_{\neg\varphi} \in L((N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}})).$$

Ohne Einschränkung der Allgemeinheit kann angenommen werden, daß t nicht in σ_h vorkommt. (Der Fall, daß t in σ_h vorkommt, wird in (3) betrachtet). Wegen $s^\bullet = \{t\}$ folgt daraus, daß keine Marke von s für das Ausführen von σ_h benötigt wird, und somit gilt

$$\sigma_h \in L^\omega((N_h^r, M|_{S_h^r})).$$

(2) Sei t in $\sigma_{\neg\varphi}$ enthalten. Dann müßte $t \in T_B$ gelten, was jedoch im Widerspruch zu $t \notin T_B$ steht.

(3) Sei t in σ_h enthalten. Dann müssen zwei Fälle betrachtet werden.

(a) Sei $u \in {}^\bullet s \setminus s^\bullet$ in σ_h enthalten. Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar eine Zerlegung von σ_h in $\sigma_{h_1} u \sigma_{h_2} t \sigma_{h_3}$ und die Existenz einer aus der Abstraktion resultierenden Schaltfolge

$$\sigma_{h_1}\sigma_{h_2} u_t \sigma_{h_3} \in L^\omega((N_h^r, M|_{S_h^r})),$$

wobei $u_t \in T^r$ eine Transition bezeichnet, die durch Abstraktion von s und t entstanden ist.

(b) Sei $u \in {}^\bullet s \setminus s^\bullet$ in σ enthalten. Dann gibt es Zerlegungen von σ und σ_h in $\sigma_1 u \sigma_2$ und $\sigma_{h_1} t \sigma_{h_2}$. Da N_h ein Teilnetz von $N_\mathcal{B}$ darstellt, ist jede Schaltfolge von $(N_h, M|_{S_h})$ auch in $(N_\mathcal{B}, M)$ vorhanden. Deshalb existiert die Schaltfolge

$$M_0 \xrightarrow{\sigma_1 u \sigma_2} M \xrightarrow{\sigma_{h_1} t \sigma_{h_2}}$$

in $\Sigma_{S\times\neg\varphi}$. Aus Proposition 5.2.9 auf Seite 151 folgt unmittelbar

$$M_0 \xrightarrow{\sigma_1\sigma_2} M_1 \xrightarrow{\sigma_{h_1} u t \sigma_{h_2}},$$

wobei

$$\forall s' \in S \setminus \{s\} : M_1(s') \geq M(s'). \tag{5.2.3}$$

Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar

$$M_0^r \xrightarrow{\sigma_1\sigma_2} M_1|_{S\setminus\{s\}}$$

und die Existenz einer aus der Abstraktion resultierenden Schaltfolge

$$\sigma_{h_1} u_t \sigma_{h_2} \in L^\omega((N_h^r, M_1|_{S_h^r})),$$

wobei $u_t \in T^r$ eine Transition bezeichnet, die durch Abstraktion von s und t entstanden ist. Wegen $t \notin T_B$ kann t nicht in $\sigma_{\neg\varphi}$ vorkommen. Folglich werden für das Ausführen von $\sigma_{\neg\varphi}$ keine Marken von s benötigt. Mit Gleichung 5.2.3 folgt unmittelbar

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}^r, M_1|_{S_{\neg\varphi}^r})).$$

($\Leftarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma^r_{S\times\neg\varphi}$, d.h., es existiert eine Schaltfolge $M_0^r \xrightarrow{\sigma} M$, sodaß $L((N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}})) \neq \emptyset$ und $L^\omega((N^r_h, M|_{S^r_h})) \neq \emptyset$. Seien also $\sigma_{\neg\varphi} \in L((N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}}))$ und $\sigma_h \in L^\omega((N^r_h, M|_{S^r_h}))$ gegeben, und $u_t \in T^r$ bezeichne eine Transition, die durch Abstraktion von s und t entstanden ist.

(1) Sei u_t in σ enthalten. Dann gibt es eine Zerlegung von σ in $\sigma_1 u_t \sigma_2$. Aus Beweisrichtung (i) ($\Leftarrow$) folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1 u t \sigma_2} M'$$

in $\Sigma_{S\times\neg\varphi}$, wobei

$$\forall s' \in S \setminus \{s\} : M'(s') = M(s'). \tag{5.2.4}$$

Folglich gilt auch

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M'|_{S_{\neg\varphi}})) \quad \text{und} \quad \sigma_h \in L^\omega((N_h, M'|_{S_h})).$$

(2) Sei u_t in $\sigma_{\neg\varphi}$ enthalten. Dann müßte auch t in $\sigma_{\neg\varphi}$ vorkommen und somit $t \in T_B$ gelten, was jedoch im Widerspruch zu $t \notin T_B$ steht.

(3) Sei u_t in σ_h enthalten. Dann gibt es eine Zerlegung von σ_h in $\sigma_{h_1} u_t \sigma_{h_2}$. Aus Beweisrichtung (i) ($\Leftarrow$) folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma} M'$$

in $\Sigma_{S\times\neg\varphi}$, wobei auch hier Gleichung 5.2.4 gilt, sowie

$$\sigma_{h_1} u t \sigma_{h_2} \in L^\omega((N_h, M'|_{S_h})).$$

Aus Gleichung 5.2.4 folgt dann auch

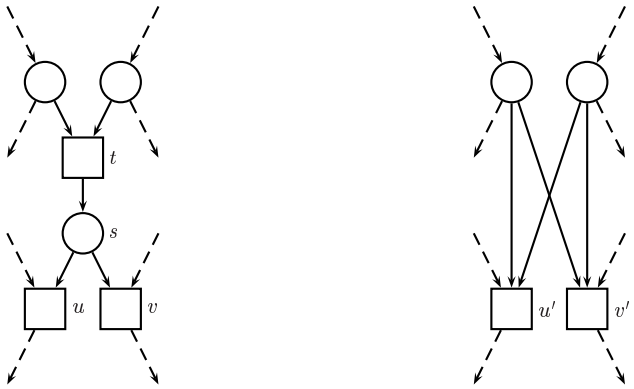
$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M'|_{S_{\neg\varphi}})).$$

■

5.2.6 Pre-Agglomeration

In diesem Abschnitt wird eine Pre-Agglomeration vorgestellt, die in dieser Form erstmals in [ES01a] erwähnt wurde. Die Idee der Pre-Agglomeration, welche in Abbildung 5.11 (a) auf der nächsten Seite dargestellt ist, besteht darin, daß jede Schaltfolge des Netzsystems, in der die Transition t und eine der Ausgangstransitionen von s vorkommen, in eine entsprechende Schaltfolge überführt werden kann, sodaß t unmittelbar vor einer Transition aus $s^\bullet$ schaltet. Die Pre-Agglomeration impliziert, daß ein Schalten von t „verzögert“ werden kann, und „versteckt“ das Schalten von t, indem t mit den Ausgangstransitionen von s verschmolzen wird (Abbildung 5.11 (b) auf der nächsten Seite).

Abbildung 5.11 *Teilnetz eines sicheren S/T-Netzsystems, für das die Stelle s und die Transition t der Pre-Agglomeration genügen (a), und das reduzierte Teilnetz nach durchgeführter Pre-Agglomeration (b)*



(a) (b)

In [Ber85, PPP00] ist ebenfalls eine Form der Pre-Agglomeration betrachtet worden, jedoch mit der zusätzlichen Bedingung

$$({}^\bullet t)^\bullet = \{t\}.$$

Diese Einschränkung kann jedoch für die Einhaltung der Bedingungen von Satz 5.1.3 auf Seite 126 weggelassen werden, weshalb die hier betrachtete Pre-Agglomeration eine Verallgemeinerung der in [Ber85, PPP00] eingeführten Pre-Agglomeration darstellt.

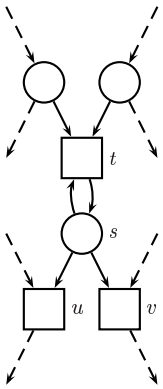
Definition 5.2.9 (Pre-Agglomeration [ES01a])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Stelle $s \in S$ und eine Transition $t \in {}^\bullet s \setminus s^\bullet$ genügen der Pre-Agglomeration, falls gilt:

$$\begin{aligned} {}^\bullet s &= \{t\} \\ s^\bullet &\neq \emptyset \\ t^\bullet &= \{s\} \\ M_0(s) &= 0 \end{aligned}$$

Das durch Anwendung der Pre-Agglomeration erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r = (S^r, T^r, F^r)$ ist definiert als:

$$\begin{aligned} S^r &= S \setminus \{s\} \\ T^r &= T \setminus (\{t\} \cup \{u \in s^\bullet \mid {}^\bullet u \cap {}^\bullet t \neq \emptyset\}) \\ F^r &= (F \cap ((S^r \times T^r) \cup (T^r \times S^r))) \cup (({}^\bullet t \cap S^r) \times (s^\bullet \cap T^r)) \\ M_0^r &= M_0|_{S^r} \end{aligned}$$

Abbildung 5.12 *Teilnetz eines sicheren S/T-Netzsystems, für das die Stelle s und die Transition t der Pre-Agglomeration genügen würden, falls die Bedingung $t \in s^\bullet$ zulässig wäre*



Die Transitionen der Menge

$$\{u \in s^\bullet \mid {}^\bullet u \cap {}^\bullet t \neq \emptyset\}$$

genügen der Regel TT3 (Definition 5.2.6 auf Seite 140) und sind somit nach Proposition 5.2.6 auf Seite 141 tot. Folglich müssen diese Transitionen bei Durchführung der Pre-Agglomeration aus dem Netzsystem entfernt werden.
Im folgenden soll näher auf die Notwendigkeit der einzelnen Bedingungen von Definition 5.2.9 auf der vorherigen Seite eingegangen werden, sofern sie nicht offensichtlich ist. Die Bedingung $t \notin s^\bullet$ wird gefordert, damit die in Abbildung 5.12 dargestellte Situation nicht eintreten kann. In diesem Fall genügt die Transition t der Regel $TT2$ (Definition 5.2.5 auf Seite 139) und ist nach Proposition 5.2.5 auf Seite 139 tot. Somit darf keine Pre-Agglomeration durchgeführt werden, jedoch können t und die übrigen Transitionen aus $s^\bullet$ nach Regel TT2 aus dem Netzsystem entfernt werden.
Die Bedingung ${}^\bullet s = \{t\}$ wird gefordert, damit die in Abbildung 5.10 (b) auf Seite 150 dargestellte Situation nicht eintreten kann. In diesem Fall würden die Stelle s und die Transition u der Pre-Agglomeration genügen, falls die Bedingung $|{}^\bullet s| > 1$ zulässig wäre. Die Transition $t \in s^\bullet$ ist jetzt trotz ${}^\bullet t \cap {}^\bullet u \neq \emptyset$ (aber wegen $|{}^\bullet s| > 1$) nicht tot und darf nicht aus dem Netz entfernt werden. Die einzige Schaltfolge des Netzsystems

$$M_0 \xrightarrow{ut_1t_2t} M$$

kann nicht derart umgeordnet werden, sodaß u unmittelbar vor t schaltet. Deshalb kann in diesem Fall keine Pre-Agglomeration vorgenommen werden.
Die Bedingung $t^\bullet = \{s\}$ wird gefordert, damit die in Abbildung 5.10 (c) auf Seite 150 dargestellte Situation nicht eintreten kann. In diesem Fall würden die Stelle s und die Transition u der Pre-Agglomeration genügen, falls die Bedingung $|u^\bullet| > 1$ zulässig wäre. Die einzige Schaltfolge des Netzsystems

$$M_0 \xrightarrow{ut_1t} M$$

kann nicht derart umgeordnet werden, sodaß u unmittelbar vor t schaltet. Deshalb kann in diesem Fall keine Pre-Agglomeration vorgenommen werden.

Proposition 5.2.10 (Pre-Agglomeration)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Stelle $s \in S$ und eine Transition $t \in {}^\bullet s \setminus s^\bullet$, die der Pre-Agglomeration (Definition 5.2.9 auf Seite 157) genügen, sowie eine Transition $u \in s^\bullet$. Es gilt:

$$M_0 \xrightarrow{\sigma_1 t \sigma_2 u} M \qquad \Rightarrow \qquad M_0 \xrightarrow{\sigma_1 \sigma_2 t u} M$$

Beweis: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Stelle $s \in S$ und eine Transition $t \in {}^\bullet s \setminus s^\bullet$, die der Pre-Agglomeration (Definition 5.2.9 auf Seite 157) genügen, sowie eine Transition $u \in s^\bullet$.
Angenommen, es gibt eine Schaltfolge

$$M_0 \xrightarrow{\sigma_1 t \sigma_2 u} M,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß keine Transition aus $s^\bullet$ in σ_2 vorkommt. Daraus folgt die Existenz von erreichbaren Markierungen $M_1, M_2, M_3 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t} M_2 \xrightarrow{\sigma_2} M_3 \xrightarrow{u} M.$$

Aus $s \in t^\bullet$ und $s \in {}^\bullet u$ folgt unmittelbar

$$M_2(s) = M_3(s) = 1.$$

Die Marke auf der Stelle s wird für das Schalten von $M_2 \xrightarrow{\sigma_2} M_3$ nicht benötigt.
(Ansonsten müßte eine Transition aus $s^\bullet$ in der Schaltfolge σ_2 vorkommen, was jedoch einen Widerspruch zu der Annahme ergäbe, daß dieses nicht der Fall ist).
Mit der Beziehung

$$\forall s' \in S \setminus \{s\} \colon M_1(s') \geq M_2(s')$$

folgt also unmittelbar

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4.$$

Weiterhin gilt

$$\forall s' \in {}^\bullet t \colon M_1(s') = 1.$$

Für das Schalten von $M_1 \xrightarrow{\sigma_2} M_4$ werden keine Marken von den Stellen aus ${}^\bullet t$ benötigt.
(Ansonsten müßte σ_2 wegen

$$\forall s' \in {}^\bullet t \colon M_2(s') = 0$$

erstmal neue Marken auf den Stellen aus ${}^\bullet t$ erzeugen. Daraus folgt, daß es mindestens eine Stelle $s' \in {}^\bullet t$ und eine Transition $t' \in {}^\bullet s' \setminus s'^\bullet$ geben müßte, sodaß eine Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma' t'} M'$$

existiert, wobei ohne Einschränkung der Allgemeinheit angenommen werden könnte, daß keine Transition aus ${}^\bullet s' \cup s'^\bullet$ in σ' vorkommt. In diesem Fall würde dann $M'(s') = 2$ gelten, was jedoch einen Widerspruch zur Sicherheit des Netzsystems ergäbe).
Folglich gilt auch

$$\forall s' \in {}^\bullet t\colon M_4(s') = 1$$

und somit

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4 \xrightarrow{t} M_3.$$

Die Transition u ist unter der Markierung M_3 aktiviert, woraus sich die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_4 \xrightarrow{t} M_3 \xrightarrow{u} M$$

ergibt. ■

Satz 5.2.6 (Pre-Agglomeration bewahrt LTL-X-Eigenschaften [ES01a])
Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ und das nach Definition 5.2.9 auf Seite 157 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi}$. Die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 gelten genau dann in $\Sigma_{S\times\neg\varphi}$, wenn sie in $\Sigma^r_{S\times\neg\varphi}$ gelten.

Beweis: Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi} = (N_\mathcal{B}, M_0)$ mit $N_\mathcal{B} = (S, T, F, S_E)$, eine Stelle $s \in S$ und eine Transition $t \in {}^\bullet s \setminus s^\bullet$, die der Pre-Agglomeration (Definition 5.2.9 auf Seite 157) genügen, sowie das nach Definition 5.2.9 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi} = (N^r_\mathcal{B}, M^r_0)$ mit $N^r_\mathcal{B} = (S^r, T^r, F^r, S^r_E)$. Aus $t^\bullet = \{s\}$ und Proposition 5.1.1 auf Seite 122 folgt $t \in T_H$. Somit gilt $s \in S_H$ und folglich $s^\bullet \not\subseteq T_B$. Wegen $s \notin (s^\bullet)^\bullet$ folgt mit Proposition 5.1.1 auf Seite 122 auch $s^\bullet \not\subseteq T_L$.

(i) ($\Rightarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. Falls eine Transition $u \in s^\bullet$ in σ vorkommt, muß wegen $s \in {}^\bullet u$ und $M_0(s) = 0$ die Transition t vor u in σ vorkommen. Deshalb gibt es eine Zerlegung von σ in

$$M_0 \xrightarrow{\sigma_1 t \sigma_2 u \sigma_3},$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß keine Transition aus $s^\bullet$ weder in σ_1 noch in σ_2 vorkommt. Mit Proposition 5.2.10 auf der vorherigen Seite folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1 \sigma_2 t u \sigma_3}.$$

Das bedeutet, daß erreichbare Markierungen $M_1, M_2, M_3 \in \mathcal{R}(M_0)$ existieren, sodaß

$$M_0 \xrightarrow{\sigma_1 \sigma_2} M_1 \xrightarrow{t} M_2 \xrightarrow{u} M_3 \xrightarrow{\sigma_3}.$$

Aus der Definition von $N^r_\mathcal{B}$ (Definition 5.2.9 auf Seite 157) folgt die Existenz einer Transition $u_t \in T^r$ mit

$${}^\bullet u_t = ({}^\bullet u \setminus \{s\}) \cup {}^\bullet t \quad \text{und} \quad u_t^\bullet = u^\bullet.$$

Weiterhin gilt

$$\forall s' \in {}^\bullet u \colon M_2(s') = 1,$$

und wegen $t^\bullet = \{s\}$ folgt unmittelbar

$$\forall s' \in {}^\bullet u \setminus \{s\} \colon M_1(s') = 1.$$

Zusammen mit

$$\forall s' \in {}^\bullet t \colon M_1(s') = 1$$

ergibt sich

$$\forall s' \in {}^\bullet u_t \colon M_1(s') = 1.$$

Daraus folgt jedoch die Existenz der Schaltfolge

$$M_0^r \xrightarrow{\sigma_1\sigma_2} M_1|_{S\setminus\{s\}} \xrightarrow{u_t} M_3|_{S\setminus\{s\}}$$

in $\Sigma^r_{S\times\neg\varphi}$. Wegen $u \in s^\bullet \setminus {}^\bullet s$ gilt $M_3(s) = 0$, woraus unmittelbar

$$M_0^r \xrightarrow{\sigma_1\sigma_2} M_1|_{S\setminus\{s\}} \xrightarrow{u_t} M_3|_{S\setminus\{s\}} \xrightarrow{\sigma_3}$$

in $\Sigma^r_{S\times\neg\varphi}$ abgeleitet werden kann. (Sollte erneut eine Transition aus $s^\bullet$ in σ_3 vorkommen, kann wiederum wie eben beschrieben verfahren werden). Wegen $t \in T_H$ und $u \notin T_B$ enthält die Schaltfolge auch weiterhin unendlich viele unendliche Trace-Wächter (Transitionen aus T_I).

($\Leftarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma^r_{S\times\neg\varphi}$, d.h., es existiert eine unendliche Schaltfolge $M_0^r \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. Falls eine durch Pre-Agglomeration entstandene Transition u_t in σ vorkommt, gibt es eine Zerlegung von σ in

$$M_0^r \xrightarrow{\sigma_1 u_t \sigma_2},$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß u_t nicht in σ_1 vorkommt. Das bedeutet, daß erreichbare Markierungen $M_1, M_2, \in \mathcal{R}(M_0^r)$ existieren, sodaß

$$M_0^r \xrightarrow{\sigma_1} M_1 \xrightarrow{u_t} M_2 \xrightarrow{\sigma_2} .$$

Daraus folgt nach Definition 5.2.9 auf Seite 157 die Existenz einer Stelle $s \in S$ sowie zweier Transitionen $t \in {}^\bullet s \setminus s^\bullet$ und $u \in s^\bullet$, sodaß

$$\begin{aligned} {}^\bullet u_t &= ({}^\bullet u \setminus \{s\}) \cup {}^\bullet t \\ u_t^\bullet &= u^\bullet \\ t^\bullet &= \{s\} \\ {}^\bullet s &= \{t\} \\ {}^\bullet u \cap {}^\bullet t &= \emptyset \end{aligned}$$

Daraus folgt unmittelbar

$$\forall s' \in ({}^\bullet u \setminus \{s\}) \cup {}^\bullet t \colon M_1(s') = 1. \tag{5.2.5}$$

Da u_t laut Annahme nicht in σ_1 vorkommt, existiert in $\Sigma_{S\times\neg\varphi}$ die Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1',$$

wobei

$$\forall s' \in S \setminus \{s\} \colon M_1'(s') = M_1(s'). \tag{5.2.6}$$

Wegen $s \notin {}^\bullet t$ ergibt sich dann unmittelbar mit den Gleichungen 5.2.5 und 5.2.6

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{t} M_3.$$

Wegen $s \in t^\bullet$ gilt $M_3(s) = 1$ und zusammen mit ${}^\bullet u \cap {}^\bullet t = \emptyset$ und den Gleichungen 5.2.5 und 5.2.6 folgt

$$\forall s' \in {}^\bullet u \colon M_3(s') = 1.$$

Daraus ergibt sich die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{t} M_3 \xrightarrow{u} M_2',$$

wobei

$$\forall s' \in S \setminus \{s\} \colon M_2'(s') = M_2(s').$$

Daraus folgt, daß in $\Sigma_{S\times\neg\varphi}$ die unendliche Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{t} M_3 \xrightarrow{u} M_2' \xrightarrow{\sigma_2}$$

existiert, in der wegen $u_t \notin T_I$ auch weiterhin unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. (Sollte u_t erneut in σ_2 vorkommen, kann wiederum wie eben beschrieben verfahren werden).

(ii) ($\Rightarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine Schaltfolge $M_0 \xrightarrow{\sigma} M$, sodaß $L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})) \neq \emptyset$ und $L^\omega((N_h, M|_{S_h})) \neq \emptyset$. Seien also $\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M|_{S_{\neg\varphi}}))$ und $\sigma_h \in L^\omega((N_h, M|_{S_h}))$ gegeben.

(1) Sei $u \in s^\bullet$ in σ enthalten. Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar eine Zerlegung von σ in $\sigma_1 t \sigma_2 u \sigma_3$ und die Existenz einer aus der Pre-Agglomeration resultierenden Schaltfolge

$$M_0^r \xrightarrow{\sigma_1\sigma_2 u_t \sigma_3} M|_{S\setminus\{s\}}$$

in $\Sigma^r_{S\times\neg\varphi}$, wobei $u_t \in T^r$ eine Transition bezeichnet, die durch Pre-Agglomeration von s und t entstanden ist. Wegen $s \in S_H$ gilt $S_{\neg\varphi} \subseteq (S \setminus \{s\})$, und mit $S_{\neg\varphi} = S^r_{\neg\varphi}$ folgt unmittelbar

$$\sigma_{\neg\varphi} \in L((N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}})).$$

Ohne Einschränkung der Allgemeinheit kann angenommen werden, daß keine Transition aus $s^\bullet$ in σ_h vorkommt. (Der Fall, daß eine Transition aus $s^\bullet$ in σ_h vorkommt, wird in (3) betrachtet). Daraus folgt, daß keine Marke von s für das Ausführen von σ_h benötigt wird, und somit gilt

$$\sigma_h \in L^\omega((N_h^r, M|_{S_h^r})).$$

(2) Sei $u \in s^\bullet$ in $\sigma_{\neg\varphi}$ enthalten. Dann müßte $u \in T_B$ gelten, was jedoch im Widerspruch zu $s^\bullet \not\subseteq T_B$ steht.

(3) Sei $u \in s^\bullet$ in σ_h enthalten. Dann müssen zwei Fälle betrachtet werden.

(a) Sei t in σ_h enthalten. Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar eine Zerlegung von σ_h in $\sigma_{h_1} t \sigma_{h_2} u \sigma_{h_3}$ und die Existenz einer aus der Pre-Agglomeration resultierenden Schaltfolge

$$\sigma_{h_1}\sigma_{h_2} u_t \sigma_{h_3} \in L^\omega((N_h^r, M|_{S_h^r})),$$

wobei $u_t \in T^r$ eine Transition bezeichnet, die durch Pre-Agglomeration von s und t entstanden ist.

(b) Sei t in σ enthalten. Dann gibt es Zerlegungen von σ und σ_h in $\sigma_1 t \sigma_2$ und $\sigma_{h_1} u \sigma_{h_2}$. Da N_h ein Teilnetz von $N_\mathcal{B}$ darstellt, ist jede Schaltfolge von $(N_h, M|_{S_h})$ auch in $(N_\mathcal{B}, M)$ vorhanden. Deshalb existiert die Schaltfolge

$$M_0 \xrightarrow{\sigma_1 t \sigma_2} M \xrightarrow{\sigma_{h_1} u \sigma_{h_2}}$$

in $\Sigma_{S\times\neg\varphi}$. Aus Proposition 5.2.10 auf Seite 159 folgt unmittelbar

$$M_0 \xrightarrow{\sigma_1\sigma_2} M_1 \xrightarrow{\sigma_{h_1} t u \sigma_{h_2}},$$

wobei

$$\forall s' \in S \setminus \{s\} : M_1(s') \geq M(s'). \tag{5.2.7}$$

Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar

$$M_0^r \xrightarrow{\sigma_1\sigma_2} M_1|_{S\setminus\{s\}}$$

und die Existenz einer aus der Pre-Agglomeration resultierenden Schaltfolge

$$\sigma_{h_1} u_t \sigma_{h_2} \in L^\omega((N_h^r, M_1|_{S_h^r})),$$

wobei $u_t \in T^r$ eine Transition bezeichnet, die durch Pre-Agglomeration von s und t entstanden ist. Wegen $s^\bullet \not\subseteq T_B$ kann keine Transition aus $s^\bullet$ in $\sigma_{\neg\varphi}$ vorkommen. Folglich werden für das Ausführen von $\sigma_{\neg\varphi}$ keine Marken von s benötigt. Mit Gleichung 5.2.7 folgt unmittelbar

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}^r, M_1|_{S_{\neg\varphi}^r})).$$

($\Leftarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma^r_{S\times\neg\varphi}$, d.h., es existiert eine Schaltfolge $M^r_0 \xrightarrow{\sigma} M$, sodaß $L((N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}})) \neq \emptyset$ und $L^\omega((N^r_h, M|_{S^r_h})) \neq \emptyset$. Seien also $\sigma_{\neg\varphi} \in L((N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}}))$ und $\sigma_h \in L^\omega((N^r_h, M|_{S^r_h}))$ gegeben, und $u_t \in T^r$ bezeichne eine Transition, die durch Pre-Agglomeration von s und t entstanden ist.

(1) Sei u_t in σ enthalten. Dann gibt es eine Zerlegung von σ in $\sigma_1 u_t \sigma_2$. Aus Beweisrichtung (i) ($\Leftarrow$) folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1 tu\sigma_2} M'$$

in $\Sigma_{S\times\neg\varphi}$, wobei

$$\forall s' \in S \setminus \{s\} : M'(s') = M(s'). \tag{5.2.8}$$

Folglich gilt auch

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M'|_{S_{\neg\varphi}})) \quad \text{und} \quad \sigma_h \in L^\omega((N_h, M'|_{S_h})).$$

(2) Sei u_t in $\sigma_{\neg\varphi}$ enthalten. Dann müßte auch eine Transition $u \in s^\bullet$ in $\sigma_{\neg\varphi}$ vorkommen und somit $u \in T_B$ gelten, was jedoch im Widerspruch zu $s^\bullet \not\subseteq T_B$ steht.

(3) Sei u_t in σ_h enthalten. Dann gibt es eine Zerlegung von σ_h in $\sigma_{h_1} u_t \sigma_{h_2}$. Aus Beweisrichtung (i) ($\Leftarrow$) folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma} M'$$

in $\Sigma_{S\times\neg\varphi}$, wobei auch hier Gleichung 5.2.8 gilt, sowie

$$\sigma_{h_1} tu\sigma_{h_2} \in L^\omega((N_h, M'|_{S_h})).$$

Aus Gleichung 5.2.8 folgt dann auch

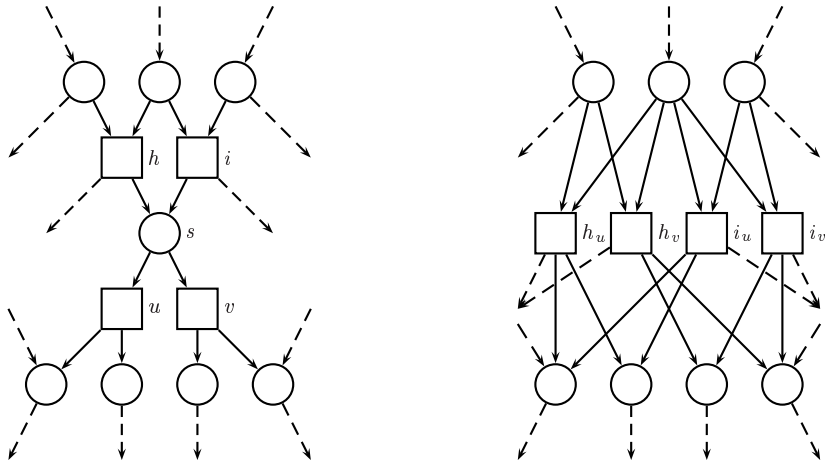
$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M'|_{S_{\neg\varphi}})).$$

■

5.2.7 Post-Agglomeration

In diesem Abschnitt wird die Post-Agglomeration vorgestellt, welche erstmals in [Ber85] erwähnt und später auch in [PPP00] verwendet wurde. Die Idee der Post-Agglomeration, welche in Abbildung 5.13 (a) auf der nächsten Seite dargestellt ist, besteht darin, daß jede Schaltfolge des Netzsystems, in der eine Transition $h \in {}^\bullet s$ und eine Transition $u \in s^\bullet$ vorkommen, in eine entsprechende Schaltfolge überführt werden kann, sodaß unmittelbar nach der Transition h die Transition u schaltet. Die Post-Agglomeration impliziert, daß ein Schalten der Ausgangstransitionen von s „vorgezogen“ werden kann, und „versteckt“ deren Schalten, indem sie mit den Eingangstransitionen von s verschmolzen werden (Abbildung 5.13 (b) auf der nächsten Seite).

Abbildung 5.13 *Teilnetz eines sicheren S/T-Netzsystems, für das die Stelle s der Post-Agglomeration genügt (a), und das reduzierte Teilnetz nach durchgeführter Post-Agglomeration (b)*



(a) (b)

Definition 5.2.10 (Post-Agglomeration [Ber85])
Gegeben sei ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$. Eine Stelle $s \in S$ mit $s \notin (s^\bullet)^\bullet$ genügt der Post-Agglomeration, falls gilt:

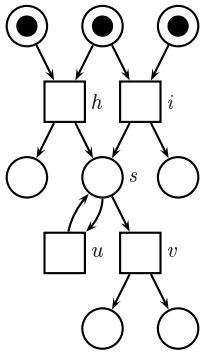
$$
\begin{aligned}
{}^\bullet s &\neq \emptyset \\
s^\bullet &\neq \emptyset \\
{}^\bullet(s^\bullet) &= \{s\} \\
M_0(s) &= 0
\end{aligned}
$$

Das durch Anwendung der Post-Agglomeration erhaltene Netzsystem $\Sigma^r = (N^r, M_0^r)$ mit $N^r = (S^r, T^r, F^r)$ ist definiert als:

$$
\begin{aligned}
S^r &= S \setminus \{s\} \\
T^r &= (T \setminus ({}^\bullet s \cup s^\bullet)) \cup \{t_u \mid (t,u) \in ({}^\bullet s \times s^\bullet)\} \\
F^r &= (F \cap ((S^r \times T^r) \cup (T^r \times S^r))) \cup \\
&\quad \{(s', t_u) \mid s' \in S^r \wedge t_u \in T^r \wedge (s', t) \in F\} \cup \\
&\quad \{(t_u, s') \mid s' \in S^r \wedge t_u \in T^r \wedge ((t, s') \in F \vee (u, s') \in F)\} \\
M_0^r &= M_0|_{S^r}
\end{aligned}
$$

Die Bedingung $s \notin (s^\bullet)^\bullet$ wird gefordert, damit die in Abbildung 5.14 auf der nächsten Seite dargestellte Situation nicht eintreten kann. In diesem Beispiel würde die Stelle s

Abbildung 5.14 *Sicheres S/T-Netzsystem, für das die Stelle s der Post-Agglomeration genügen würde, falls die Bedingung $s \in (s^\bullet)^\bullet$ zulässig wäre*



der Post-Agglomeration genügen, falls die Bedingung $s \in (s^\bullet)^\bullet$ zulässig wäre. Das Netzsystem enthält die Schaltfolge

$$M_0 \xrightarrow{h(u)^* v} M,$$

die auch in dem reduzierten Netzsystem repräsentiert sein müßte. Deshalb kann hier keine Post-Agglomeration durchgeführt werden.

Proposition 5.2.11 (Post-Agglomeration)
Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Stelle $s \in S$, die der Post-Agglomeration (Definition 5.2.10 auf der vorherigen Seite) genügt, sowie Transitionen $h \in {}^\bullet s$ und $u \in s^\bullet$. Es gilt:

$$M_0 \xrightarrow{\sigma_1 h \sigma_2 u} M \qquad \Rightarrow \qquad M_0 \xrightarrow{\sigma_1 h u \sigma_2} M$$

Beweis: Gegeben seien ein sicheres S/T-Netzsystem $\Sigma = (N, M_0)$ mit $N = (S, T, F)$, eine Stelle $s \in S$, die der Post-Agglomeration (Definition 5.2.10 auf der vorherigen Seite) genügt, sowie Transitionen $h \in {}^\bullet s$ und $u \in s^\bullet$.
Angenommen, es gibt eine Schaltfolge

$$M_0 \xrightarrow{\sigma_1 h \sigma_2 u} M,$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß keine Transition aus $s^\bullet$ in σ_2 vorkommt. Daraus folgt die Existenz von erreichbaren Markierungen $M_1, M_2, M_3 \in \mathcal{R}(M_0)$, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{h} M_2 \xrightarrow{\sigma_2} M_3 \xrightarrow{u} M.$$

Aus $s \in h^\bullet$ folgt unmittelbar $M_2(s) = 1$, und mit ${}^\bullet u = \{s\}$ erhält man

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{h} M_2 \xrightarrow{u} M_4,$$

wobei

$$\forall s' \in S \setminus \{s\} : M_4(s') \geq M_2(s'). \tag{5.2.9}$$

Die Marke auf der Stelle s wird für das Schalten von $M_2 \xrightarrow{\sigma_2} M_3$ nicht benötigt. (Ansonsten müßte eine Transition aus $s^\bullet$ in der Schaltfolge σ_2 vorkommen, was jedoch einen Widerspruch zu der Annahme ergäbe, daß dieses nicht der Fall ist).
Mit Gleichung 5.2.9 folgt also unmittelbar

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{h} M_2 \xrightarrow{u} M_4 \xrightarrow{\sigma_2} M.$$

■

Satz 5.2.7 (Post-Agglomeration bewahrt LTL-X-Eigenschaften [ES01a])
Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ und das nach Definition 5.2.10 auf Seite 165 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi}$. Die Bedingungen (i) und (ii) von Satz 5.1.3 auf Seite 126 gelten genau dann in $\Sigma_{S\times\neg\varphi}$, wenn sie in $\Sigma^r_{S\times\neg\varphi}$ gelten.

Beweis: Gegeben seien ein Produktnetzsystem $\Sigma_{S\times\neg\varphi} = (N_\mathcal{B}, M_0)$ mit $N_\mathcal{B} = (S, T, F, S_E)$, eine Stelle $s \in S$, die der Post-Agglomeration (Definition 5.2.10 auf Seite 165) genügt, sowie das nach Definition 5.2.10 erhaltene Netzsystem $\Sigma^r_{S\times\neg\varphi} = (N^r_\mathcal{B}, M^r_0)$ mit $N^r_\mathcal{B} = (S^r, T^r, F^r, S^r_E)$. Aus ${}^\bullet(s^\bullet) = \{s\}$ und Proposition 5.1.1 auf Seite 122 folgt $s^\bullet \subseteq T_H$. Somit gilt $s \in S_H$ und folglich ${}^\bullet s \not\subseteq T_B$. Wegen $s \notin (s^\bullet)^\bullet$ folgt mit Proposition 5.1.1 auf Seite 122 auch ${}^\bullet s \not\subseteq T_L$.

- **Vorbemerkung:** Es kann der Fall eintreten, daß es in $\Sigma_{S\times\neg\varphi}$ eine Schaltfolge

  $$M_0 \xrightarrow{\sigma_1 h \sigma_2}$$

  mit $h \in {}^\bullet s$ gibt, in der aber keine Transition aus $s^\bullet$ vorkommt. Da aber laut Voraussetzung $s^\bullet \neq \emptyset$ gilt, gibt es mindestens eine durch Post-Agglomeration entstandene Transition $h_u \in T^r$ mit

  $${}^\bullet h = {}^\bullet h_u \quad \text{und} \quad h^\bullet \subseteq h^\bullet_u.$$

  Daraus folgt unmittelbar die Existenz der Schaltfolge

  $$M^r_0 \xrightarrow{\sigma_1 h_u \sigma_2}$$

  in $\Sigma^r_{S\times\neg\varphi}$. Im weiteren Verlauf des Beweises brauchen also nur noch Fälle betrachtet zu werden, bei denen mindestens eine Transition aus $s^\bullet$ in den Schaltfolgen vorkommt.

(i) ($\Rightarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. Falls eine Transition $u \in s^\bullet$ in σ

vorkommt, muß es wegen $M_0(s) = 0$ eine Transition $h \in {}^\bullet s \setminus s^\bullet$ geben, die in σ vor u vorkommt. Deshalb gibt es eine Zerlegung von σ in

$$M_0 \xrightarrow{\sigma_1 h \sigma_2 u \sigma_3},$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß keine Transition aus $s^\bullet$ in σ_2 vorkommt. Mit Proposition 5.2.11 auf Seite 166 folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1 h u \sigma_2 \sigma_3} .$$

Das bedeutet, daß erreichbare Markierungen $M_1, M_2, M_3 \in \mathcal{R}(M_0)$ existieren, sodaß

$$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{h} M_2 \xrightarrow{u} M_3 \xrightarrow{\sigma_2 \sigma_3} .$$

Aus der Definition von $N^r_{\mathcal{B}}$ (Definition 5.2.10 auf Seite 165) folgt die Existenz einer Transition $h_u \in T^r$ mit

$${}^\bullet h_u = {}^\bullet h \quad \text{und} \quad h_u^\bullet = (h^\bullet \setminus \{s\}) \cup u^\bullet.$$

Daraus folgt jedoch die Existenz der Schaltfolge

$$M^r_0 \xrightarrow{\sigma_1} M_1|_{S\setminus\{s\}} \xrightarrow{h_u} M_3|_{S\setminus\{s\}}$$

in $\Sigma^r_{S\times\neg\varphi}$, und wegen $M_3(s) = 0$ kann unmittelbar

$$M^r_0 \xrightarrow{\sigma_1} M_1|_{S\setminus\{s\}} \xrightarrow{h_u} M_3|_{S\setminus\{s\}} \xrightarrow{\sigma_2\sigma_3}$$

in $\Sigma^r_{S\times\neg\varphi}$ abgeleitet werden. (Sollte erneut eine Transition aus $s^\bullet$ in σ_3 vorkommen, kann wiederum wie eben beschrieben verfahren werden). Wegen $h \notin T_B$ und $u \in T_H$ enthält die Schaltfolge auch weiterhin unendlich viele unendliche Trace-Wächter (Transitionen aus T_I).

($\Leftarrow$) Angenommen, Bedingung (i) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma^r_{S\times\neg\varphi}$, d.h., es existiert eine unendliche Schaltfolge $M^r_0 \xrightarrow{\sigma}$, in der unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. Falls eine durch Post-Agglomeration entstandene Transition $h_u \in T^r$ in σ vorkommt, gibt es eine Zerlegung von σ in

$$M^r_0 \xrightarrow{\sigma_1 h_u \sigma_2},$$

wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß h_u nicht in σ_1 vorkommt. Das bedeutet, daß erreichbare Markierungen $M_1, M_2, \in \mathcal{R}(M^r_0)$ existieren, sodaß

$$M^r_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{h_u} M_2 \xrightarrow{\sigma_2} .$$

Daraus folgt nach Definition 5.2.10 auf Seite 165 die Existenz einer Stelle $s \in S$ sowie zweier Transitionen $h \in {}^\bullet s \setminus s^\bullet$ und $u \in s^\bullet \setminus {}^\bullet s$, sodaß

$$\begin{aligned} {}^\bullet h_u &= {}^\bullet h \\ h_u^\bullet &= (h^\bullet \setminus \{s\}) \cup u^\bullet \\ {}^\bullet u &= \{s\} \end{aligned}$$

Daraus folgt unmittelbar

$$\forall s' \in {}^\bullet h\colon M_1(s') = 1. \tag{5.2.10}$$

Da h_u laut Annahme nicht in σ_1 vorkommt, existiert in $\Sigma_{S\times\neg\varphi}$ die Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1',$$

wobei

$$\forall s' \in S \setminus \{s\}\colon M_1'(s') = M_1(s'). \tag{5.2.11}$$

Wegen $s \notin {}^\bullet h$ ergibt sich dann unmittelbar mit den Gleichungen 5.2.10 und 5.2.11

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{h} M_3,$$

wobei $M_3(s) = 1$. Daraus folgt mit ${}^\bullet u = \{s\}$ die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{h} M_3 \xrightarrow{u} M_2',$$

wobei

$$\forall s' \in S \setminus \{s\}\colon M_2'(s') = M_2(s').$$

Daraus folgt, daß in $\Sigma_{S\times\neg\varphi}$ die unendliche Schaltfolge

$$M_0 \xrightarrow{\sigma_1} M_1' \xrightarrow{h} M_3 \xrightarrow{u} M_2' \xrightarrow{\sigma_2}$$

existiert, in der wegen $h_u \notin T_I$ auch weiterhin unendlich viele unendliche Trace-Wächter (Transitionen aus T_I) vorkommen. (Sollte h_u erneut in σ_2 vorkommen, kann wiederum wie eben beschrieben verfahren werden).

(ii) ($\Rightarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma_{S\times\neg\varphi}$, d.h., es existiert eine Schaltfolge $M_0 \xrightarrow{\sigma} M$, sodaß $L((N_{\neg\varphi}, M|_{S_{\neg\varphi}})) \neq \emptyset$ und $L^\omega((N_h, M|_{S_h})) \neq \emptyset$. Seien also $\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M|_{S_{\neg\varphi}}))$ und $\sigma_h \in L^\omega((N_h, M|_{S_h}))$ gegeben.

(1) Sei $u \in s^\bullet$ in σ enthalten. Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar eine Zerlegung von σ in $\sigma_1 h \sigma_2 u \sigma_3$ mit $h \in {}^\bullet s$ und die Existenz einer aus der Post-Agglomeration resultierenden Schaltfolge

$$M_0^r \xrightarrow{\sigma_1 h_u \sigma_2 \sigma_3} M|_{S\setminus\{s\}}$$

in $\Sigma^r_{S\times\neg\varphi}$, wobei $h_u \in T^r$ eine Transition bezeichnet, die durch Post-Agglomeration von h und u entstanden ist. Wegen $s \in S_H$ gilt $S_{\neg\varphi} \subseteq (S \setminus \{s\})$, und mit $S_{\neg\varphi} = S^r_{\neg\varphi}$ folgt unmittelbar

$$\sigma_{\neg\varphi} \in L((N^r_{\neg\varphi}, M|_{S^r_{\neg\varphi}})).$$

Ohne Einschränkung der Allgemeinheit kann angenommen werden, daß keine Transition aus $s^\bullet$ in σ_h vorkommt. (Der Fall, daß eine Transition aus $s^\bullet$ in σ_h vorkommt, wird in (3) betrachtet). Daraus folgt, daß keine Marke von s für das Ausführen von σ_h benötigt wird, und somit gilt

$$\sigma_h \in L^\omega((N^r_h, M|_{S^r_h})).$$

(2) Sei $u \in s^\bullet$ in $\sigma_{\neg\varphi}$ enthalten. Dann müßte $u \in T_B$ gelten, was jedoch im Widerspruch zu $s^\bullet \subseteq T_H$ steht.

(3) Sei $u \in s^\bullet$ in σ_h enthalten. Dann gilt es, 2 Fälle zu unterscheiden.

(a) Sei $h \in {}^\bullet s$ in σ_h enthalten. Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar eine Zerlegung von σ_h in $\sigma_{h_1} h \sigma_{h_2} u \sigma_{h_3}$ und die Existenz einer aus der Post-Agglomeration resultierenden Schaltfolge

$$\sigma_{h_1} h_u \sigma_{h_2} \sigma_{h_3} \in L^\omega((N^r_h, M|_{S^r_h})),$$

wobei $h_u \in T^r$ eine Transition bezeichnet, die durch Post-Agglomeration von h und u entstanden ist.

(b) Sei $h \in {}^\bullet s$ in σ enthalten. Dann gibt es Zerlegungen von σ und σ_h in $\sigma_1 h \sigma_2$ und $\sigma_{h_1} u \sigma_{h_2}$, wobei ohne Einschränkung der Allgemeinheit angenommen werden kann, daß keine Transition aus $s^\bullet$ in σ_2 oder σ_{h_1} vorkommt. Da N_h ein Teilnetz von $N_\mathcal{B}$ darstellt, ist jede Schaltfolge von $(N_h, M|_{S_h})$ auch in $(N_\mathcal{B}, M)$ vorhanden. Deshalb existiert die Schaltfolge

$$M_0 \xrightarrow{\sigma_1 h \sigma_2} M \xrightarrow{\sigma_{h_1} u \sigma_{h_2}}$$

in $\Sigma_{S\times\neg\varphi}$. Aus Proposition 5.2.11 auf Seite 166 folgt unmittelbar

$$M_0 \xrightarrow{\sigma_1 h u \sigma_2} M_1 \xrightarrow{\sigma_{h_1} \sigma_{h_2}},$$

wobei

$$\forall s' \in S \setminus \{s\} : M_1(s') \geq M(s'). \tag{5.2.12}$$

Aus Beweisrichtung (i) ($\Rightarrow$) folgt unmittelbar die Existenz einer aus der Post-Agglomeration resultierenden Schaltfolge

$$M^r_0 \xrightarrow{\sigma_1 h_u \sigma_2} M_1|_{S\setminus\{s\}},$$

wobei $h_u \in T^r$ eine Transition bezeichnet, die durch Post-Agglomeration von h und u entstanden ist. Mit Gleichung 5.2.12 auf der vorherigen Seite folgt

$$\sigma_{h_1}\sigma_{h_2} \in L^\omega((N_h^r, M_1|_{S_h^r})).$$

Wegen $s^\bullet \not\subseteq T_B$ kann keine Transition aus $s^\bullet$ in $\sigma_{\neg\varphi}$ vorkommen. Folglich werden für das Ausführen von $\sigma_{\neg\varphi}$ keine Marken von s benötigt. Mit Gleichung 5.2.12 auf der vorherigen Seite folgt unmittelbar

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}^r, M_1|_{S_{\neg\varphi}^r})).$$

($\Leftarrow$) Angenommen, Bedingung (ii) von Satz 5.1.3 auf Seite 126 gilt in $\Sigma^r_{S\times\neg\varphi}$, d.h., es existiert eine Schaltfolge $M_0^r \xrightarrow{\sigma} M$, sodaß $L((N_{\neg\varphi}^r, M|_{S_{\neg\varphi}^r})) \neq \emptyset$ und $L^\omega((N_h^r, M|_{S_h^r})) \neq \emptyset$. Seien also $\sigma_{\neg\varphi} \in L((N_{\neg\varphi}^r, M|_{S_{\neg\varphi}^r}))$ und $\sigma_h \in L^\omega((N_h^r, M|_{S_h^r}))$ gegeben, und $h_u \in T^r$ bezeichne eine Transition, die durch Post-Agglomeration von $h \in {}^\bullet s$ und $u \in s^\bullet$ entstanden ist.

(1) Sei h_u in σ enthalten. Dann gibt es eine Zerlegung von σ in $\sigma_1 h_u \sigma_2$. Aus Beweisrichtung (i) ($\Leftarrow$) folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma_1 hu\sigma_2} M'$$

in $\Sigma_{S\times\neg\varphi}$, wobei

$$\forall s' \in S \setminus \{s\} : M'(s') = M(s'). \tag{5.2.13}$$

Folglich gilt auch

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M'|_{S_{\neg\varphi}})) \quad \text{und} \quad \sigma_h \in L^\omega((N_h, M'|_{S_h})).$$

(2) Sei h_u in $\sigma_{\neg\varphi}$ enthalten. Dann müßte auch eine Transition $h \in {}^\bullet s$ in $\sigma_{\neg\varphi}$ vorkommen und somit $h \in T_B$ gelten, was jedoch im Widerspruch zu ${}^\bullet s \not\subseteq T_B$ steht.

(3) Sei h_u in σ_h enthalten. Dann gibt es eine Zerlegung von σ_h in $\sigma_{h_1} h_u \sigma_{h_2}$. Aus Beweisrichtung (i) ($\Leftarrow$) folgt unmittelbar die Existenz der Schaltfolge

$$M_0 \xrightarrow{\sigma} M'$$

in $\Sigma_{S\times\neg\varphi}$, wobei auch hier Gleichung 5.2.13 gilt, sowie

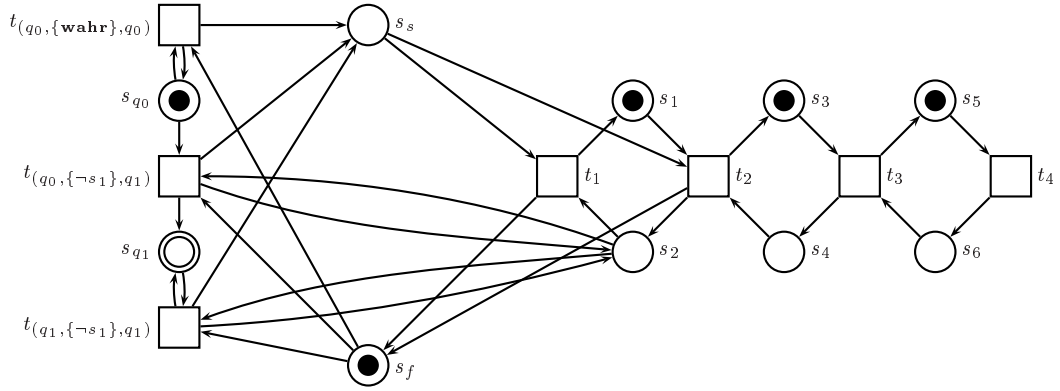
$$\sigma_{h_1} hu\sigma_{h_2} \in L^\omega((N_h, M'|_{S_h})).$$

Aus Gleichung 5.2.13 folgt dann auch

$$\sigma_{\neg\varphi} \in L((N_{\neg\varphi}, M'|_{S_{\neg\varphi}})).$$

■

Abbildung 5.15 *Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ für die Verifikation der Eigenschaft $\varphi = \Box\Diamond s_1$*



5.3 Anwendungsbeispiel

In diesem Abschnitt soll anhand eines Beispiels gezeigt werden, wie ein Produktnetzsystem zur Verifikation einer LTL-X-Eigenschaft effizient durch das Zusammenspiel der in den vorherigen Abschnitten vorgestellten Reduktionsregeln verkleinert werden kann.
Abbildung 5.15 zeigt das Produktnetzsystem $\Sigma_{S\times\neg\varphi}$, das zum Nachweis der LTL-X-Eigenschaft

$$\varphi = \Box\Diamond s_1$$

(„*die Stelle s_1 wird immer wieder markiert sein*“) für ein Speichermodell der Kapazität 3 herangezogen wird. Das Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ ergibt sich aus der Synchronisation des Büchiautomaten, der genau diejenigen Abläufe akzeptiert, welche die Eigenschaft φ verletzen, und dem ursprünglichen System. Alle die Eigenschaft φ verletzenden Abläufe erfüllen genau die Eigenschaft

$$\begin{aligned}\neg\varphi &= \neg(\Box\Diamond s_1)\\ &= \Diamond\Box\neg s_1\\ &= \Diamond\Box\overline{s_1}\\ &= \Diamond\Box s_2 \qquad (\text{da } s_2 = \overline{s_1})\end{aligned}$$

(„*irgendwann wird die Stelle s_2 immer markiert sein*“).
Es ist leicht einzusehen, daß das Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ weder einen illegalen ω-Trace noch einen illegalen Livelock enthält. Ein illegaler ω-Trace läge dann vor, falls es einen unendlichen Ablauf $M_0 \xrightarrow{\sigma}$ von $\Sigma_{S\times\neg\varphi}$ gäbe, in dem unendlich viele Transitionen aus

$$T_I = \{t_{(q_0,\{\neg s_1\},q_1)}, t_{(q_1,\{\neg s_1\},q_1)}\}$$

vorkämen. Jedoch führt jeder Ablauf von $\Sigma_{S\times\neg\varphi}$, in dem eine Transition aus T_I vorkommt, zu einer Verklemmung, wie die Schaltfolge

$$M_0 \xrightarrow{\sigma_1} (s_{q_0}, s_f, s_2, H_i) \xrightarrow{t_{(q_0,\{\neg s_1\},q_1)}} (s_{q_1}, s_s, s_2, H_i) \xrightarrow{\sigma_2} (s_{q_1}, s_s, s_2, H_j) \xrightarrow{t_1} (s_{q_1}, s_f, s_1, H_j) \xrightarrow{\sigma_3} (s_{q_1}, s_f, s_1, H_k)$$

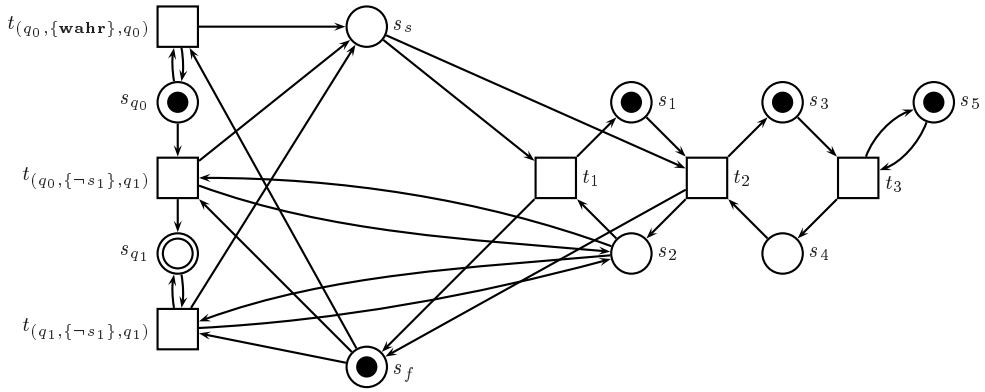
zeigt, wobei die Markierungen als 4-Tupel, unterteilt nach Büchistellen, Schedulerstellen, sichtbaren und unsichtbaren Stellen, dargestellt werden und ohne Einschränkung der Allgemeinheit angenommen werden kann, daß keine Transition aus T_I in σ_1 vorkommt und $\sigma_2, \sigma_3 \in T_H^*$. In $\Sigma_{S\times\neg\varphi}$ kann es auch keinen illegalen Livelock geben, da es keine unendliche Schaltfolge aus T_H^ω gibt.
Nun kann gezeigt werden, wie das Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ aus Abbildung 5.15 auf der vorherigen Seite unter Bewahrung dieser Eigenschaften mittels der Reduktionstechniken verkleinert werden kann.
Die Stelle s_6 und die Transition t_4 genügen der Pre-Agglomeration (Definition 5.2.9 auf Seite 157) und können somit entfernt werden. Das Ergebnis dieser Reduktion ist in Abbildung 5.16 (a) auf der nächsten Seite dargestellt. Durch die Anwendung der Pre-Agglomeration ist s_5 nun zu einer impliziten Stelle geworden, da sie ein Schalten der Transition t_3 niemals verhindern kann. Bleibt zu zeigen, daß die Stelle s_5 auch tatsächlich mit der Regel zur Erkennung impliziter Stellen (Definition 5.2.3 auf Seite 133) entdeckt wird. Dieses ist aber einfach einzusehen, da der Inzidenzvektor von s_5 dem Nullvektor $\mathbf{0}$ entspricht und somit $Y = \mathbf{0}$ und $\mu = 1$ offensichtlich eine optimale Lösung des Ungleichungssystems aus Definition 5.2.3 auf Seite 133 ergeben. Nach Entfernen der impliziten Stelle s_5 erhält man das Netzsystem aus Abbildung 5.16 (b). Nun können erneut die Stelle s_4 und die Transition t_3 mittels Pre-Agglomeration und die daraus resultierende implizite Stelle s_3 aus dem Netzsystem entfernt werden. Daraus ergibt sich das in Abbildung 5.16 (c) dargestellte Netzsystem, welches nicht mehr weiter reduziert werden kann. Es ist leicht einzusehen, daß auch dieses Netzsystem weder einen illegalen ω-Trace noch einen illegalen Livelock aufweist.
Die aufgezeigten Reduktionen wirken sich sehr positiv auf den entfaltungsbasierten LTL-X-Model-Checking-Algorithmus von [EH01, Hel02] aus, da beispielsweise bei der Verifikation des obigen Speichermodells mit Kapazität 5 bereits ein Präfix mit 70 Bedingungen und 33 Ereignissen erzeugt wird, wohingegen das Präfix des reduzierten Produktnetzsystem (Abbildung 5.16 (c)) lediglich 18 Bedingungen und 7 Ereignisse aufweist.

5.4 Implementierungsaspekte

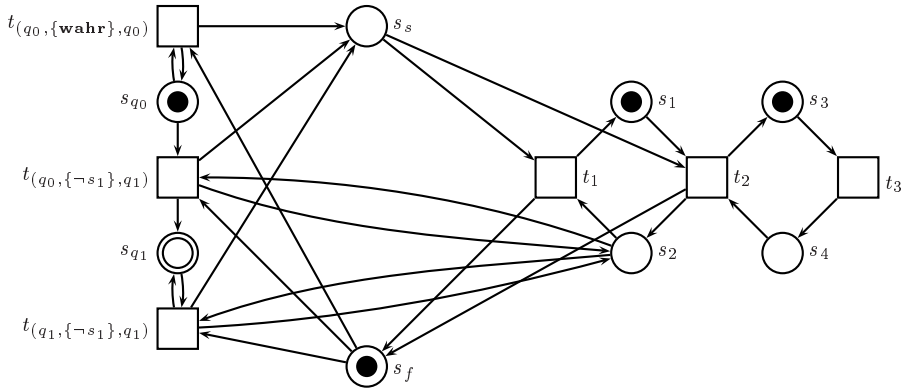
Bei der Implementierung der Reduktionsregeln sollte besondere Sorgfalt darauf gelegt werden, den Zeitaufwand für die Reduktionen so gering wie möglich zu halten. Von daher werden in diesem Abschnitt einige Heuristiken diskutiert, die bei der Umsetzung der Reduktionen berücksichtigt wurden.
Zuerst wird das in Abschnitt 5.2.2 auf Seite 134 beschriebene Verfahren zur Eliminierung toter Transitionen angewendet, da es das Produktnetzsystem global nach Stellen

Abbildung 5.16 *Produktnetzsystem $\Sigma_{S\times\neg\varphi}$ nach durchgeführter Pre-Agglomeration von s_6 und t_4 (a), nach Eliminierung der impliziten Stelle s_5 (b) sowie nach erneuter Anwendung der Reduktionen (c).*

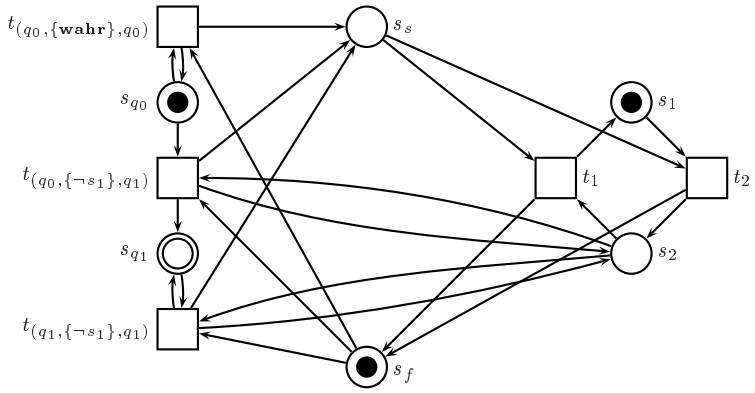
(a)



(b)



(c)



und Transitionen durchsucht, die gleichzeitig aus dem Netz entfernt werden können. Wie bereits erwähnt wurde, liefert die Anwendung dieses Verfahrens eine Menge von Stellen, die unter jeder erreichbaren Markierung unmarkiert sind. Das Ungleichungssystem kann jedoch mehrere Lösungen besitzen, und deshalb wird die Methode wiederholt angewendet, um möglichst viele unmarkierte Stellen und tote Transitionen schon im Vorfeld zu eliminieren. Darüberhinaus stellt $Y = \mathbf{0}$ immer eine Lösung des Ungleichungssystems dar. Um diese Lösung nach Möglichkeit auszuschließen und eine möglichst große Menge unmarkierter Stellen zu ermitteln, wird das Ungleichungssystem um die Zielfunktion

$$\mathit{Maximiere}\ Y^t \cdot \mathbf{1}$$

erweitert, wobei $\mathbf{0} \leq Y \leq \mathbf{1}$. Anschließend werden die Regeln zur Erkennung toter Transitionen eingesetzt, die auf lokalen Teilnetzen arbeiten.
Die übrigen Regeln sind Bestandteil einer *while*-Schleife, die solange durchlaufen wird, bis keine der Regeln mehr angewendet werden kann. Zunächst werden *redundante Stellen* aus dem Netzsystem entfernt. Dabei wird eine Stelle s_1 genau dann als *redundant* bezeichnet, wenn es eine Stelle s_2 gibt, welche dieselbe Anfangsmarkierung und denselben Vor- und Nachbereich wie s_1 aufweist. Eine redundante Stelle beschreibt also den einfachsten Fall einer impliziten Stelle und würde auch nach dem in Abschnitt 5.2.1 auf Seite 130 beschriebenen Verfahren erkannt werden, jedoch dauert es unter Umständen länger, ein lineares Ungleichungssystem zu lösen, als eine doppelt verkettete Liste von Stellen zu durchlaufen und deren Vor- und Nachbereiche zu vergleichen. Anschließend werden nacheinander die Abstraktion, die Pre- und die Post-Agglomeration angewendet. Sie basieren auf einfachen Operationen über effizienten Datenstrukturen, mit denen die Netzsysteme verwaltet werden. Dabei wird auf eine von Römer [Röm00] vorgeschlagene universelle Datenstruktur zurückgegriffen, die mittels doppelt verketteter Listen einen schnellen Zugriff auf Stellen und Transitionen sowie deren Vor- und Nachbereiche ermöglicht. Deshalb stellen die Abstraktion, die Pre- und die Post-Agglomeration schnelle Transformationen im Vergleich zu den Regeln dar, die auf Techniken der linearen Programmierung zurückgreifen.
Das Verfahren aus Abschnitt 5.2.1 auf Seite 130 zur Eliminierung impliziter Stellen wird zuletzt angewendet, da die Anzahl und Größe der zu lösenden Ungleichungssysteme von der Größe des Netzsystems abhängen. Daher empfiehlt es sich, das Netzsystem bereits vor der Anwendung dieses Verfahrens so stark wie möglich zu reduzieren. Darüberhinaus kann das Ungleichungssystem für jede Stelle ohne signifikanten Zeitverlust aufgestellt werden, da für alle Stellen dieselbe Zielfunktion

$$\mathit{Minimiere}\ Y^t \cdot M_0 + \mu$$

betrachtet wird und auch mit $Y^t \cdot \mathbf{N}$ ein Großteil des Ungleichungssystems immer gleich bleibt. Folglich brauchen diese nur einmal erstellt zu werden und können dann für den Test jeder Stelle wiederverwendet werden. Lediglich die rechten Seiten des Ungleichungssystems müssen für jeden Test geringfügig modifiziert werden, und für eine zu testende Stelle s müssen noch $|s^\bullet|$ Ungleichungen hinzugefügt werden, aber der Aufwand dafür ist vernachlässigbar klein.

Tabelle 5.1 *Experimentelle Ergebnisse*

	$\Sigma_{S\times\neg\varphi}$		$\text{PREF}(\Sigma_{S\times\neg\varphi})$			$\Sigma^r_{S\times\neg\varphi}$			$\text{PREF}(\Sigma^r_{S\times\neg\varphi})$		
	$\lvert S\rvert$	$\lvert T\rvert$	$\lvert B\rvert$	$\lvert E\rvert$	t_{LTL-X}	$\lvert S\rvert$	$\lvert T\rvert$	t_r	$\lvert B\rvert$	$\lvert E\rvert$	t_{LTL-X}
BUFFER(100)	205	107	10111	5054	1274.9	7	8	41.6	13	5	< 0.1
FIFO(20)	171	132	64167	42107	47229.8	39	31	3.9	736	364	2.5
PRODCELL(5)	239	214	1803	810	35.8	117	128	2.9	998	416	8.4
DPH(7)	71	127	72472	35021	11280.9	31	101	1.0	2971	1406	4.5
FURNACE(3)	58	105	33363	19322	1541.7	36	92	0.6	12819	7640	132.4
KEY(4)	169	180	138052	68585	49516.4	121	153	2.8	95416	57062	31196.4
RW(3)(1)	111	276	29717	15862	3828.5	80	202	0.9	25882	12078	1796.2
RW(1)(2)	214	1488	19874	9770	2384.8	159	981	8.2	17644	7550	1395.4
SLOTRING(8)	85	86	24734	17195	5379.5	54	55	0.5	15283	7956	930.4
SLOTRING(10)	105	106	67266	48145	44407.4	67	68	0.7	44562	23901	8842.9

5.5 Experimentelle Ergebnisse

In diesem Abschnitt soll anhand von Testergebnissen ermittelt werden, welche Reduktionsraten für die Testsysteme erzielt werden können und wie sich die Reduktionszeiten für die Testsysteme im Vergleich zu den Verifikationszeiten für die LTL-X-Eigenschaften verhalten. Eine Beschreibung aller innerhalb der Testreihen verwendeten Systeme kann Anhang A auf Seite 223 entnommen werden.
Alle Systeme außer PRODCELL(5) wurden auf die LTL-X-Eigenschaft

$$\neg\Diamond(s_1 \wedge s_2)$$

(„*es gibt keinen Ablauf, in dem irgendwann die Stellen s_1 und s_2 gleichzeitig markiert sind*“) überprüft, wohingegen das System PRODCELL(5) auf Gültigkeit der Invariante

$$\Box((s_1 \wedge \neg s_2 \wedge \neg s_3) \vee (\neg s_1 \wedge s_2 \wedge \neg s_3) \vee (\neg s_1 \wedge \neg s_2 \wedge s_3))$$

(„*in jedem Systemzustand ist jeweils nur genau eine der Stellen s_1, s_2 und s_3 markiert*“) getestet wurde.
Die Experimente wurden auf einem SUN Ultra 60 Rechner durchgeführt, der mit 1,5 GByte Hauptspeicher und einem 295 MHz UltraSPARC-II Prozessor ausgestattet war. Für das Lösen der linearen Ungleichungssysteme wurde CPLEX^{TM} in der Version 6.5.1 verwendet, und der LTL-X Model-Checker UNFSMODELS von [EH00a, Hel02] lag in der Version 0.9 vor.
Die Ergebnisse sind in Tabelle 5.1 dargestellt. Die Spalten $|S|$ ($|B|$) und $|T|$ ($|E|$) bezeichnen die Anzahl von Stellen (Bedingungen) und Transitionen (Ereignissen) des Produktnetzsystems (Produktnetzpräfixes). Die Spalten t_{LTL-X} und t_r geben die Zeiten in Sekunden an, die für die Verifikation der LTL-X-Eigenschaften mit UNFSMODELS und die Durchführung der Reduktionen benötigt wurden.
Die Ergebnisse zeigen, daß für alle Testsysteme hohe Reduktionsraten erzielt werden. Die

Tabelle 5.2 *Experimentelle Ergebnisse bei Anwendung der in [DE95, Ber85] vorgeschlagenen Abstraktions- und Pre-Agglomerationsregel*

	Verallgemeinerte Regeln					Regeln in [DE95, Ber85, PPP00]				
	$\Sigma^r_{S\times\neg\varphi}$		$\textsc{Pref}(\Sigma^r_{S\times\neg\varphi})$			$\Sigma^r_{S\times\neg\varphi}$		$\textsc{Pref}(\Sigma^r_{S\times\neg\varphi})$		
	$\lvert S\rvert$	$\lvert T\rvert$	$\lvert B\rvert$	$\lvert E\rvert$	t_{LTL-X}	$\lvert S\rvert$	$\lvert T\rvert$	$\lvert B\rvert$	$\lvert E\rvert$	t_{LTL-X}
Dph(7)	31	101	2971	1406	4.5	50	114	27922	13281	709.5
Slotring(10)	67	68	44562	23901	8842.9	67	77	125351	62067	31674.9

Präfixe der reduzierten Netzsysteme sind viel kleiner als die Präfixe der Ausgangsnetzsysteme. Diese Tatsache wirkt sich erheblich auf den LTL-X Model-Checker UnfSmodels aus, da dieser entfaltungsbasierte Verifikationstechniken einsetzt. Die experimentellen Ergebnisse bestätigen, daß die t_{LTL-X}-Zeiten für die reduzierten Netzsysteme deutlich unter den Verifikationszeiten für die Ausgangssysteme liegen. Beispielsweise konnte die Verifikationszeit für das Fifo(20)-System durch vorherige Anwendung der Reduktionsmethoden von 13 Stunden auf nahezu 3 Sekunden verringert werden. Ebenso wurde die Verifikation des Dph(7)-Systems anstatt in 3 Stunden in lediglich 5 Sekunden durchgeführt, und auch die Verifikationszeit für Slotring(10) konnte von 12 Stunden auf nur 2,5 Stunden verkürzt werden. Darüberhinaus zeigen die Ergebnisse deutlich, daß die für die Reduktionen benötigte Zeit t_r vernachlässigbar klein im Verhältnis zur Verifikationszeit t_{LTL-X} der Ausgangssysteme ausfällt.
Wie bereits erwähnt wurde, sind die in Abschnitt 5.2.5 auf Seite 148 beschriebene Abstraktion und die in Abschnitt 5.2.6 auf Seite 156 vorgestellte Pre-Agglomeration bezüglich der in [DE95, Ber85, PPP00] betrachteten Regeln verallgemeinert worden. Daher liegt die Vermutung nahe, daß diese Regeln in einigen Fällen öfter angewendet und somit höhere Reduktionsraten erzielt werden können. Zur näheren Untersuchung dieser Vermutung wurden die Abstraktions- und Pre-Agglomerationsregel derart implementiert, wie es in [DE95, Ber85, PPP00] beschrieben worden ist, und anschließend neuerlichen Tests unterzogen.
Die Ergebnisse sind in Tabelle 5.2 dargestellt. Es wird deutlich, daß die Anwendung der in [DE95, Ber85, PPP00] betrachteten Regeln bei den Systemen Dph(7) und Slotring(10) zur Bildung größerer Präfixe führt und als Folge davon auch die Verifikationszeiten für die LTL-X-Eigenschaften deutlich ansteigen. Beispielsweise konnte das System Dph(7) der speisenden Philosophen nur noch auf 50 Stellen und 114 Transitionen reduziert werden (im Gegensatz zu 31 Stellen und 101 Transitionen bei Anwendung der verallgemeinerten Regeln). Als Folge davon wurde die Präfixgröße verneunfacht, und auch die Verifikationszeit für die LTL-X-Eigenschaft verschlechterte sich von 5 Sekunden auf nunmehr 12 Minuten. Nach Anwendung der Regeln aus [DE95, Ber85, PPP00] auf die Modellierung des Slotted-Ring-Protokolls (Slotring(10)) unterscheiden sich die reduzierten Netzsysteme um 9 Transitionen. Auf den ersten Blick scheint dieses ein zu vernachlässigender Faktor zu sein, jedoch hat diese Differenz um 9 Transitio-

nen erhebliche Auswirkungen auf die Größe der Präfixe und die daraus resultierende Verifikationszeit für die LTL-X-Eigenschaft nach sich gezogen. Eine Vorverarbeitung des Netzsystems mit den verallgemeinerten Regeln hat die Präfixgröße auf ein Drittel reduziert, und die Verifikationszeit konnte von 9 Stunden auf gerade einmal 2,5 Stunden verringert werden. Darüberhinaus besitzt das Netzsystem, welches mit den Regeln aus [DE95, Ber85, PPP00] vorverarbeitet wurde, ein größeres Präfix als das ursprüngliche Netzsystem. Dieser Sachverhalt kann im engen Zusammenhang mit der Post-Agglomeration stehen, deren Anwendung in einigen Fällen sogar zur Vergrößerung von Präfixen führen kann.
Die Reduktionsraten, die für ein System erzielt werden können, hängen zu einem nicht zu vernachlässigbaren Faktor von der zu verifizierenden LTL-X-Eigenschaft ab. Alle Stellen des Netzsystems, die atomaren Propositionen der LTL-X-Formel entsprechen, sowie deren Eingangs- und Ausgangstransitionen, bleiben von den Reduktionsregeln unberührt. Das bedeutet, daß die Anzahl von Stellen und Transitionen, die von vornherein nicht entfernt werden können, von der Anzahl der Stellen abhängt, die als atomare Propositionen in der Formel vorkommen.

Kapitel 6

Nachweis von temporallogischen Eigenschaften für M-Netzsysteme

Das Konzept der Netzentfaltungen und die in [ERV02, ES01a, ES01b, Hel99, Hel02, KK00, KK01, KKV02, McM92, MR97, Röm00] beschriebenen Algorithmen vermindern das Problem der Zustandsraumexplosion bei der automatischen Verifikation von S/T-Netzsystemen. Darüberhinaus wurde in [HKK02] gezeigt, daß die Präfixkonstruktion effizient parallelisiert werden kann. Allerdings besteht ein Nachteil dieser Methoden bezüglich ihrer praktischen Anwendbarkeit darin, daß S/T-Netzsysteme nicht als Modellierungssprache geeignet sind, da sie ein unstrukturiertes Low-Level-Modell darstellen. Aus diesem Grund ist es erstrebenswert, die angesprochenen Techniken auf einen Formalismus zu übertragen, der für Modellierungszwecke geeigneter erscheint, wie zum Beispiel gefärbte Petri-Netze („*coloured Petri nets*“) [Jen91, Jen92, Jen94, Jen97, Jen98]. Diese erlauben es, auf natürliche Art und Weise viele Konstrukte zu modellieren, die in „High-Level“-Sprachen zur Spezifikation von nebenläufigen Systemen zur Verfügung stehen, siehe beispielsweise [BH93, BFF+95b, FG96, FG97]. Darüberhinaus gibt es verschiedene Tools [GRT+95, SSE03], mit deren Hilfe High-Level-Spezifikationen automatisch in gefärbte Petri-Netze übersetzt werden können. Obwohl es auch möglich ist, gefärbte Petri-Netze unter bestimmten Voraussetzungen in entsprechende S/T-Netzsysteme zu expandieren und dann deren Präfixe für die Verifikation zu verwenden, besteht ein erheblicher Nachteil darin, daß die expandierten S/T-Netzsysteme oftmals exponentiell größer sind als deren Präfixe. Darüberhinaus ist es in vielen Fällen unmöglich, ein gefärbtes Netzsystem mit vertretbarem Speicher- und Zeitaufwand in ein S/T-Netzsystem zu überführen. Deshalb wurde in [KK03] ein Verfahren beschrieben, das gefärbte Petri-Netzsysteme direkt in endliche, vollständige Präfixe entfaltet, ohne den Umweg über die kostenintensiven Expansionen (S/T-Netzsysteme) gehen zu müssen.

Basierend auf diesem Hintergrund und dem in Abschnitt 5.1.4 auf Seite 119 beschriebenen Ansatz für die Verifikation von temporallogischen Eigenschaften sicherer S/T-Netzsysteme wurde in [SK04] ein paralleler Algorithmus zur Verifikation temporallogischer Eigenschaften von gefärbten Petri-Netzsystemen vorgeschlagen, der in seiner Art der erste war und im weiteren Verlauf dieses Kapitels vorgestellt werden soll.

6.1 Vorbemerkungen

In diesem Abschnitt werden zunächst Begriffe und Definitionen eingeführt, die für das Verständnis der folgenden Abschnitte von grundlegender Bedeutung sind.

6.1.1 M-Netzexpansionen

Jedes beschränkte M-Netzsystem kann derart in ein „äquivalentes“ S/T-Netzsystem überführt werden, sodaß deren Erreichbarkeitsgraphen isomorph sind. Eine solche Transformation wird in [BFF+95b] als „Entfaltung“ bezeichnet, aber da der Begriff in dieser Arbeit bereits in einem anderen Zusammenhang Verwendung findet, wird stattdessen die Bezeichnung *Expansion* eingeführt.

Definition 6.1.1 (M-Netzexpansion)
Gegeben sei ein beschränktes M-Netzsystem $\Upsilon = (N, M_0)$ mit $N = (S, T, F, \iota)$. Das S/T-Netzsystem $\Sigma = (N_e, M_{0_e})$ mit $N_e = (S_e, T_e, F_e, W_e)$ wird als *Expansion* von Υ bezeichnet, falls

$$\begin{aligned}
S_e &= \{s^z \mid s \in S \wedge z \in \iota(s)\} \\
T_e &= \{t^\varsigma \mid t \in T \wedge (\varsigma \text{ ist ein Schaltmodus von } t)\} \\
F_e &= \{(s^z, t^\varsigma) \mid (s,t) \in F \wedge z \in \iota((s,t))[\varsigma]\} \cup \\
&\quad \{(t^\varsigma, s^z) \mid (t,s) \in F \wedge z \in \iota((t,s))[\varsigma]\} \\
W_e((s^z, t^\varsigma)) &= (\iota((s,t))[\varsigma])(z) \quad \text{für jedes Tupel } (s^z, t^\varsigma) \in F_e \\
W_e((t^\varsigma, s^z)) &= (\iota((t,s))[\varsigma])(z) \quad \text{für jedes Tupel } (t^\varsigma, s^z) \in F_e \\
M_{0_e}(s^z) &= (M_0(s))(z) \quad \text{für jedes } s^z \in S_e
\end{aligned}$$

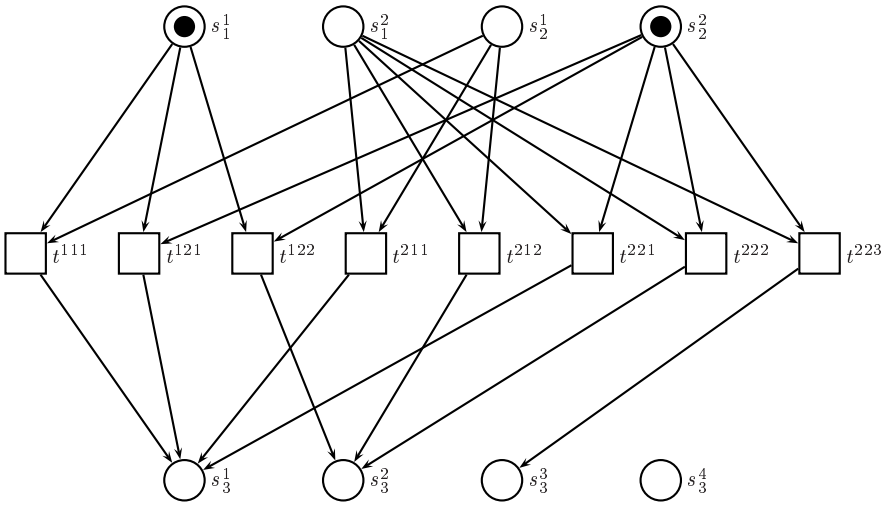
Abbildung 6.1 auf der nächsten Seite zeigt die Expansion für das M-Netzsystem aus Abbildung 6.2 (a) auf Seite 183. Das M-Netzsystem besitzt die beiden Schaltmodi

$$\varsigma_1(v) = \begin{cases} 1 & \text{falls } v = v_1 \\ 2 & \text{falls } v = v_2 \\ 1 & \text{falls } v = v_3 \end{cases} \quad \text{und} \quad \varsigma_2(v) = \begin{cases} 1 & \text{falls } v = v_1 \\ 2 & \text{falls } v = v_2 \\ 2 & \text{falls } v = v_3 \end{cases}$$

für welche die Transition t unter der Anfangsmarkierung $M_0 = \{s_1^1, s_2^2\}$ aktiviert ist. Die Expansion weist genau die beiden Schaltfolgen

$$\{s_1^1, s_2^2\} \xrightarrow{t^{121}} \{s_3^1\} \quad \text{und} \quad \{s_1^1, s_2^2\} \xrightarrow{t^{122}} \{s_3^2\}$$

Abbildung 6.1 *Expansion des streng sicheren M-Netzsystems aus Abbildung 6.2 (a) auf Seite 183. Der Schaltmodus ς einer Schaltinstanz t^{ς} wird als Folge $\varsigma(v_1)\varsigma(v_2)\varsigma(v_3)$ angegeben.*



auf, d.h., sie modelliert das M-Netzsystem verhaltensgetreu. Folglich könnte der Nachweis temporallogischer Eigenschaften von M-Netzsystemen unter Verwendung von deren Expansionen durchgeführt werden. Diese Vorgehensweise besäße jedoch den entscheidenden Nachteil, daß die Transformation von M-Netzsystemen in deren Expansionen normalerweise sehr große S/T-Netzsysteme erzeugt, für die der Nachweis temporallogischer Eigenschaften in vielen Fällen nicht mehr mit vertretbarem Speicher- und Zeitaufwand durchgeführt werden könnte. Oftmals weisen diese Netzsysteme sogar eine unnötige Größe in der Hinsicht auf, daß sie viele unerreichbare Stellen und tote Transitionen enthalten. Dieses kann darin begründet liegen, daß die Typen der M-Netzstellen üblicherweise Überapproximationen darstellen und die Transitionen folglich viele Schaltmodi aufweisen, von denen aber in Abhängigkeit von der Anfangsmarkierung eines M-Netzsystems nur wenige als Schaltinstanzen innerhalb der möglichen Schaltsequenzen des M-Netzsystems vorkommen. Beispielsweise sind in der Expansion aus Abbildung 6.1 nur vier der acht Stellen erreichbar, und lediglich zwei der acht Transitionen können ausgeführt werden. Unter diesem Hintergrund ist es sinnvoll, die Verifikation temporallogischer Eigenschaften unmittelbar unter Einbeziehung der M-Netzsysteme vorzunehmen und den Umweg über deren Expansionen zu vermeiden.

6.1.2 M-Netzentfaltungen

Die Entfaltungen von M-Netzsystemen werden in analoger Weise zu den Entfaltungen von S/T-Netzsystemen (Abschnitt 3.1 auf Seite 37) definiert. Sofern Begriffe und Konzepte aus Abschnitt 3.1 auf Seite 37 im folgenden nicht neu definiert werden, lassen sie

sich direkt auf M-Netzsysteme übertragen und werden nicht mehr explizit erwähnt.

Definition 6.1.2 (Mengen $\mathcal{B}$ und $\mathcal{E}$ von Bezeichnern)
Gegeben sei ein M-Netz (S, T, F, ι). Die Mengen $\mathcal{B}$ und $\mathcal{E}$ von Bezeichnern sind wie folgt induktiv definiert:

- $\perp \in \mathcal{E}$, wobei $\perp$ ein spezielles Symbol darstellt
- falls $e \in \mathcal{E}$, dann $((s, z), e) \in \mathcal{B}$ für jedes $s \in S$ und jedes $z \in \iota(s)$
- falls $\emptyset \subset B' \subseteq \mathcal{B}$, dann $((t, \varsigma), B') \in \mathcal{E}$ für jedes $t \in T$ und jeden Schaltmodus ς von t

Ein Branching-Prozeß eines M-Netzsystems (N, M_0) wird nun mittels zweier Teilmengen $B \subseteq \mathcal{B}$ und $E \subseteq \mathcal{E}$ dieser Bezeichner definiert. Dabei werden die Bezeichner der Mengen $\mathcal{B}$ und $\mathcal{E}$ folgendermaßen verwendet: Die bezüglich der Kausalrelation $<$ minimalen Bedingungen (die Anfangsmarkierung) des Branching-Prozesses werden (wird) durch die folgende Bedingung festgelegt:

$$\forall s \in S\ \forall z \in Z\colon ((s, z), \perp) \in B \Leftrightarrow z \in M_0(s)$$

Die Beschriftung einer Bedingung $((s, z), e) \in B$ lautet (s, z), und e stellt deren einziges Eingangsereignis dar. Entsprechendes gilt für ein Ereignis $((t, \varsigma), B') \in E$, das einen Repräsentanten für die Schaltinstanz (t, ς) darstellt, und dessen Vorbereich aus der Menge B' von Bedingungen besteht.
Mit dieser Notation können Branching-Prozesse vollständig durch ihre Mengen von Bedingungen und Ereignissen als Tupel (B, E) mit $B \subseteq \mathcal{B}$ und $E \subseteq \mathcal{E}$ beschrieben werden.

Definition 6.1.3 (Branching-Prozeß)
Gegeben sei ein M-Netzsystem $\Upsilon = (N, M_0)$ mit $N = (S, T, F, \iota)$ sowie der streng 1-beschränkten Anfangsmarkierung

$$M_0 = \{(s_1, z_1), \ldots, (s_n, z_n)\}$$

und einfachen Kantengewichten, d.h.,

$$\forall f \in F\colon |\iota(f)| = 1$$

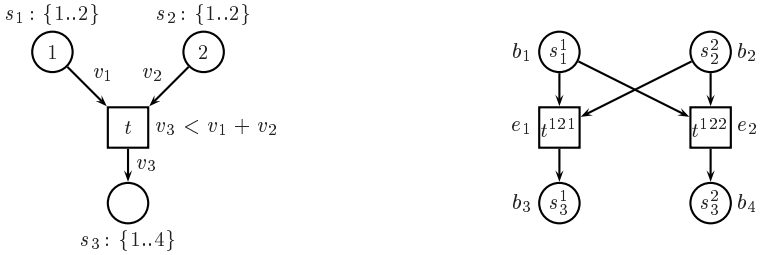
Die Menge der endlichen *Branching-Prozesse* von Υ ist induktiv definiert durch:

- $(\{((s_1, z_1), \perp), \ldots, ((s_n, z_n), \perp)\}, \emptyset)$ ist ein Branching-Prozeß von Υ.
- Falls (B, E) ein Branching-Prozeß von Υ ist und $t \in T$ eine Transition, ς einen Schaltmodus von t und $B' \subseteq B$ eine Co-Menge bezeichnen, sodaß in B' für jedes $s \in {}^\bullet t$ und jedes $v \in \iota((s, t))$ genau ein Element mit der Beschriftung $(s, \varsigma(v))$ vorkommt, dann stellt auch

  $$(B \cup \{((s', \varsigma(v')), e) \mid s' \in t^\bullet \wedge v' \in \iota((t, s'))\}, E \cup \{e\}) \quad \text{mit} \quad e = ((t, \varsigma), B')$$

  einen Branching-Prozeß von Υ dar. Das Ereignis e wird als *mögliche Erweiterung* von (B, E) bezeichnet, falls $e \notin E$.

Abbildung 6.2 *Streng sicheres M-Netzsystem (a) und dessen endliches, vollständiges Präfix (b). Der Schaltmodus ς einer Schaltinstanz t^{ς} wird als Folge $\varsigma(v_1)\varsigma(v_2)\varsigma(v_3)$ angegeben.*



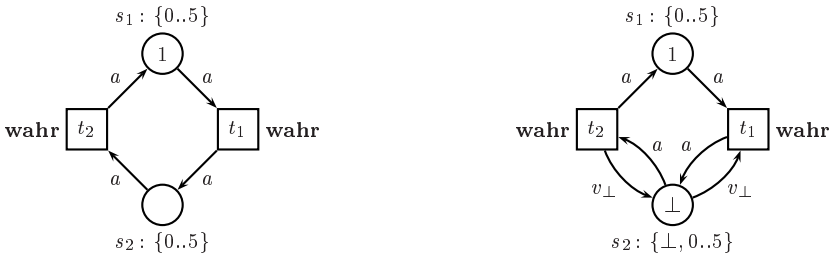
(a) (b)

Die Menge aller Branching-Prozesse eines M-Netzsystems erhält man durch die Vereinigung beliebiger endlicher oder unendlicher Mengen von Branching-Prozessen, da diese unter komponentenweiser Vereinigung von Bedingungen und Ereignissen abgeschlossen sind. Zudem garantiert die Abgeschlossenheit unter Vereinigung die Existenz eines eindeutigen, maximalen Branching-Prozesses, der als *Entfaltung* eines M-Netzsystems bezeichnet wird.
Die Definitionen für Konfigurationen, Schnitte, adäquate Ordnungen, Terminalereignisse und endliche, vollständige Präfixe können direkt aus Abschnitt 3.1 auf Seite 37 übernommen werden.
Abbildung 6.2 zeigt ein streng sicheres M-Netzsystem (a) und dessen endliches, vollständiges Präfix (b). Das M-Netzsystem besitzt die beiden Schaltmodi

$$\varsigma_1(v) = \begin{cases} 1 & \text{falls } v = v_1 \\ 2 & \text{falls } v = v_2 \\ 1 & \text{falls } v = v_3 \end{cases} \quad \text{und} \quad \varsigma_2(v) = \begin{cases} 1 & \text{falls } v = v_1 \\ 2 & \text{falls } v = v_2 \\ 2 & \text{falls } v = v_3 \end{cases}$$

für welche die Transition t unter der Anfangsmarkierung $M_0 = \{s_1^1, s_2^2\}$ aktiviert ist. Genau diese beiden Schaltmöglichkeiten werden in dem endlichen, vollständigen Präfix durch die Ereignisse e_1 und e_2 repräsentiert, die den Schaltinstanzen t^{ς_1} und t^{ς_2} entsprechen.
Der in [HKK02] vorgeschlagene parallele Algorithmus zur Präfixerzeugung sicherer S/T-Netzsysteme wurde in [KK03] auf streng sichere M-Netzsysteme erweitert. Dabei wurde gezeigt, daß die endlichen, vollständigen Präfixe eines streng sicheren M-Netzsystems und dessen Expansion zueinander isomorph sind. Abbildung 6.2 (b) auf dieser Seite zeigt sowohl das endliche, vollständige Präfix für das M-Netzsystem aus Abbildung 6.2 (a) als auch das endliche, vollständige Präfix für dessen Expansion aus Abbildung 6.1 auf Seite 181. Folglich können alle Verifikationsmethoden, die auf der Präfixbildung von S/T-Netzsystemen basieren, auch unmittelbar zur Verifikation von M-Netzsystemen herangezogen werden.

Abbildung 6.3 *Streng sicheres M-Netzsystem* Υ *(a) und dessen Modifikation* Υ' *(b)*



6.1.3 LTL auf streng sicheren M-Netzsystemen

Wie bei S/T-Netzsystemen (Abschnitt 5.1.2 auf Seite 116) kann LTL auch auf M-Netzsystemen in einer *aktions-* und *zustandsbasierten* Variante interpretiert werden. Im weiteren Verlauf des Kapitels wird aber nur die zustandsbasierte Variante betrachtet, weshalb der aktionsbasierten Variante an dieser Stelle keine weitere Beachtung geschenkt wird. In der *zustandsbasierten* Variante bestehen die atomaren Propositionen einer LTL-Eigenschaft aus booleschen Ausdrücken, die über der Stellenmenge eines M-Netzsystems und deren Typen gebildet werden, d.h., für ein M-Netzsystem $\Upsilon = (N, M_0)$ mit $N = (S, T, F, \iota)$ sei

$$\Pi \subseteq \mathcal{B}_{exp}(S \cup (Z \setminus \{\perp\}))|_{\{=,\neq,\leq,<,\geq,>\}}$$

gegeben, wobei die Variablenmenge S der booleschen Ausdrücke mit der Stellenmenge S des M-Netzsystems identifiziert wird, die Wertemenge Z die Vereinigungsmenge aller Stellentypen bezeichnet und $\perp$ einen speziellen Wert darstellt, dessen Bedeutung noch innerhalb der nachfolgenden Ausführungen verdeutlicht wird. Um zu entscheiden, ob für eine unendliche Schaltfolge $M_0 \xrightarrow{t_1^{S1}} M_1 \xrightarrow{t_2^{S2}} M_2 \ldots$ von Υ die unendliche Zustandsfolge

$$\xi = M_0 M_1 M_2 \ldots$$

eine LTL-Eigenschaft φ erfüllt, wird ξ zunächst in ein ω-Wort über dem Alphabet $2^{\Pi(\varphi)}$ überführt. Dabei wird für jede Markierung M_i von ξ die Menge der atomaren Propositionen ermittelt, die unter M_i zu **wahr** evaluiert werden können.
Abbildung 6.3 (a) auf dieser Seite zeigt ein streng sicheres M-Netzsystem, für das untersucht werden soll, ob es die LTL-Eigenschaft

$$\varphi = \Box\Diamond(s_2 < 1) \tag{6.1.1}$$

(„s_2 *wird immer wieder einen Wert kleiner als 1 aufweisen*“) erfüllt. Die Sprache $L(\varphi)$ enthält alle ω-Wörter über dem Alphabet $2^{\Pi(\varphi)}$, die φ erfüllen. Die Wörter der Sprache $L(\varphi)$ weisen alle die Form

$$((\emptyset + \{(s_2 < 1)\})^*\{(s_2 < 1)\})^\omega$$

auf. Das M-Netzsystem Υ enthält

$$\xi = (\{s_1^1\}\{s_2^1\})^\omega$$

als einzige unendliche Zustandsfolge, die nun in ein ω-Wort über dem Alphabet $2^{\Pi(\varphi)}$ überführt werden muß. Die Evaluierung einer atomaren Proposition $\pi \in \Pi(\varphi)$ unter einer Markierung $M \in \mathcal{R}(M_0)$ zu **wahr** oder **falsch** geschieht mittels einer Abbildung

$$\varsigma_M : Var(\Pi(\varphi)) \to Z,$$

welche die Eigenschaft

$$\forall s \in Var(\Pi(\varphi))\ \forall z \in Z\colon \varsigma_M(s) = z \Leftrightarrow z \in M(s)$$

aufweist.[1] ς_M weist also jeder in einer atomaren Proposition vorkommenden Variablen s_i den Wert der entsprechenden Stelle s_i unter der Markierung M zu. Nun muß nur noch getestet werden, ob $\varsigma_M \models \pi$.
Bezüglich des M-Netzsystems Υ (Abbildung 6.3 (a) auf der vorherigen Seite) und der LTL-Eigenschaft φ (Zeile 6.1.1 auf der vorherigen Seite) ergibt sich jedoch das Problem, daß beispielsweise $\varsigma_M \models (s_2 < 1)$ unter der Markierung $M = \{s_1^1\}$ nicht entschieden werden kann, da die Stelle s_2 unter M unmarkiert ist. Dieser Mißstand kann jedoch behoben werden, indem die Wertemenge Z um den speziellen Wert $\perp$ erweitert wird, der kennzeichnet, daß eine Stelle unmarkiert ist. Das M-Netzsystem Υ wird also ähnlich zu dem Konzept der komplementären Stellen bei S/T-Netzsystemen dahingehend modifiziert, daß alle Stellen, die als Variablen innerhalb der atomaren Propositionen von φ vorkommen, genau dann in dem modifizierten Netzsystem den Wert $\perp$ aufweisen, wenn sie im ursprünglichen Netzsystem unmarkiert sind.
Das modifizierte M-Netzsystem Υ' ist in Abbildung 6.3 (b) auf der vorherigen Seite dargestellt. Die neu hinzugefügten Kanten werden mit der speziellen Variablen $v_\perp$ beschriftet, die symbolisieren soll, daß der Wert $\perp$ ausschließlich an $v_\perp$ gebunden werden darf. Mit dieser Maßnahme wird verhindert, daß der Wert $\perp$ fälschlicherweise anderen Variablen zugewiesen wird und dadurch das Systemverhalten in unerwünschter Weise verändert wird. Deshalb wird gefordert, daß für jede Transition t von Υ' und jeden Schaltmodus ς von t gilt:

$$\forall v \in V(t)\colon \varsigma(v) = \perp \Leftrightarrow v = v_\perp$$

Desweiteren wird per Definition festgelegt, daß eine atomare Proposition π unter einer Markierung M zu **falsch** evaluiert wird, falls eine in π vorkommende Variable unter ς_M den Wert $\perp$ zugewiesen bekommt, d.h., die entsprechende Stelle unter M unmarkiert ist. Dazu muß jedoch die Erfüllbarkeitsrelation $\models$ aus Definition 2.1.9 auf Seite 25 an einer

[1] Für streng sichere M-Netzsysteme gilt, daß für jede Stelle $s \in S$ und jede erreichbare Markierung $M \in \mathcal{R}(M_0)$ gilt: $|M(s)| \leq 1$

Stelle geändert werden. Die modifizierte Erfüllbarkeitsrelation wird mit $\models_\perp$ bezeichnet, und sie unterscheidet sich von $\models$ lediglich in

$$\varsigma \models_\perp (a_1\ op\ a_2) \quad \Leftrightarrow \quad (\forall v \in Var(a_1\ op\ a_2)\colon \varsigma(v) \neq \perp) \wedge (a_1[\varsigma]\ op\ a_2[\varsigma]) \qquad (6.1.2)$$

für $op \in \{=, \neq, \leq, <, \geq, >\}$.
Mittels der Erfüllbarkeitsrelation $\models_\perp$ kann nun die Menge aller unter einer Markierung $M \in \mathcal{R}(M_0)$ gültigen atomaren Proposition unter Verwendung einer Abbildung

$$\nu\colon (\mathcal{M}_f(Z))^S \to 2^{\Pi(\varphi)}$$

ermittelt werden, sodaß

$$\nu(M) = \{\pi \in \Pi(\varphi) \mid \varsigma_M \models_\perp \pi\}$$

Die Erweiterung von ν auf ω-Wörter und die Definition der Erfüllbarkeitsrelation $\models^\nu_{LTL}$ können direkt aus Abschnitt 5.1.2 auf Seite 116 übernommen werden.
Wird die Funktion ν auf die einzige unendliche Zustandsfolge ξ' des M-Netzsystems Υ' aus Abbildung 6.3 (b) auf Seite 184 angewendet, so erhält man

$$\nu(\xi') = (\nu(\{s_1^1, s_2^\perp\})\nu(\{s_2^1\}))^\omega = (\emptyset\emptyset)^\omega = \emptyset^\omega \notin L(\varphi)$$

Daraus folgt, daß das M-Netzsystem Υ die LTL-Eigenschaft φ nicht erfüllt.
Wie bereits erwähnt wurde, wird eine atomare Proposition π unter einer Markierung M zu **falsch** evaluiert, falls in π eine Variable vorkommt, deren entsprechende Stelle unter M unmarkiert ist. Allerdings ist es auch wünschenswert, Aussagen in Form atomarer Propositionen darüber zu treffen, daß eine Stelle markiert oder unmarkiert ist. Aus diesem Grund wird die Menge Π der atomaren Propositionen um eine spezielle Menge

$$\Pi_\perp = \{(s = \perp) \mid s \in S\}$$

von atomaren Propositionen erweitert, wobei für eine Stelle s die Proposition $(s = \perp)$ genau dann unter einer Markierung M gelten soll, falls $\perp \in M(s)$. Dazu wird die Definition der Erfüllbarkeitsrelation $\models_\perp$ in Zeile 6.1.2 derart geändert, sodaß

$$\varsigma \models_\perp (a_1\ op\ a_2) \quad \Leftrightarrow \quad \begin{cases} \varsigma(s) = \perp & \text{falls } (a_1\ op\ a_2) = (s = \perp) \in \Pi_\perp \\ (\forall v \in Var(a_1\ op\ a_2)\colon \varsigma(v) \neq \perp) \wedge (a_1[\varsigma]\ op\ a_2[\varsigma]) & \text{sonst} \end{cases}$$

für $op \in \{=, \neq, \leq, <, \geq, >\}$.
Unter Verwendung der neu eingeführten atomaren Propositionen aus $\Pi_\perp$ und der modifizierten Erfüllbarkeitsrelation $\models_\perp$ kann nun beispielsweise untersucht werden, ob das M-Netzsystem Υ aus Abbildung 6.3 (a) auf Seite 184 die LTL-Eigenschaft

$$\psi = \Diamond\Box((s_2 = \perp) \vee (s_2 \geq 1))$$

(„*irgendwann wird s_2 immer unmarkiert oder größer gleich 1 sein*“) erfüllt. Die Sprache $L(\psi)$ enthält alle ω-Wörter über dem Alphabet $2^{\Pi(\psi)}$, die sich nach dem Muster

$$(2^{\Pi(\psi)})^*(\{(s_2 = \bot)\} + \{(s_2 \geq 1)\} + \{(s_2 = \bot), (s_2 \geq 1)\})^\omega$$

ergeben. Für die einzige unendliche Zustandsfolge ξ' des modifizierten M-Netzsystems Υ' aus Abbildung 6.3 (b) auf Seite 184 gilt:

$$\nu(\xi') = (\nu(\{s_1^1, s_2^\bot\})\nu(\{s_2^1\}))^\omega = (\{(s_2 = \bot)\}\{(s_2 \geq 1)\})^\omega \in L(\psi)$$

Daraus folgt unmittelbar, daß das M-Netzsystem Υ die LTL-Eigenschaft ψ erfüllt.

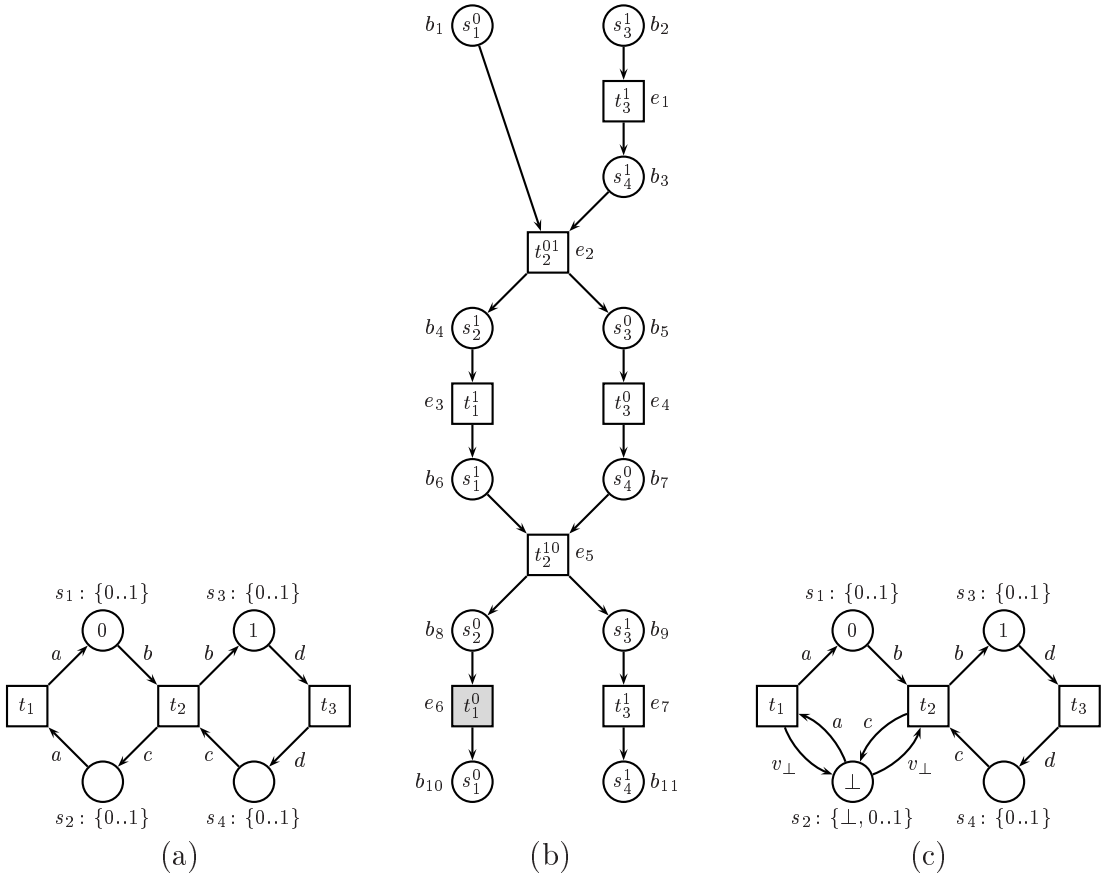
6.2 Tableau-Ansatz zum LTL-X Model-Checking

Der in diesem Abschnitt vorgestellte Ansatz zur Verifikation von temporallogischen Eigenschaften streng sicherer M-Netzsysteme basiert auf den in [EH01, Hel02] eingeführten Methoden zum Model-Checking sicherer S/T-Netzsysteme (siehe auch Abschnitt 5.1.4 auf Seite 119). Als das die Ausführungen in diesem Abschnitt begleitende Beispiel wird das M-Netzsystem Υ aus Abbildung 6.4 (a) auf der nächsten Seite herangezogen. Das M-Netzsystem modelliert einen Speicher, der in den beiden Speicherstellen s_2 und s_4 insgesamt höchstens zwei Werte vom Typ $\{0..1\}$ hinterlegen kann. Dabei arbeitet der Speicher nach dem FIFO-Prinzip („*first in – first out*“), d.h., die Werte werden in genau derselben Reihenfolge wieder aus dem Speicher ausgelesen, in der sie vorher reingeschrieben wurden. Die möglichen Ausführungssequenzen von Υ werden kompakt und übersichtlich durch dessen endliches, vollständiges Präfix repräsentiert, das in Abbildung 6.4 (b) auf der nächsten Seite zu sehen ist.
Das M-Netzsystem Υ besteht aus 4 Stellen und 3 Transitionen, wohingegen die Expansion von Υ bereits 8 Stellen und 8 Transitionen aufweist. Υ stellt ein klassisches Beispiel für ein M-Netzsystem dar, dessen Expansion exponentiell mit Änderung der Speicherkapazität und/oder den Stellentypen wächst. Beispielsweise bewirkt eine Erweiterung der Stellentypen von Υ auf den Wertebereich $\{0..150\}$, daß die Expansion von Υ 604 Stellen und 23103 Transitionen aufweist, wohingegen das endliche, vollständige Präfix von Υ nach wie vor aus 11 Bedingungen und 7 Ereignissen besteht (Abbildung 6.4 (b) auf der nächsten Seite). Darüberhinaus ist es für größere Systeme nicht mehr möglich, deren Expansionen in vertretbarem Zeitaufwand zu erzeugen. Folglich besteht ein erheblicher Vorteil des in den folgenden Abschnitten behandelten Verfahrens im Vergleich zu dem Ansatz von Esparza und Heljanko [EH01, Hel02] darin, daß die M-Netzsysteme direkt entfaltet werden ohne den Umweg über deren Expansionen gehen zu müssen.
Begleitend zu den Ausführungen in diesem Abschnitt soll gezeigt werden, daß das M-Netzsystem Υ aus Abbildung 6.4 (a) die LTL-X-Eigenschaft

$$\varphi = \Diamond\Box((s_2 = \bot) \vee (s_2 \neq 0)) \tag{6.2.1}$$

(„*irgendwann wird s_2 immer unmarkiert oder ungleich Null sein*“) nicht erfüllt. Dazu wird Υ zunächst auf die in Abschnitt 6.1.3 auf Seite 184 beschriebene Art und Weise

Abbildung 6.4 *Streng sicheres M-Netzsystem Υ eines Datenspeichers der Kapazität 2 (a). Bei Transitionen, die mit dem Wächter* ***wahr*** *beschriftet sind, wird dieser aus Gründen der Übersichtlichkeit weggelassen. Endliches, vollständiges Präfix von Υ, wobei Terminalereignisse grau unterlegt sind (b). Stelleninstanzen werden als s^z und Schaltinstanzen als $t_1^{\varsigma(a)}$, $t_2^{\varsigma(b)\varsigma(c)}$ und $t_3^{\varsigma(d)}$ notiert. Für LTL-X Model-Checking modifiziertes M-Netzsystem Υ' (c).*



modifiziert und in das M-Netzsystem Υ' aus Abbildung 6.4 (c) überführt. Nun liefert beispielsweise die Schaltfolge

$$M_0 \xrightarrow{(t_3^1 t_2^{01\perp} t_1^{1\perp} t_3^0 t_2^{10\perp} t_1^{0\perp})^\omega}$$

einen unendlichen Ablauf von Υ', der die Eigenschaft φ verletzt, da die Stelle s_2 durch das Schalten der Transitionsinstanz $t_2^{10\perp}$ jedesmal den Wert 0 erhält. Dabei ist der Schaltmodus einer Transition t jeweils als Folge $(\varsigma(a)\varsigma(b)\varsigma(c)\varsigma(d)\varsigma(v_\perp))|_{V(t)}$ notiert.
Um einen solchen, die LTL-X-Eigenschaft φ verletzenden Ablauf automatisch zu erkennen, wird zunächst ein Büchiautomat $\mathcal{A}_{\neg\varphi}$ erstellt, der genau diejenigen Abläufe

Abbildung 6.5 *Büchiautomat $\mathcal{A}_{\neg\varphi}$, der für die Verifikation der LTL-X-Eigenschaft $\varphi = \Diamond\Box\,((s_2 = \bot) \vee (s_2 \neq 0))$ erzeugt wird (a), und das mit $\mathcal{A}_{\neg\varphi}$ assoziierte M-Netzsystem $\Upsilon_{\neg\varphi}$ (b). Aus Gründen der Übersichtlichkeit werden die Transitionen $t_{(q_i,x,q_j)}$ als t_{i_j} notiert, wobei x in die Transitionswächter hineinkodiert ist.*

(a) (b)

akzeptiert, die φ nicht erfüllen, d.h., $L(\mathcal{A}_{\neg\varphi}) = L(\neg\varphi)$. Der Büchiautomat $\mathcal{A}_{\neg\varphi}$ mit

$$\begin{aligned}\neg\varphi &= \neg\Diamond\Box((s_2 = \bot) \vee (s_2 \neq 0)) \\ &= \Box\Diamond\neg((s_2 = \bot) \vee (s_2 \neq 0)) \\ &= \Box\Diamond(\neg(s_2 = \bot) \wedge \neg(s_2 \neq 0))\end{aligned}$$

(„s_2 *wird immer irgendwann mit dem Wert 0 belegt sein*“) ist in Abbildung 6.5 (a) dargestellt, wobei die Funktion ζ (Definition 5.1.5 auf Seite 120) auf die Kantenbeschriftungen angewendet und $\mathcal{A}_{\neg\varphi}$ anschließend noch optimiert wurde. Der Büchiautomat $\mathcal{A}_{\neg\varphi}$ wird mit dem M-Netzsystem $\Upsilon_{\neg\varphi}$ aus Abbildung 6.5 (b) assoziiert. Dabei gibt es für jeden Zustand q von $\mathcal{A}_{\neg\varphi}$ eine entsprechende Stelle s_q in $\Upsilon_{\neg\varphi}$. Alle Stellen weisen den Typ $\{\bullet\}$ auf, wobei ausschließlich s_{q_0} anfänglich markiert ist. Für jeden Übergang (q, x, q') von $\mathcal{A}_{\neg\varphi}$ weist $\Upsilon_{\neg\varphi}$ eine Transition $t_{(q,x,q')}$ auf, wobei s_q deren Eingangs- und $s_{q'}$ deren Ausgangsstelle bezeichnen. Der Wächter von $t_{(q,x,q')}$ wird durch $\zeta(x)$ beschrieben. Die Kanten von $\Upsilon_{\neg\varphi}$ können mit einer beliebigen Variablen beschriftet werden. Aus Gründen der Übersichtlichkeit soll jedoch eine neue Variable $v_\bullet$ eingeführt werden, die noch nicht innerhalb der Kantenbeschriftungen des zu verifizierenden M-Netzsystems Υ_S vorkommt. Damit $\Upsilon_{\neg\varphi}$ die einzelnen Schritte von Υ_S überwachen kann, wird in ähnlicher Weise, wie es in [EH01, Hel02] für S/T-Netzsysteme beschrieben ist, ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ aufgestellt. Der Standardansatz zur Produktnetzbildung zwischen einem M-Netzsystem Υ_S und einem den Büchiautomaten repräsentierenden M-Netzsystem $\Upsilon_{\neg\varphi}$ besteht darin, diese an allen Transitionen zu synchronisieren. Somit ist $\Upsilon_{\neg\varphi}$ in der Lage, jeden einzelnen Schritt von Υ_S zu überwachen. Ein erheblicher Nachteil dieser Vorgehensweise besteht jedoch in der daraus resultierenden vollständigen Sequentialisierung des Produktnetzsystems $\Upsilon_{S\times\neg\varphi}$. Da jedoch die Stärken von entfaltungsbasierten Techniken gerade in der Ausnutzung von Nebenläufigkeit liegen, wäre ein solches Produktnetzsystem für entfaltungsbasierte Verifikationsmethoden äußerst ungeeignet. Aus diesem Grund wird der Synchronisationsansatz für sichere S/T-Netzsysteme aus [EH01, Hel02] auf die M-Netzebene angehoben. Unter dem Hintergrund, vorhandene Nebenläufigkeit

in Υ_S während der Verifikation ausnutzen zu können, werden Υ_S und $\Upsilon_{\neg\varphi}$ lediglich an solchen Transitionen synchronisiert, deren Schalten die Werte der in den atomaren Propositionen der zu verifizierenden Formel φ vorkommenden Stellen beeinflussen. Alle anderen Transitionen bleiben von der Synchronisation unberührt und können gegebenenfalls auch weiterhin nebenläufig ausgeführt werden. Dieses bedeutet bezüglich des M-Netzsystems Υ aus Abbildung 6.4 (a) auf Seite 188 und der LTL-X-Eigenschaft φ aus Zeile 6.2.1 auf Seite 187, daß lediglich die Transitionen t_1 und t_2 von Υ an der Synchronisation mit $\Upsilon_{\neg\varphi}$ (Abbildung 6.5 (b) auf der vorherigen Seite) teilnehmen, da nur deren Schalten den Wert der Stelle s_2 ändern kann.
Da die Stellennamen des zu verifizierenden M-Netzsystems als Variablen innerhalb der atomaren Propositionen der LTL-X-Eigenschaft φ verwendet werden und folglich auch als Variablen innerhalb der Transitionswächter von $\Upsilon_{\neg\varphi}$ vorkommen, wird die Variablenmenge V um die Menge

$$\{v_\bullet\} \cup \{s \mid s \in S\}$$

erweitert, wobei S die Stellenmenge des zu verifizierenden M-Netzsystems bezeichnet und ohne Einschränkung der Allgemeinheit angenommen werden kann, daß S zur bisherigen Variablenmenge V disjunkt ist.
Da die Definition des Produktnetzsystems $\Upsilon_{S\times\neg\varphi}$ in [SK04] eher semiformal angegeben wurde, soll an dieser Stelle nun eine formalere Definition erfolgen.

Definition 6.2.1 (Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$)
Gegeben seien ein streng sicheres M-Netzsystem $\Upsilon_S = (N_S, M_{0_S})$ mit $N_S = (S_S, T_S, F_S, \iota_S)$ und ein Büchiautomat $\mathcal{A}_{\neg\varphi} = (Q, 2^{\Pi(\varphi)}, \delta, q_0, Q_E)$ mit $L(\mathcal{A}_{\neg\varphi}) = L(\neg\varphi)$. Das Tupel $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ heißt *Produktnetzsystem*, falls $N = (S, T, F, \iota)$ ein M-Netz beschreibt, wobei

- $S = S_V \cup S_H \cup S_B \cup \{s_s, s_f\}$, wobei

$$\begin{aligned} S_V &= \{s \mid s \in S_S \cap Var(\Pi(\varphi))\} \\ S_H &= S_S \setminus S_V \\ S_B &= \{s_q \mid q \in Q\} \end{aligned}$$

  Für eine Stelle $s \in S$ gilt bezüglich ihres Typs:

$$\iota(s) = \begin{cases} \iota_S(s) \cup \{\bot\} & \text{falls } s \in S_V \\ \{\bullet\} & \text{falls } s \in S_B \cup \{s_s, s_f\} \\ \iota_S(s) & \text{sonst} \end{cases}$$

  S_V und S_H bezeichnen die Mengen der *sichtbaren* („*visible*“) bzw. *unsichtbaren* („*hidden*“) *Stellen*. S_B beschreibt die Menge der *Büchistellen* und s_s, s_f modellieren einen Scheduler, der gewährleistet, daß der Büchiautomat die einzelnen Systemschritte überwachen kann.

- $T = T_V \cup T_H \cup T_B \cup T_L$, wobei

$$
\begin{aligned}
T_V &= \{t \in T_S \mid \exists s \in S_V : s \in {}^\bullet t \cup t^\bullet\} \\
T_H &= T_S \setminus T_V \\
T_B &= \{t_{(q,x,q')} \mid (q,x,q') \in \delta\} \\
T_L &= \{t_{(q,M)} \mid q \in Q \wedge M \in \mathcal{R}(M_0)|_{\Upsilon_{S \times \neg\varphi} \setminus T_L} \wedge \bullet \in M(s_q) \cap M(s_f) \wedge \\
&\qquad (\nu(M))^\omega \in L(\mathcal{A}_{\neg\varphi}, q)\}
\end{aligned}
$$

  Für eine Transition $t \in T$ gilt bezüglich ihres Wächters:

$$
\iota(t) = \begin{cases} \textbf{wahr} & \text{falls } t \in T_L \\ \zeta(x) & \text{falls } t = t_{(q,x,q')} \in T_B \\ \iota_S(t) & \text{sonst} \end{cases}
$$

  T_V und T_H bezeichnen die Mengen der *sichtbaren* bzw. *unsichtbaren Transitionen*. T_B beschreibt die Menge der *Büchitransitionen*. Dabei bedeutet $t_{(q,x,q')}$, daß die Transition die Marke $\bullet$ von der Stelle s_q abzieht und auf die Stelle $s_{q'}$ legt, falls der Wächter $\zeta(x)$ unter der aktuellen Markierung zu **wahr** evaluiert wird. Die Menge T_L enthält sogenannte *Livelock-Wächter*[2], auf die im weiteren Verlauf noch näher eingegangen wird.
  Ferner wird eine Menge $T_I \subseteq T_B$ von *unendlichen Trace-Wächtern* definiert, die alle Büchitransitionen enthält, deren Schalten die Marke $\bullet$ auf eine Stelle legt, die eine Endstelle des Büchiautomaten repräsentiert, d.h.,

$$T_I = \{t_{(q,x,q')} \in T_B \mid q' \in Q_E\}$$

- $F = F_S \cup F_B \cup F_V \cup F_\perp \cup F_{s_s} \cup F_{s_f} \cup F_L$, wobei

$$
\begin{aligned}
F_B &= \{(s_q, t_{(q,x,q')}) \mid s_q \in S_B \wedge t_{(q,x,q')} \in T_B\} \cup \\
&\qquad \{(t_{(q,x,q')}, s_{q'}) \mid s_{q'} \in S_B \wedge t_{(q,x,q')} \in T_B\} \\
F_V &= \{(s, t_{(q,x,q')}) \mid s \in Var(\iota(t_{(q,x,q')})) \wedge t_{(q,x,q')} \in T_B\} \cup \\
&\qquad \{(t_{(q,x,q')}, s) \mid s \in Var(\iota(t_{(q,x,q')})) \wedge t_{(q,x,q')} \in T_B\} \cup \\
F_\perp &= \{(s,t) \mid s \in S_V \wedge (t,s) \in F_S \wedge (s,t) \notin F_S\} \cup \\
&\qquad \{(t,s) \mid s \in S_V \wedge (s,t) \in F_S \wedge (t,s) \notin F_S\} \\
F_{s_s} &= \{(s_s, t) \mid t \in T_V\} \cup \{(t_{(q,x,q')}, s_s) \mid t_{(q,x,q')} \in T_B\} \\
F_{s_f} &= \{(s_f, t_{(q,x,q')}) \mid t_{(q,x,q')} \in T_B\} \cup \{(t, s_f) \mid t \in T_V\} \\
F_L &= \{(s, t_{(q,M)}) \mid s \in S \wedge t_{(q,M)} \in T_L \wedge M(s) \neq \emptyset\} \cup \\
&\qquad \{(t_{(q,M)}, s) \mid s \in S_H \wedge t_{(q,M)} \in T_L \wedge M(s) \neq \emptyset\}
\end{aligned}
$$

[2]Da es für jeden Zustand $q \in Q$ exponentiell in der Größe von Υ_S viele Transitionen $t_{(q,M)}$ geben kann, werden diese nicht explizit, sondern „On-the-Fly“ während der Präfixbildung erzeugt.

Für eine Kante $f \in F$ gilt bezüglich ihrer Beschriftung:

$$\iota(f) = \begin{cases} \{v_\bullet\} & \text{falls } f \in F_B \cup F_{s_s} \cup F_{s_f} \\ \{v_\bullet\} & \text{falls } (f = (s, t_{(q,M)}) \vee f = (t_{(q,M)}, s)) \in F_L \wedge s \in S_B \cup \{s_s, s_f\} \\ \{s\} & \text{falls } (f = (s, t_{(q,M)}) \vee f = (t_{(q,M)}, s)) \in F_L \wedge s \in S_V \cup S_H \\ \{s\} & \text{falls } (f = (s, t_{(q,x,q')}) \vee f = (t_{(q,x,q')}, s)) \in F_V \\ \{v_\perp\} & \text{falls } f \in F_\perp \\ \iota_S(f) & \text{sonst} \end{cases}$$

Durch F_B wird die Struktur des Teilnetzes festgelegt, das den Büchiautomaten repräsentiert. Die Synchronisation der sichtbaren Stellen mit den Büchitransitionen geschieht durch F_V. Durch $F_\perp$ wird sichergestellt, daß eine sichtbare Stelle unter jeder erreichbaren Markierung des Netzsystems einen definierten Wert besitzt. Mittels F_{s_s} und F_{s_f} werden die Schedulerstellen in das Netzsystem integriert. Zuletzt werden durch F_L noch die Livelock-Wächter mit dem Netzsystem verbunden.

- Für eine Stelle $s \in S$ ist die Anfangsmarkierung durch

$$M_0(s) = \begin{cases} \emptyset & \text{falls } s \in S_B \setminus \{s_{q_0}\} \vee s = s_s \\ \{\bullet\} & \text{falls } s = s_{q_0} \vee s = s_f \\ \{\perp\} & \text{falls } s \in S_V \wedge M_{0_S}(s) = \emptyset \\ M_{0_S}(s) & \text{sonst} \end{cases}$$

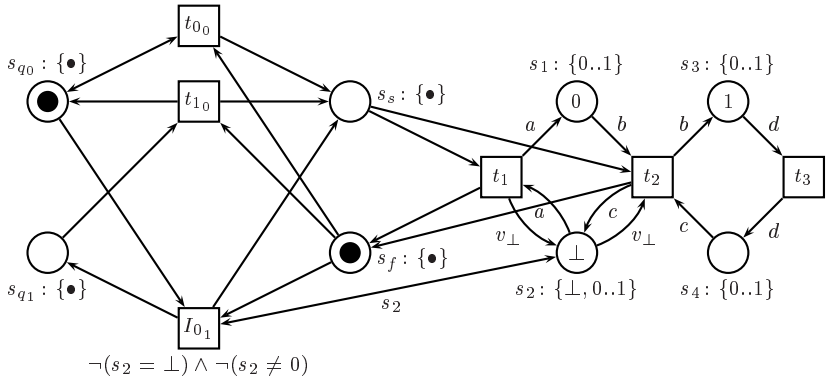
gegeben. Die Stelle s_f muß anfänglich markiert sein, damit das den Büchiautomaten repräsentierende M-Netzsystem die Anfangsmarkierung des Ausgangssystems Υ_S überwachen kann.

Damit LTL-X auf dem Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ in der in Abschnitt 6.1.3 auf Seite 184 eingeführten Art und Weise interpretiert wird, muß die Erfüllbarkeitsrelation $\models_\perp$ zur Evaluierung der Transitionswächter herangezogen werden. Darüberhinaus muß für jede Transition t von$\Upsilon_{S\times\neg\varphi}$ und für jeden Schaltmodus ς von t gelten:

$$\forall v \in V(t) \colon \varsigma(v) = \perp \Rightarrow (v = v_\perp \vee v = s \in S_V)$$
$$\forall v \in V(t) \colon v = v_\perp \Rightarrow \varsigma(v) = \perp$$

Abbildung 6.6 auf der nächsten Seite zeigt das Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ für die M-Netzsysteme Υ_S und $\Upsilon_{\neg\varphi}$ aus den Abbildungen 6.4 (a) auf Seite 188 und 6.5 (b) auf Seite 189, wobei die Darstellung aus Gründen der Übersichtlichkeit vereinfacht wurde und nur die zum Verständnis des Beispiels notwendigen Netzbestandteile zu sehen sind. Der Nachweis, daß das M-Netzsystem Υ_S die LTL-X-Eigenschaft φ nicht erfüllt, geschieht wiederum durch die Suche nach illegalen ω-Traces und illegalen Livelocks, deren Definition (Definition 5.1.7 auf Seite 122) ebenfalls auf die M-Netzebene angehoben wird.

Abbildung 6.6 *Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ für die M-Netzsysteme Υ_S und $\Upsilon_{\neg\varphi}$ aus den Abbildungen 6.4 (a) auf Seite 188 und 6.5 (b) auf Seite 189, wobei die Darstellung aus Gründen der Übersichtlichkeit vereinfacht wurde und nur die zum Verständnis des Beispiels notwendigen Netzbestandteile zu sehen sind. Desweiteren wurden auch die Transitionswächter **wahr** und die Kantenbeschriftungen $\{v_\bullet\}$ weggelassen.*



Definition 6.2.2 (Illegale ω-Traces und illegale Livelocks)
Gegeben sei ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$.

- Eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma}$ heißt *illegaler ω-Trace* von $\Upsilon_{S\times\neg\varphi}$, falls σ unendlich viele Schaltinstanzen $t_i^{\varsigma_i} \in \mathcal{I}_T$ mit $t_i \in T_I$ enthält.
- Eine unendliche Schaltfolge $M_0 \xrightarrow{\sigma t^\varsigma} M \xrightarrow{\sigma_1}$ heißt *illegaler Livelock* von $\Upsilon_{S\times\neg\varphi}$, falls $t \in T_L$ und $\sigma_1 \in \{t_i^{\varsigma_i} \in \mathcal{I}_T \mid t_i \in T_H\}^\omega$.

Ein Blick auf das Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ aus Abbildung 6.6 verdeutlicht, daß der illegale ω-Trace

$$t_{0_0}^\bullet (t_3^1 t_2^{\bullet 01\perp} t_{0_0}^\bullet t_1^{\bullet 1\perp} t_{0_0}^\bullet t_3^0 t_2^{\bullet 10\perp} I_{0_1}^{\bullet 0} t_1^{\bullet 0\perp} t_{1_0}^\bullet)^\omega$$

mit $I_{0_1} \in T_I$ ausführbar ist, welcher einem die LTL-X-Eigenschaft φ (Zeile 6.2.1 auf Seite 187) verletzenden Ablauf entspricht. Der Schaltmodus einer Transition t ist jeweils als Folge $(\varsigma(v_\bullet)\varsigma(a)\varsigma(b)\varsigma(c)\varsigma(d)\varsigma(s_2)\varsigma(v_\perp))|_{V(t)}$ notiert.

Satz 6.2.1 (LTL-X Model-Checking von M-Netzsystemen [SK04])
Gegeben seien ein streng sicheres M-Netzsystem Υ_S und eine LTL-X-Eigenschaft φ. Υ_S erfüllt genau dann φ, wenn in $\Upsilon_{S\times\neg\varphi}$ weder illegale ω-Traces noch illegale Livelocks vorkommen.

Beweis: Gegeben seien ein streng sicheres M-Netzsystem $\Upsilon_S = (N_S, M_{0_S})$ mit $N_S = (S_S, T_S, F_S, \iota_S)$, ein Büchiautomat $\mathcal{A}_{\neg\varphi} = (Q, \Gamma, \delta, q_0, Q_E)$ und das Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$. Die Mengen S_V und S_H der sichtbaren und unsichtbaren Stellen sowie die Mengen T_V und T_H der sichtbaren und unsichtbaren Transitionen sind in den Netzsystemen Υ_S und $\Upsilon_{S\times\neg\varphi}$ identisch. Darüberhinaus wird eine

Markierung M von Υ_S bzw. $\Upsilon_{S\times\neg\varphi}$ mit dem Tupel (O,H) bzw. dem 4-Tupel (s_q,s,O,H) identifiziert, wobei $\bullet \in M(s_q)\cap M(s)$ für $s_q \in S_B$, $s \in \{s_s, s_f\}$, $O = M|_{S_V}$ und $H = M|_{S_H}$ gelten.
Zunächst werden alle die LTL-X-Eigenschaft φ verletzenden unendlichen Abläufe von Υ_S in zwei Klassen unterteilt: Abläufe vom *Typ I* enthalten unendlich viele Schaltinstanzen sichtbarer Transitionen, wohingegen in Abläufen vom *Typ II* lediglich endlich viele Schaltinstanzen sichtbarer Transitionen vorkommen. Mit dieser Klassifizierung genügt es, die beiden folgenden Eigenschaften nachzuweisen:

- Es gibt in Υ_S genau dann eine die LTL-X-Eigenschaft φ verletzende Schaltfolge vom Typ I, wenn $\Upsilon_{S\times\neg\varphi}$ einen illegalen ω-Trace aufweist.

  ($\Rightarrow$) Gegeben sei in Υ_S eine die LTL-X-Eigenschaft φ verletzende Schaltfolge

  $$(O_0,H_0) \xrightarrow{t_1^{\varsigma_1}} (O_0,H_1) \xrightarrow{t_2^{\varsigma_2}} \ldots \xrightarrow{t_i^{\varsigma_i}} (O_0,H_i) \xrightarrow{t_{i+1}^{\varsigma_{i+1}}} (O_1,H_{i+1}) \xrightarrow{t_{i+2}^{\varsigma_{i+2}}} (O_1,H_{i+2})$$
  $$\xrightarrow{t_{i+3}^{\varsigma_{i+3}}} \ldots \xrightarrow{t_j^{\varsigma_j}} (O_1,H_j) \xrightarrow{t_{j+1}^{\varsigma_{j+1}}} (O_2,H_{j+1}) \xrightarrow{t_{j+2}^{\varsigma_{j+2}}} (O_2,H_{j+2}) \xrightarrow{t_{j+3}^{\varsigma_{j+3}}} \ldots$$

  vom Typ I, d.h., es gibt unendlich viele $t_i \in T_V$. Da dieser Ablauf von Υ_S die LTL-X-Eigenschaft φ nicht erfüllt, gilt unmittelbar

  $$\nu((O_0,H_0))\nu((O_0,H_1))\ldots\nu((O_0,H_i))\nu((O_1,H_{i+1}))\nu((O_1,H_{i+2}))\ldots$$
  $$\ldots\nu((O_1,H_j))\nu((O_2,H_{j+1}))\nu((O_2,H_{j+2}))\ldots \in L(\mathcal{A}_{\neg\varphi})$$

  Da $\nu((O_i,H_j)) = \nu(O_i)$ für alle $i,j \geq 0$ und LTL-X zudem unter Stottern abgeschlossen ist, gilt auch

  $$\nu(O_0)\nu(O_1)\nu(O_2)\ldots \in L(\mathcal{A}_{\neg\varphi})$$

  Dann gibt es jedoch in $\mathcal{A}_{\neg\varphi}$ einen Ablauf

  $$q_0 \xrightarrow{\nu(O_0)} q_1 \xrightarrow{\nu(O_1)} q_2 \xrightarrow{\nu(O_2)} q_3 \ldots$$

  mit unendlich vielen $q_i \in Q_E$. Aufgrund der Konstruktion des Produktnetzsystems $\Upsilon_{S\times\neg\varphi}$ (Definition 6.2.1 auf Seite 190) besteht zwischen $\Upsilon_{S\times\neg\varphi}$ und $\mathcal{A}_{\neg\varphi}$ die Beziehung, daß $\forall M \in \mathcal{R}(M_0)$ und $\forall q_i,q_j \in Q$ gilt:

  $$\exists(q_i,\nu(M),q_j) \in \delta \Leftrightarrow \exists t_{(q_i,\nu(M),q_j)} \in T_B : \varsigma_M \models_{\perp} \iota(t_{(q_i,\nu(M),q_j)})\ (=\zeta(\nu(M))) \tag{6.2.2}$$

  Daraus folgt jedoch, daß in $\Upsilon_{S\times\neg\varphi}$ der unendliche Ablauf

  $$(s_{q_0},s_f,O_0,H_0) \xrightarrow{t^{\varsigma'}_{(q_0,\nu(O_0),q_1)}} (s_{q_1},s_s,O_0,H_0) \xrightarrow{t_1^{\varsigma_1}} (s_{q_1},s_s,O_0,H_1) \xrightarrow{t_2^{\varsigma_2}} \ldots \xrightarrow{t_i^{\varsigma_i}}$$
  $$(s_{q_1},s_s,O_0,H_i) \xrightarrow{t_{i+1}^{\varsigma_{i+1}}} (s_{q_1},s_f,O_1,H_{i+1}) \xrightarrow{t^{\varsigma''}_{(q_1,\nu(O_1),q_2)}} (s_{q_2},s_s,O_1,H_{i+1}) \xrightarrow{t_{i+2}^{\varsigma_{i+2}}}$$
  $$(s_{q_2},s_s,O_1,H_{i+2}) \xrightarrow{t_{i+3}^{\varsigma_{i+3}}} \ldots \xrightarrow{t_j^{\varsigma_j}} (s_{q_2},s_s,O_1,H_j) \xrightarrow{t_{j+1}^{\varsigma_{j+1}}} (s_{q_2},s_f,O_2,H_{j+1})$$
  $$\xrightarrow{t^{\varsigma'''}_{(q_2,\nu(O_2),q_3)}} (s_{q_3},s_s,O_2,H_{j+1}) \xrightarrow{t_{j+2}^{\varsigma_{j+2}}} (s_{q_3},s_s,O_2,H_{j+2}) \xrightarrow{t_{j+3}^{\varsigma_{j+3}}} \ldots$$

mit unendlich vielen $t_{(q_i,x_j,q_j)} \in T_I$ existiert. Somit wurde ein illegaler ω-Trace nachgewiesen.

($\Leftarrow$) Gegeben sei ein illegaler ω-Trace $M_0 \xrightarrow{\sigma}$ von $\Upsilon_{S\times\neg\varphi}$. Da in jedem illegalen ω-Trace unendlich viele Schaltinstanzen von Transitionen aus $T_I \subseteq T_B$ vorkommen, muß σ unendlich viele Instanzen von Transitionen der Art $t_{(q_i,x,q_j)}$ mit $q_j \in Q_E$ aufweisen. Daraus ergibt sich jedoch unmittelbar, daß jeder illegale ω-Trace von $\Upsilon_{S\times\neg\varphi}$ die folgende Form aufweist:

$$(s_{q_0}, s_f, O_0, H_0) \xrightarrow{t^{\varsigma'}_{(q_0,\nu(O_0),q_1)}} (s_{q_1}, s_s, O_0, H_0) \xrightarrow{t^{\varsigma_1}_1} (s_{q_1}, s_s, O_0, H_1) \xrightarrow{t^{\varsigma_2}_2} \ldots \xrightarrow{t^{\varsigma_i}_i}$$
$$(s_{q_1}, s_s, O_0, H_i) \xrightarrow{t^{\varsigma_{i+1}}_{i+1}} (s_{q_1}, s_f, O_1, H_{i+1}) \xrightarrow{t^{\varsigma''}_{(q_1,\nu(O_1),q_2)}} (s_{q_2}, s_s, O_1, H_{i+1}) \xrightarrow{t^{\varsigma_{i+2}}_{i+2}}$$
$$(s_{q_2}, s_s, O_1, H_{i+2}) \xrightarrow{t^{\varsigma_{i+3}}_{i+3}} \ldots \xrightarrow{t^{\varsigma_j}_j} (s_{q_2}, s_s, O_1, H_j) \xrightarrow{t^{\varsigma_{j+1}}_{j+1}} (s_{q_2}, s_f, O_2, H_{j+1})$$
$$\xrightarrow{t^{\varsigma'''}_{(q_2,\nu(O_2),q_3)}} (s_{q_3}, s_s, O_2, H_{j+1}) \xrightarrow{t^{\varsigma_{j+2}}_{j+2}} (s_{q_3}, s_s, O_2, H_{j+2}) \xrightarrow{t^{\varsigma_{j+3}}_{j+3}} \ldots$$

Aus der Tatsache, daß Υ_S ein echtes Teilnetz von $\Upsilon_{S\times\neg\varphi}$ derart beschreibt, daß die Mengen der sichtbaren und unsichtbaren Stellen und Transitionen in beiden Netzsystemen identisch sind, folgt unmittelbar die Existenz des Ablaufs

$$(O_0, H_0) \xrightarrow{t^{\varsigma_1}_1} (O_0, H_1) \xrightarrow{t^{\varsigma_2}_2} \ldots \xrightarrow{t^{\varsigma_i}_i} (O_0, H_i) \xrightarrow{t^{\varsigma_{i+1}}_{i+1}} (O_1, H_{i+1}) \xrightarrow{t^{\varsigma_{i+2}}_{i+2}} (O_1, H_{i+2})$$
$$\xrightarrow{t^{\varsigma_{i+3}}_{i+3}} \ldots \xrightarrow{t^{\varsigma_j}_j} (O_1, H_j) \xrightarrow{t^{\varsigma_{j+1}}_{j+1}} (O_2, H_{j+1}) \xrightarrow{t^{\varsigma_{j+2}}_{j+2}} (O_2, H_{j+2}) \xrightarrow{t^{\varsigma_{j+3}}_{j+3}} \ldots$$

in Υ_S. Da der illegale ω-Trace unendlich viele Büchitransitionsinstanzen aufweist und aufgrund der Schedulerstellen s_s und s_f zwischen dem Vorkommen zweier Büchitransitionsinstanzen immer mindestens eine sichtbare Transitionsinstanz schalten muß, weist der Ablauf auch unendlich viele Schaltinstanzen sichtbarer Transitionen auf, weshalb er Abläufen vom Typ I zuzuordnen ist.
Bleibt noch zu zeigen, daß der Ablauf die LTL-X-Eigenschaft φ verletzt. Aus Zeile 6.2.2 auf der vorherigen Seite folgt, daß es für jeden Übergang

$$(s_{q_i}, s_f, O_k, H_l) \xrightarrow{t^{\varsigma}_{(q_i,\nu(O_k),q_j)}} (s_{q_j}, s_s, O_k, H_l)$$

des illegalen ω-Traces einen entsprechenden Übergang

$$q_i \xrightarrow{\nu(O_k)} q_j$$

in $\mathcal{A}_{\neg\varphi}$ gibt. Da zudem nach Voraussetzung $q_j \in Q_E$ für unendlich viele q_j gilt, folgt unmittelbar

$$\nu(O_0)\nu(O_1)\nu(O_2)\ldots \in L(\mathcal{A}_{\neg\varphi})$$

Mit der Argumentation, daß LTL-X unter Stottern abgeschlossen ist und zudem $\nu(O_i) = \nu((O_i, H_j))$ für $i, j \geq 0$ gilt, folgt das gewünschte Ergebnis

$$\nu((O_0,H_0))\nu((O_0,H_1))\ldots\nu((O_0,H_i))\nu((O_1,H_{i+1}))\nu((O_1,H_{i+2}))\ldots$$
$$\ldots\nu((O_1,H_j))\nu((O_2,H_{j+1}))\nu((O_2,H_{j+2}))\ldots \in L(\mathcal{A}_{\neg\varphi})$$

- Es gibt in Υ_S genau dann eine die LTL-X-Eigenschaft φ verletzende Schaltfolge vom Typ II, wenn $\Upsilon_{S\times\neg\varphi}$ einen illegalen Livelock aufweist.

  ($\Rightarrow$) Gegeben sei in Υ_S eine die LTL-X-Eigenschaft φ verletzende Schaltfolge

  $$(O_0,H_0) \xrightarrow{t_1^{\varsigma 1}} (O_0,H_1) \xrightarrow{t_2^{\varsigma 2}} \ldots \xrightarrow{t_i^{\varsigma i}} (O_0,H_i) \xrightarrow{t_{i+1}^{\varsigma i+1}} (O_1,H_{i+1}) \xrightarrow{t_{i+2}^{\varsigma i+2}} (O_1,H_{i+2})$$
  $$\xrightarrow{t_{i+3}^{\varsigma i+3}} \ldots \xrightarrow{t_j^{\varsigma j}} (O_1,H_j) \xrightarrow{t_{j+1}^{\varsigma j+1}} (O_2,H_{j+1}) \xrightarrow{t_{j+2}^{\varsigma j+2}} (O_2,H_{j+2}) \xrightarrow{t_{j+3}^{\varsigma j+3}} \ldots$$

  vom Typ II, d.h., ohne Einschränkung der Allgemeinheit bezeichne t_{j+1} die letzte sichtbare Transition, deren Instanz in der Schaltfolge vorkommt. Da dieser Ablauf von Υ_S die LTL-X-Eigenschaft φ nicht erfüllt, gilt unmittelbar

  $$\nu((O_0,H_0))\nu((O_0,H_1))\ldots\nu((O_0,H_i))\nu((O_1,H_{i+1}))\nu((O_1,H_{i+2}))\ldots$$
  $$\ldots\nu((O_1,H_j))\nu((O_2,H_{j+1}))\nu((O_2,H_{j+2}))\ldots \in L(\mathcal{A}_{\neg\varphi})$$

  Da $\nu((O_i,H_j)) = \nu(O_i)$ für alle $i,j \geq 0$ und LTL-X zudem unter Stottern abgeschlossen ist, gilt auch

  $$\nu(O_0)\nu(O_1)(\nu(O_2))^\omega \in L(\mathcal{A}_{\neg\varphi})$$

  Dann gibt es jedoch in $\mathcal{A}_{\neg\varphi}$ einen Ablauf

  $$q_0 \xrightarrow{\nu(O_0)} q_1 \xrightarrow{\nu(O_1)} q_2 \xrightarrow{(\nu(O_2))^\omega} \ldots \tag{6.2.3}$$

  mit unendlich vielen $q_i \in Q_E$. Daraus folgt jedoch mit Zeile 6.2.2 auf Seite 194, daß in $\Upsilon_{S\times\neg\varphi}$ der Ablauf

  $$(s_{q_0},s_f,O_0,H_0) \xrightarrow{t_{(q_0,\nu(O_0),q_1)}^{\varsigma'}} (s_{q_1},s_s,O_0,H_0) \xrightarrow{t_1^{\varsigma 1}} (s_{q_1},s_s,O_0,H_1) \xrightarrow{t_2^{\varsigma 2}} \ldots \xrightarrow{t_i^{\varsigma i}}$$
  $$(s_{q_1},s_s,O_0,H_i) \xrightarrow{t_{i+1}^{\varsigma i+1}} (s_{q_1},s_f,O_1,H_{i+1}) \xrightarrow{t_{(q_1,\nu(O_1),q_2)}^{\varsigma''}} (s_{q_2},s_s,O_1,H_{i+1}) \xrightarrow{t_{i+2}^{\varsigma i+2}}$$
  $$(s_{q_2},s_s,O_1,H_{i+2}) \xrightarrow{t_{i+3}^{\varsigma i+3}} \ldots \xrightarrow{t_j^{\varsigma j}} (s_{q_2},s_s,O_1,H_j) \xrightarrow{t_{j+1}^{\varsigma j+1}} (s_{q_2},s_f,O_2,H_{j+1}) = M$$

  existiert. Aus $\nu(M) = \nu(O_2)$ und Zeile 6.2.3 folgt

  $$(\nu(M))^\omega \in L(\mathcal{A}_{\neg\varphi},q_2)$$

  Damit folgt aus der Konstruktion des Produktnetzsystems $\Upsilon_{S\times\neg\varphi}$ die Existenz der Transition $t_{(q_2,M)} \in T_L$, sodaß

  $$M \xrightarrow{t_{(q_2,M)}^{\varsigma}} H_{j+1} \xrightarrow{t_{j+2}^{\varsigma j+2}} H_{j+2} \xrightarrow{t_{j+3}^{\varsigma j+3}} \ldots$$

  Somit wurde ein illegaler Livelock in $\Upsilon_{S\times\neg\varphi}$ nachgewiesen.

($\Leftarrow$) Gegeben sei ein illegaler Livelock $M_0 \xrightarrow{\sigma t^{\varsigma}} M \xrightarrow{\sigma_1}$ von $\Upsilon_{S\times\neg\varphi}$, d.h., $t \in T_L$ und $\sigma_1 \in \{t_i^{\varsigma_i} \in \mathcal{I}_T \mid t_i \in T_H\}^\omega$. Somit weist jeder illegale Livelock von $\Upsilon_{S\times\neg\varphi}$ die folgende Form auf:

$$(s_{q_0}, s_f, O_0, H_0) \xrightarrow{t^{\varsigma'}_{(q_0,\nu(O_0),q_1)}} (s_{q_1}, s_s, O_0, H_0) \xrightarrow{t_1^{\varsigma_1}} (s_{q_1}, s_s, O_0, H_1) \xrightarrow{t_2^{\varsigma_2}} \ldots \xrightarrow{t_i^{\varsigma_i}}$$
$$(s_{q_1}, s_s, O_0, H_i) \xrightarrow{t_{i+1}^{\varsigma_{i+1}}} (s_{q_1}, s_f, O_1, H_{i+1}) \xrightarrow{t^{\varsigma''}_{(q_1,\nu(O_1),q_2)}} (s_{q_2}, s_s, O_1, H_{i+1}) \xrightarrow{t_{i+2}^{\varsigma_{i+2}}}$$
$$(s_{q_2}, s_s, O_1, H_{i+2}) \xrightarrow{t_{i+3}^{\varsigma_{i+3}}} \ldots \xrightarrow{t_j^{\varsigma_j}} (s_{q_2}, s_s, O_1, H_j) \xrightarrow{t_{j+1}^{\varsigma_{j+1}}} (s_{q_2}, s_f, O_2, H_{j+1}) = M'$$
$$\xrightarrow{t^{\varsigma}_{(q_2,M')}} H_{j+1} \xrightarrow{t_{j+2}^{\varsigma_{j+2}}} H_{j+2} \xrightarrow{t_{j+3}^{\varsigma_{j+3}}} \ldots$$

Dabei bezeichne t_{j+1} ohne Einschränkung der Allgemeinheit die letzte sichtbare Transition, deren Instanz in dem illegalen Livelock vorkommt. Desweiteren gilt $t_{(q_2,M')} \in T_L$, und die Anzahl an Vorkommen von Büchitransitionsinstanzen $t^{\varsigma_k}_{(q_i,x,q_j)}$ kann beliebig variieren, ist aber endlich. Aus der Tatsache, daß Υ_S ein echtes Teilnetz von $\Upsilon_{S\times\neg\varphi}$ derart beschreibt, daß die Mengen der sichtbaren und unsichtbaren Stellen und Transitionen in beiden Netzsystemen identisch sind, folgt unmittelbar die Existenz des Ablaufs

$$(O_0, H_0) \xrightarrow{t_1^{\varsigma_1}} (O_0, H_1) \xrightarrow{t_2^{\varsigma_2}} \ldots \xrightarrow{t_i^{\varsigma_i}} (O_0, H_i) \xrightarrow{t_{i+1}^{\varsigma_{i+1}}} (O_1, H_{i+1}) \xrightarrow{t_{i+2}^{\varsigma_{i+2}}} (O_1, H_{i+2})$$
$$\xrightarrow{t_{i+3}^{\varsigma_{i+3}}} \ldots \xrightarrow{t_j^{\varsigma_j}} (O_1, H_j) \xrightarrow{t_{j+1}^{\varsigma_{j+1}}} (O_2, H_{j+1}) \xrightarrow{t_{j+2}^{\varsigma_{j+2}}} (O_2, H_{j+2}) \xrightarrow{t_{j+3}^{\varsigma_{j+3}}} \ldots$$

in Υ_S. Da alle Schaltinstanzen des illegalen Livelocks, die nach der Transition $t_{(q_2,M')}$ vorkommen, unsichtbaren Transitionen zugeordnet werden, enthält obiger Ablauf beliebig, aber endlich viele Schaltinstanzen sichtbarer Transitionen, weshalb er Abläufen vom Typ II zuzuordnen ist.
Bleibt noch zu zeigen, daß der Ablauf die LTL-X-Eigenschaft φ verletzt. Aus Zeile 6.2.2 auf Seite 194 folgt, daß es für jeden Übergang

$$(s_{q_i}, s_f, O_k, H_l) \xrightarrow{t^{\varsigma}_{(q_i,\nu(O_k),q_j)}} (s_{q_j}, s_s, O_k, H_l)$$

des illegalen Livelocks einen entsprechenden Übergang

$$q_i \xrightarrow{\nu(O_k)} q_j$$

in $\mathcal{A}_{\neg\varphi}$ gibt, d.h.,

$$q_0 \xrightarrow{\nu(O_0)} q_1 \xrightarrow{\nu(O_1)} q_2$$

Wegen $t_{(q_2,M')} \in T_L$ gilt zudem

$$(\nu(M'))^\omega = (\nu(O_2))^\omega \in L(\mathcal{A}_{\neg\varphi}, q_2)$$

und somit

$$\nu(O_0)\nu(O_1)(\nu(O_2))^\omega \in L(\mathcal{A}_{\neg\varphi})$$

Mit der Argumentation, daß LTL-X unter Stottern abgeschlossen ist und zudem $\nu(O_i) = \nu((O_i, H_j))$ für $i, j \geq 0$ gilt, folgt das gewünschte Ergebnis

$$\nu((O_0,H_0))\nu((O_0,H_1))\ldots\nu((O_0,H_i))\nu((O_1,H_{i+1}))\nu((O_1,H_{i+2}))\ldots \\ \ldots\nu((O_1,H_j))\nu((O_2,H_{j+1}))\nu((O_2,H_{j+2}))\ldots \in L(\mathcal{A}_{\neg\varphi})$$

■

In Abschnitt 5.1.4 auf Seite 119 wurde bereits ein Beispiel für einen illegalen Livelock betrachtet. In diesem Zusammenhang spielen die Livelock-Wächter (Transitionen aus T_L) eine zentrale Rolle. Da deren Anzahl allerdings exponentiell in der Größe des M-Netzsystems Υ_S zunehmen kann, werden sie nicht explizit in das Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ eingefügt, sondern, wie es auch in [EH01, Hel02] vorgeschlagen wurde, „On-the-Fly“ während der Präfixgenerierung erzeugt. Aus diesem Grund wird $\Upsilon_{S\times\neg\varphi}$ um die Menge T'_L von Livelock-Wächterkandidaten erweitert:

- Füge für jede Büchitransition $t_{(q,x,q')} \in T_B$ eine Kopie $L_{(q,x,q')} \in T'_L$ hinzu, sodaß

$$\begin{aligned} {}^\bullet L_{(q,x,q')} &= {}^\bullet t_{(q,x,q')} \\ L^\bullet_{(q,x,q')} &= \emptyset \\ \iota(L_{(q,x,q')}) &= \iota(t_{(q,x,q')}) \\ \forall s \in {}^\bullet L_{(q,x,q')} \colon \iota((s, L_{(q,x,q')})) &= \iota((s, t_{(q,x,q')})) \end{aligned}$$

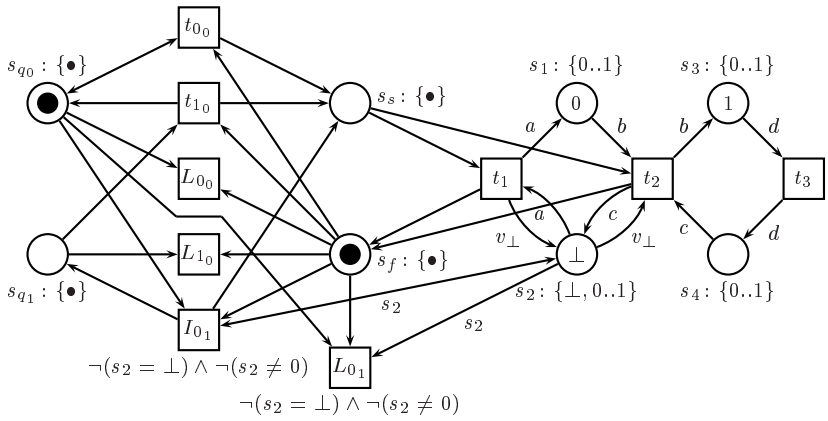
Abbildung 6.7 auf der nächsten Seite zeigt das um die Menge T'_L von Livelock-Wächterkandidaten erweiterte Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$. Die Livelock-Wächterkandidaten werden innerhalb der Entfaltungsroutine zur Ermittlung der tatsächlichen Livelock-Wächter verwendet. Dazu wird immer, wenn ein mit einer Transition aus T'_L beschriftetes Ereignis e in das Präfix eingefügt werden kann, überprüft, ob

$$(\nu(M))^\omega \in L(\mathcal{A}_{\neg\varphi}, q), \tag{6.2.4}$$

wobei $M \in \mathcal{R}(M_0)$ die von dem Schnitt induzierte Markierung bezeichnet, unter dem e in das Präfix eingefügt werden kann, und $q \in Q$ den Zustand beschreibt, für den $\bullet \in M(s_q)$ gilt. Im Falle der erfolgreichen Überprüfung wird das Ereignis e als Livelock-Wächter in das Präfix eingefügt (andernfalls wird es verworfen), wobei dessen Vorbereich um alle Bedingungen des Schnitts erweitert wird und dessen Nachbereich dem Vorbereich ohne diejenigen Bedingungen entspricht, welche Büchi-, Scheduler- und sichtbare Stellen repräsentieren. Diese Vorgehensweise garantiert, daß nach dem Schalten eines Livelock-Wächters ausschließlich nur noch Instanzen unsichtbarer Transitionen ausgeführt werden können.
Die Problemstellung in Zeile 6.2.4 kann effizient gelöst werden, indem zunächst alle Übergänge der Art

$$(q_i, x, q_j) \in \delta \quad \text{mit} \quad x \neq \nu(M)$$

Abbildung 6.7 *Das um die Menge T'_L von Livelock-Wächterkandidaten erweiterte Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$, wobei die Darstellung aus Gründen der Übersichtlichkeit vereinfacht wurde und nur die zum Verständnis des Beispiels notwendigen Netzbestandteile zu sehen sind. Desweiteren wurden auch die Transitionswächter **wahr** und die Kantenbeschriftungen $\{v_\bullet\}$ weggelassen.*



aus $\mathcal{A}_{\neg\varphi}$ entfernt werden und anschließend nach einer starken Zusammenhangskomponente[3] von $\mathcal{A}_{\neg\varphi}$ gesucht wird, die von q aus erreichbar ist und mindestens einen Endzustand $q_e \in Q_E$ aufweist. Verschiedene Algorithmen zur Berechnung starker Zusammenhangskomponenten sind beispielsweise in [Tar72, Gab00, Sha81] beschrieben.

6.3 Tableau-System

Satz 6.2.1 auf Seite 193 besagt, daß der Nachweis von LTL-X-Eigenschaften für M-Netzsysteme auf die Abwesenheit illegaler ω-Traces und illegaler Livelocks innerhalb des Produktnetzsystems $\Upsilon_{S\times\neg\varphi}$ zurückgeführt werden kann. In [EH01, Hel02] wurde diese Problemstellung mit Hilfe eines Branching-Prozesses gelöst, der als „nebenläufiges Tableau-System“ aufgefaßt wird, in dem Bedingungen als „Fakten“ und Ereignisse als „Inferenzen“ verstanden werden. Das Tableau-System wird durch schrittweises Hinzufügen von Ereignissen gemäß einer adäquaten Ordnung $\prec_{LTL-X}$ in der Menge der endlichen Konfigurationen der Entfaltung von $\Upsilon_{S\times\neg\varphi}$ erstellt.

Definition 6.3.1 (Adäquate Ordnung für LTL-X Model-Checking [EH01])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und dessen Entfaltung β. Für zwei endliche Konfigurationen C_1 und C_2 von β gilt $C_1 \prec_{LTL-X} C_2$, falls

- $BL(C_1) \prec BL(C_2)$ oder

[3] Eine starke Zusammenhangskomponente eines Automaten $\mathcal{A}$ beschreibt einen Teilautomaten $\mathcal{A}'$ von $\mathcal{A}$, in dem jeder Zustand von jedem anderen Zustand aus über einen Pfad erreichbar ist.

- $BL(C_1) = BL(C_2)$ und $C_1 \prec C_2$,

wobei $\prec$ eine adäquate Ordnung (siehe [ERV96, ERV02]) beschreibt und

$$BL(C_i) = \{e \in C_i \mid \forall e' \in [e] \setminus \{e\} \colon \lambda(e') \notin \{t^S \in \mathcal{I}_T \mid t \in T_L\}\}$$

für eine Konfiguration C_i die Menge von Ereignissen ergibt, die kausal vor einem Ereignis vorkommen, das einen Livelock-Wächter repräsentiert.

Bevor das Tableau-System für LTL-X Model-Checking von M-Netzsystemen definiert werden kann, werden die Ereignisse der Entfaltung des Produktnetzsystems $\Upsilon_{S\times\neg\varphi}$ zunächst noch in die disjunkten Mengen E_I und E_{II} unterteilt, die für die Erkennung illegaler ω-Traces (E_I-Ereignisse) bzw. illegaler Livelocks (E_{II}-Ereignisse) verwendet werden.

Definition 6.3.2 (Mengen E_I und E_{II} von Ereignissen [EH01, Hel02])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und die Entfaltung $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ von $\Upsilon_{S\times\neg\varphi}$. Für ein Ereignis $e \in E$ gilt:

$$e \in \begin{cases} E_I & \text{falls } \forall e' \in [e] \colon \lambda(e') \notin \{t^S \in \mathcal{I}_T \mid t \in T_L\} \\ E_{II} & \text{sonst} \end{cases}$$

In der nun folgenden Definition des Tableau-Systems wird die Notation $C_{\#T_I}$ verwendet, welche für eine endliche Konfiguration C die Anzahl von Ereignissen angibt, die eine Schaltinstanz für eine Transition aus T_I repräsentieren.

Definition 6.3.3 (Tableau-System für LTL-X Model-Checking [SK04])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und dessen Entfaltung β. Ein Ereignis e von β heißt *Terminal*, falls es eine (nicht notwendigerweise lokale) Konfiguration C von β gibt, sodaß $Mark(C) = Mark([e]), C \prec_{LTL-X} [e]$ und eine der folgenden, sich gegenseitig ausschließenden Bedingungen gelten:

(i) $e \in E_I$ und entweder

  (a) $C \subset [e]$ oder

  (b) $C \not\subset [e]$, C stellt eine lokale Konfiguration dar und $C_{\#T_I} \geq [e]_{\#T_I}$

(ii) $e \in E_{II}$ und entweder

  (a) C stellt eine lokale Konfiguration dar und $BL(C) \prec_{LTL-X} BL([e])$ oder

  (b) $C \subset [e]$ oder

  (c) $C \not\subset [e]$, C stellt eine lokale Konfiguration dar, $BL(C) = BL([e])$ und $|C| \geq |[e]|$

Ein *Tableau-System* $\mathcal{T}$ wird durch einen Branching-Prozeß (B, E) von $\Upsilon_{S\times\neg\varphi}$ derart beschrieben, sodaß sich für jede mögliche Erweiterung e von (B, E) mindestens ein Terminal unter den direkten kausalen Vorgängern von e befindet. Ein Terminal heißt *erfolgreich*, falls es vom Typ (i)(a) ist und die Menge $[e] \setminus C$ mindestens einen Repräsentanten für eine Schaltinstanz einer Transition aus T_I enthält, oder es vom Typ (ii)(b) ist. Ein Tableau-System $\mathcal{T}$ heißt *erfolgreich*, falls es mindestens ein erfolgreiches Terminal enthält.

Im folgenden soll nun eine Vorstellung darüber vermittelt werden, wie illegale ω-Traces und illegale Livelocks mittels erfolgreicher Terminale eines Tableau-Systems entdeckt werden können. Abbildung 6.8 auf der nächsten Seite zeigt das Tableau-System $\mathcal{T}$ für das Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ aus Abbildung 6.7 auf Seite 199. Das Tableau-System $\mathcal{T}$ enthält die drei (grau unterlegten) Terminale e_{11}, e_{16} und e_{17}, wobei das Augenmerk zunächst auf e_{17} gerichtet werden soll. Da e_{17} ein Terminal bezeichnet, muß es nach Definition 6.3.3 auf der vorherigen Seite eine Konfiguration C geben, die gemäß der adäquaten Ordnung $\prec_{LTL-X}$ (Definition 6.3.1 auf Seite 199) kleiner als die lokale Konfiguration $[e_{17}]$ ist und deren Linearisierung dieselbe Markierung beschreibt wie die Linearisierung von $[e_{17}]$. Diese Bedingungen sind für

$$\begin{aligned} C &= \{e_2\} \\ [e_{17}] &= \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_{13}, e_{15}, e_{17}\} \end{aligned}$$

erfüllt, da

$$Mark(C) = \{s^\bullet_{q_0}, s^\bullet_s, s^0_1, s^\perp_2, s^1_3\} = Mark([e_{17}]) \tag{6.3.1}$$

und $BL(C) \prec BL([e_{17}])$. (Tatsächlich induziert die Ausführung aller Ereignisse von C und $[e_{17}]$ ausgehend von der Anfangsmarkierung in $\mathcal{T}$ die Schnitte

$$\begin{aligned} Cut(C) &= \{b_1, b_2, b_3, b_7, b_8\} \\ Cut([e_{17}]) &= \{b_{21}, b_{32}, b_{33}, b_{38}, b_{39}\}, \end{aligned}$$

die jedoch dieselbe Markierung (Zeile 6.3.1) des Produktnetzsystems $\Upsilon_{S\times\neg\varphi}$ repräsentieren.) Da die Konfiguration C vollständig in der Konfiguration $[e_{17}]$ enthalten ist, d.h., $C \subset [e_{17}]$, gibt es in dem Tableau-System $\mathcal{T}$ eine Linearisierung von $[e_{17}] \setminus C$, die von $Cut(C)$ nach $Cut([e_{17}])$ führt, d.h.,

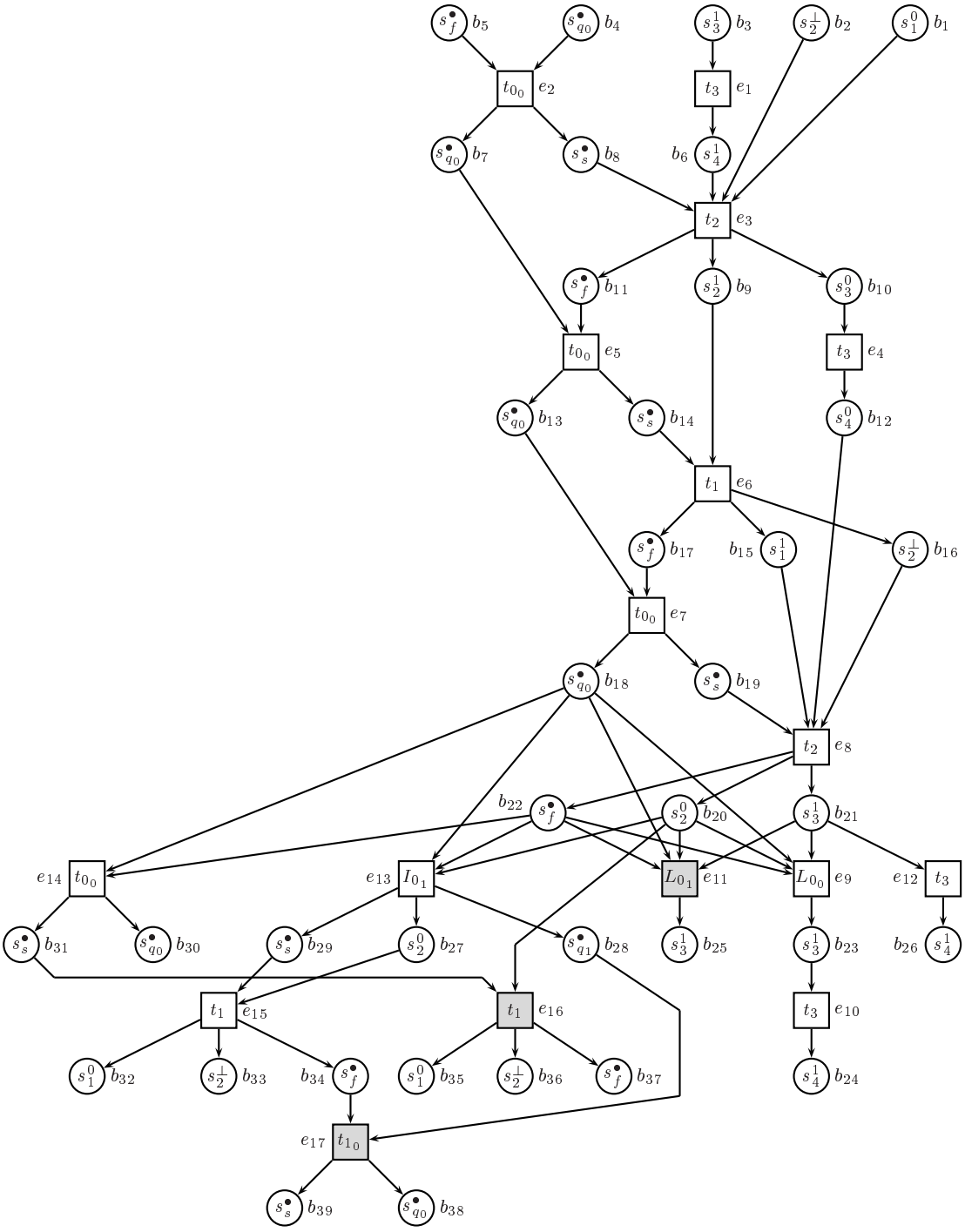
$$Cut(C) = \{b_1, b_2, b_3, b_7, b_8\} \xrightarrow{e_1 e_3 e_4 e_5 e_6 e_7 e_8 e_{13} e_{15} e_{17}} \{b_{21}, b_{32}, b_{33}, b_{38}, b_{39}\} = Cut([e_{17}])$$

Daraus folgt mit Zeile 6.3.1 jedoch unmittelbar, daß in dem Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ (Abbildung 6.7 auf Seite 199) der unendliche Ablauf

$$M_0 \xrightarrow{t^\bullet_{0_0}} \{s^\bullet_{q_0}, s^\bullet_s, s^0_1, s^\perp_2, s^1_3\} \xrightarrow{(t^1_3 t^{\bullet 01\perp}_2 t^0_3 t^\bullet_{0_0} t^{\bullet 1\perp}_1 t^\bullet_{0_0} t^{\bullet 10\perp}_2 I^{\bullet 0}_{0_1} t^{\bullet 0\perp}_1 t^\bullet_{1_0})^\omega}$$

existiert, in dem unendlich oft die Schaltinstanz $I^{\bullet 0}_{0_1}$ mit $I_{0_1} \in T_I$ vorkommt. Dieses entspricht jedoch genau Definition 6.2.2 auf Seite 193 eines illegalen ω-Traces. Somit

Abbildung 6.8 *Tableau-System $\mathcal{T}$ für das Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ aus Abbildung 6.7 auf Seite 199, wobei die Terminale grau unterlegt sind*



wurde beispielhaft verdeutlicht, daß illegale ω-Traces mittels erfolgreicher Terminale vom Typ (i)(a) entdeckt werden können.
Illegale Livelocks können auf ähnliche Art und Weise anhand von erfolgreichen Terminalen des Typs (ii)(b) erkannt werden. In diesem Fall enthält die lokale Konfiguration eines erfolgreichen Terminals e ein Ereignis, welches die Schaltinstanz eines Livelock-Wächters (einer Transition aus T_L) repräsentiert. Da e ein Terminal ist, gibt es eine bezüglich der adäquaten Ordnung $\prec_{LTL-X}$ kleinere Konfiguration C, sodaß $Mark(C) = Mark([e])$. Falls e selbst keine Schaltinstanz eines Livelock-Wächters repräsentiert, liegen die beiden Schnitte $Cut(C)$ und $Cut([e])$ jenseits eines Ereignisses, das eine Schaltinstanz eines Livelock-Wächters repräsentiert. An dieser Stelle kann nun die gleiche Argumentation wie bei den illegalen ω-Traces verwendet werden. Da C vollständig in $[e]$ enthalten ist, gibt es einen Ablauf von $Cut(C)$ nach $Cut([e])$, der ausschließlich Repräsentanten unsichtbarer Transitionsinstanzen enthält und unendlich oft wiederholt werden kann.

Satz 6.3.1 (Tableau-System für LTL-X Model-Checking [SK04])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ und dessen Tableau-System $\mathcal{T}$.

- *$\Upsilon_{S\times\neg\varphi}$ weist genau dann einen illegalen ω-Trace auf, wenn es in $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (i)(a) gibt.*
- *$\Upsilon_{S\times\neg\varphi}$ weist genau dann einen illegalen Livelock auf, wenn es in $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (ii)(b) gibt.*

6.3.1 Beweis von Satz 6.3.1

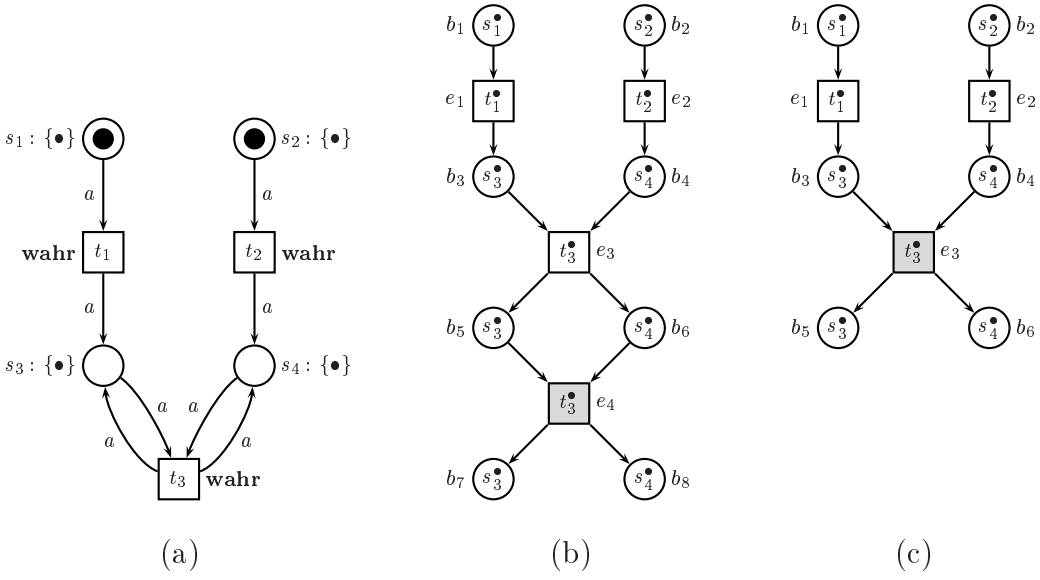
Das Tableau-System $\mathcal{T}$ aus Definition 6.3.3 auf Seite 200 für die Verifikation von LTL-X-Eigenschaften streng sicherer M-Netzsysteme unterscheidet sich von dem in [EH01, Hel02] vorgeschlagenen Tableau-System $\mathcal{T}'$ für das LTL-X Model-Checking sicherer S/T-Netzsysteme darin, daß Terminale in $\mathcal{T}'$ ausschließlich aufgrund lokaler Konfigurationen ermittelt werden, wohingegen Terminale in $\mathcal{T}$ auch aufgrund nicht notwendigerweise lokaler Konfigurationen bestimmt werden. Diese Vorgehensweis kann sich auf die Größe des Tableau-Systems auswirken, wie anhand des streng sicheren M-Netzsystems aus Abbildung 6.9 (a) auf der nächsten Seite gezeigt wird. Werden Terminale lediglich aufgrund lokaler Konfigurationen ermittelt, so ergibt sich das Tableau-System aus Abbildung 6.9 (b). Das Ereignis e_3 wird nicht als Terminal erkannt, da

$$Mark([e_3]) = \{s_3^\bullet, s_4^\bullet\} \neq \{s_2^\bullet, s_3^\bullet\} = Mark([e_1])$$
$$Mark([e_3]) = \{s_3^\bullet, s_4^\bullet\} \neq \{s_1^\bullet, s_4^\bullet\} = Mark([e_2])$$

Läßt man zur Bestimmung von Terminalen jedoch auch globale Konfigurationen zu, dann erhält man das Tableau-System aus Abbildung 6.9 (c). In diesem Fall wird das Ereignis e_3 aufgrund der Konfiguration $C = \{e_1, e_2\}$ als Terminal erkannt, da

$$Mark([e_3]) = \{s_3^\bullet, s_4^\bullet\} = Mark(C)$$

Abbildung 6.9 *Streng sicheres M-Netzsystem (a) und dessen Tableau-Systeme, bei denen die Terminale (grau unterlegt) aufgrund lokaler (b) bzw. globaler (c) Konfigurationen ermittelt werden*



und $C \prec [e_3]$. In dem Tableau-System $\mathcal{T}$ aus Definition 6.3.3 auf Seite 200 werden Terminale also unter Umständen früher erkannt als in dem Tableau-System $\mathcal{T}'$ aus [EH01, Hel02], weshalb $\mathcal{T}$ höchstens die gleiche Größe wie $\mathcal{T}'$ aufweist, aber in einigen Fällen auch kleiner sein kann.
Die Beweisführung von Satz 6.3.1 auf der vorherigen Seite orientiert sich an der Vorgehensweise in [EH01, Hel02] und ist in den Nachweis der Sätze 6.3.2 bis 6.3.5 auf Seite 211 unterteilt.

Satz 6.3.2 (Korrektheit von $\mathcal{T}$ bezüglich illegaler ω-Traces)
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und dessen Tableau-System $\mathcal{T}$. Wenn es in $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (i)(a) gibt, dann weist $\Upsilon_{S\times\neg\varphi}$ einen illegalen ω-Trace auf.

Beweis: Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$, dessen Tableau-System $\mathcal{T} = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ und ein erfolgreiches Terminal e vom Typ (i)(a). Aus Definition 6.3.3 auf Seite 200 folgt die Existens einer Konfiguration C von $\mathcal{T}$, sodaß

$$Mark(C) = Mark([e]) \tag{6.3.2}$$

$$C \subset [e] \tag{6.3.3}$$

$$\exists e' \in [e] \setminus C : \lambda(e') \in \{t^{\varsigma} \in \mathcal{I}_T \mid t \in T_I\} \tag{6.3.4}$$

Mit Zeile 6.3.3 auf der vorherigen Seite folgt, daß es in $\mathcal{T}$ einen Ablauf

$$Min(N') \xrightarrow{\sigma_1} Cut(C) \xrightarrow{\sigma_2} Cut([e])$$

gibt, wobei σ_1 und $\sigma_1\sigma_2$ Linearisierungen von C bzw. $[e]$ beschreiben. Daraus folgt jedoch unmittelbar die Existenz des Ablaufs

$$M_0 \xrightarrow{\lambda(\sigma_1)} Mark(C) \xrightarrow{\lambda(\sigma_2)} Mark([e])$$

in $\Upsilon_{S\times\neg\varphi}$. Aus Gleichung 6.3.2 auf der vorherigen Seite folgt unmittelbar die Existenz des unendlichen Ablaufs

$$M_0 \xrightarrow{\lambda(\sigma_1)} Mark(C) \xrightarrow{(\lambda(\sigma_2))^\omega},$$

in dem wegen Zeile 6.3.4 auf der vorherigen Seite unendlich viele Schaltinstanzen für Transitionen aus T_I vorkommen. ■

Für den Vollständigkeitsbeweis des Tableau-Systems $\mathcal{T}$ bezüglich illegaler ω-Traces werden zunächst noch die Definition 6.3.4 und die Proposition 6.3.1 benötigt.

Definition 6.3.4 (Schlechte Konfiguration einer Entfaltung [EH01, Hel02])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ und dessen Entfaltung β. Eine Konfiguration C von β heißt *schlecht*, falls sie mindestens $|\mathcal{R}(M_0)|+1$ Ereignisse enthält, die Schaltinstanzen für Transitionen aus T_I repräsentieren.

Proposition 6.3.1 (Eigenschaften schlechter Konfigurationen [EH01, Hel02])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und dessen Entfaltung β.

(i) $\Upsilon_{S\times\neg\varphi}$ weist genau dann einen illegalen ω-Trace auf, wenn es in β eine schlechte Konfiguration gibt.

(ii) Eine schlechte Konfiguration von β enthält mindestens ein erfolgreiches Terminal vom Typ (i)(a).

Beweis: Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$ und dessen Entfaltung $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$.

(i) ($\Rightarrow$) Wenn $\Upsilon_{S\times\neg\varphi}$ einen illegalen ω-Trace aufweist, dann gibt es einen Präfix $M_0 \xrightarrow{\sigma} M$ derart, daß $|\mathcal{R}(M_0)| + 1$ Schaltinstanzen für Transitionen aus T_I in σ vorkommen. Folglich gibt es eine Konfiguration C von β derart, sodaß σ' eine Linearisierung von C beschreibt, für die $\lambda(\sigma') = \sigma$ gilt. Nach Definition 6.3.4 beschreibt C eine schlechte Konfiguration von β.

($\Leftarrow$) Gegeben sei ein schlechte Konfiguration C von β. Nach Definition 6.3.4 enthält C mindestens $|\mathcal{R}(M_0)|+1$ Ereignisse, die Schaltinstanzen für Transitionen aus T_I repräsentieren. Da alle Transitionen aus T_I unter anderem die Eingangsstelle s_f aufweisen, können sie niemals nebenläufig ausgeführt werden. Demzufolge

müssen die sie repräsentierenden Ereignisse kausal angeordnet sein. Daher muß es nach dem Schubfachprinzip unter den $|\mathcal{R}(M_0)| + 1$ Ereignissen mindestens zwei Ereignisse $e' < e$ geben, welche die Eigenschaft

$$Mark([e']) = Mark([e]) \tag{6.3.5}$$

aufweisen. Daraus folgt unmittelbar die Existenz des Ablaufs

$$Min(N') \xrightarrow{\sigma_1} Cut([e']) \xrightarrow{\sigma_2} Cut([e])$$

in β, wobei σ_1 und $\sigma_1\sigma_2$ Linearisierungen von $[e']$ bzw. $[e]$ beschreiben. Daraus folgt jedoch unmittelbar die Existenz des Ablaufs

$$M_0 \xrightarrow{\lambda(\sigma_1)} Mark([e']) \xrightarrow{\lambda(\sigma_2)} Mark([e])$$

in $\Upsilon_{S\times\neg\varphi}$. Aus Gleichung 6.3.5 folgt unmittelbar die Existenz des unendlichen Ablaufs

$$M_0 \xrightarrow{\lambda(\sigma_1)} Mark([e']) \xrightarrow{(\lambda(\sigma_2))^\omega},$$

in dem unendlich viele Schaltinstanzen für Transitionen aus T_I vorkommen. Dieser Ablauf stellt einen illegalen ω-Trace von $\Upsilon_{S\times\neg\varphi}$ dar.

(ii) Das Ereignis e in Beweisrichtung (i)($\Leftarrow$) entspricht einem erfolgreichen Terminal vom Typ (i)(a). ($[e]$ kann kein Ereignis enthalten, das eine Schaltinstanz für eine Transition aus T_L repräsentiert, da nach der Ausführung einer solchen Schaltinstanz keine Marke mehr auf der Scheduler-Stelle s_f liegen würde (und auch im weiteren Verlauf nicht mehr produziert werden könnte), diese aber spätestens für das Ausführen der das Ereignis e repräsentierenden Schaltinstanz benötigt werden würde. Somit kann $[e]$ keine Terminale vom Typ (ii) enthalten.)

■

Satz 6.3.3 (Vollständigkeit von $\mathcal{T}$ bezüglich illegaler ω-Traces)
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und dessen Tableau-System $\mathcal{T}$. Wenn $\Upsilon_{S\times\neg\varphi}$ einen illegalen ω-Trace aufweist, dann gibt es in $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (i)(a).

Beweis: Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$, dessen Entfaltung β und dessen Tableau-System $\mathcal{T}$. Nach Proposition 6.3.1 (i) auf der vorherigen Seite genügt es zu zeigen, daß $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (i)(a) enthält, falls es in β eine schlechte Konfiguration gibt.
Dabei wird die folgende Beweisstrategie verfolgt: Für eine schlechte Konfiguration C von β wird gezeigt, daß entweder C ein erfolgreiches Terminal vom Typ (i)(a) enthält, das auch in $\mathcal{T}$ vorkommt, oder es eine schlechte Konfiguration C' von β mit $C' \prec_{LTL-X} C$ gibt. Da $\prec_{LTL-X}$ nach Definition 6.3.1 auf Seite 199 eine adäquate Ordnung darstellt und somit nach Definition 3.1.8 auf Seite 44 fundiert ist, enthält $\mathcal{T}$ ein erfolgreiches

Terminal vom Typ (i)(a).
Nach Proposition 6.3.1 auf Seite 205 enthält die Konfiguration C von β ein erfolgreiches Terminal e vom Typ (i)(a). Dabei kann e ohne Einschränkung der Allgemeinheit derart gewählt werden, daß es das bezüglich der Kausalrelation $<$ minimale erfolgreiche Terminal vom Typ (i)(a) in C darstellt. Falls e auch in $\mathcal{T}$ vorkommt, ist der Beweis beendet. Andernfalls muß es ein Terminal $e_1 < e$ in C geben, das aufgrund der Minimalität von e nicht erfolgreich sein kann. Aus $e \in E_I$ (siehe Definition 6.3.2 auf Seite 200) folgt unmittelbar $e_1 \in E_I$, weshalb e_1 ein nicht erfolgreiches Terminal vom Typ (i) sein muß. Es gilt

$$C = [e_1] \oplus (C \setminus [e_1]).$$

Nun bezeichne C_1 die Konfiguration von β, deretwegen e_1 als Terminal erkannt wurde. Desweiteren sei der Isomorphismus f zwischen $\uparrow C_1$ und $\uparrow[e_1]$ gegeben. Als nächstes wird die Konfiguration

$$C' = C_1 \oplus f(C \setminus [e_1])$$

definiert.
Es bleibt zu zeigen, daß es sich bei C' um eine schlechte Konfiguration von β handelt, die der Eigenschaft $C' \prec_{LTL-X} C$ genügt.
Aus $C_1 \prec_{LTL-X} [e_1]$ folgt mit der dritten Bedingung von Definition 3.1.8 auf Seite 44 einer adäquaten Ordnung unmittelbar $C' \prec_{LTL-X} C$. Es bleibt noch zu zeigen, daß C' eine schlechte Konfiguration von β bezeichnet. Dazu müssen die beiden Fälle unterschieden werden, deretwegen das Terminal e_1 als nicht erfolgreich eingestuft wurde.

(a) $C_1 \subset [e_1]$ und $[e_1] \setminus C_1$ enthält keinen Repräsentanten für eine Schaltinstanz einer Transition aus T_I.
Es genügt zu zeigen, daß C und C' dieselbe Anzahl von Ereignissen aufweisen, die Schaltinstanzen für Transitionen aus T_I repräsentieren. Da $[e_1] \setminus C_1$ keinen Repräsentanten für eine Schaltinstanz einer Transition aus T_I enthält, gilt $(C_1)_{\#T_I} = [e_1]_{\#T_I}$. Da zudem der Isomorphismus f auch die Ereignisbeschriftungen bewahrt, gilt

$$(f(C \setminus [e_1]))_{\#T_I} = (C \setminus [e_1])_{\#T_I}.$$

Daraus folgt dann $C'_{\#T_I} = C_{\#T_I}$.

(b) $C_1 \not\subset [e_1]$, C_1 stellt eine lokale Konfiguration dar und $(C_1)_{\#T_I} \geq [e_1]_{\#T_I}$.
Es genügt zu zeigen, daß C' mindestens soviele Ereignisse wie C aufweist, die Schaltinstanzen für Transitionen aus T_I repräsentieren. Nach Voraussetzung gilt $(C_1)_{\#T_I} \geq [e_1]_{\#T_I}$. Da zudem der Isomorphismus f auch die Ereignisbeschriftungen bewahrt, gilt

$$(f(C \setminus [e_1]))_{\#T_I} = (C \setminus [e_1])_{\#T_I}.$$

Daraus folgt dann $C'_{\#T_I} \geq C_{\#T_I}$.

■

Für den Nachweis der Korrektheit des Tableau-Systems $\mathcal{T}$ bezüglich illegaler Livelocks wird zunächst noch die Propositionen 6.3.2 auf der nächsten Seite benötigt.

Proposition 6.3.2 (Eigenschaften von Konfigurationen [EH01, Hel02])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$, dessen Entfaltung $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ sowie eine Konfiguration C von β.

(i) $BL(C)$ enthält höchstens einen Repräsentanten für eine Schaltinstanz einer Transition aus T_L. Falls $BL(C)$ einen solchen Repräsentanten enthält, dann stellt dieser das eindeutige, bezüglich der Kausalrelation $<$ maximale Ereignis von $BL(C)$ dar.

(ii) Alle Ereignisse der Menge $C \setminus BL(C)$ repräsentieren Schaltinstanzen unsichtbarer Transitionen.

(iii) Falls eine Linearisierung von C einen illegalen Livelock

$$M_0 \xrightarrow{\lambda(\sigma e)} M \xrightarrow{\lambda(\sigma_1)}$$

mit $\lambda(e) \in \{t^s \in \mathcal{I}_T \mid t \in T_L\}$ und $\lambda(\sigma_1) \in \{t^s \in \mathcal{I}_T \mid t \in T_H\}^\omega$ beschreibt, dann stellt σe eine Linearisierung von $BL(C)$ dar.

Beweis:

(i) Aus Definition 6.3.1 auf Seite 199 geht hervor, daß Repräsentanten von Schaltinstanzen für Transitionen aus T_L innerhalb der Menge $BL(C)$ von Ereignissen maximal bezüglich der Kausalrelation $<$ sind. Da eine Schaltinstanz einer Transition aus T_L aufgrund der Konstruktionsweise von $\Upsilon_{S\times\neg\varphi}$ (Definition 6.2.1 auf Seite 190) zu keiner anderen Schaltinstanz von $\Upsilon_{S\times\neg\varphi}$ nebenläufig ausgeführt werden kann, ist die Menge der bezüglich der Kausalrelation $<$ maximalen Ereignisse von $BL(C)$ entweder leer, oder sie besteht aus genau einem Repräsentanten einer Schaltinstanz für eine Transition aus T_L.

(ii) Falls $(C \setminus BL(C)) \neq \emptyset$, so folgt aus Definition 6.3.1 auf Seite 199 und Proposition 6.3.2 (i), daß das eindeutige, bezüglich der Kausalrelation $<$ maximale Ereignis von $BL(C)$ eine Schaltinstanz für eine Transition aus T_L repräsentiert. Nach Ausführen dieser Schaltinstanz in $\Upsilon_{S\times\neg\varphi}$ liegt auf keiner der Schedulerstellen s_s und s_f mehr eine Marke. Da diese aber im Vorbereich von Büchitransitionen, sichtbaren Transitionen und Livelock-Wächtern liegen, können folglich nur noch Schaltinstanzen unsichtbarer Transitionen ausgeführt werden.

(iii) Die Konfiguration C enthält mit e ein Ereignis, das eine Schaltinstanz für eine Transition aus T_L repräsentiert. Nach Proposition 6.3.2 (i) bezeichnet e das eindeutige, bezüglich der Kausalrelation $<$ maximale Ereignis von $BL(C)$.

■

Satz 6.3.4 (Korrektheit von $\mathcal{T}$ bezüglich illegaler Livelocks)
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und dessen Tableau-System $\mathcal{T}$. Wenn es in $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (ii)(b) gibt, dann weist $\Upsilon_{S\times\neg\varphi}$ einen illegalen Livelock auf.

BEWEIS: Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$, dessen Tableau-System $\mathcal{T} = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ und ein erfolgreiches Terminal e vom Typ (ii)(b) von $\mathcal{T}$. Aus Definition 6.3.3 auf Seite 200 folgt die Existenz einer Konfiguration C von $\mathcal{T}$, sodaß

$$Mark(C) = Mark([e]) \tag{6.3.6}$$
$$C \subset [e] \tag{6.3.7}$$

Aus $e \in E_{II}$ (siehe Definition 6.3.2 auf Seite 200) folgt mit Gleichung 6.3.6 die Existenz eines Ereignisses $e' \in C \cap E_{II}$. Da jede Konfiguration nach Proposition 6.3.2 (i) auf der vorherigen Seite höchstens einen Repräsentanten für eine Schaltinstanz einer Transition aus T_L enthält, folgt mit Zeile 6.3.7 unmittelbar $BL(C) = BL([e])$. Daraus folgt wiederum mit Zeile 6.3.7, daß es in dem Tableau-System $\mathcal{T}$ einen Ablauf

$$Min(N') \xrightarrow{\sigma_1} Cut(BL(C)) \xrightarrow{\sigma_2} Cut(C) \xrightarrow{\sigma_3} Cut([e])$$

gibt, wobei σ_1, $\sigma_1\sigma_2$ und $\sigma_1\sigma_2\sigma_3$ Linearisierungen von $BL(C)$, C bzw. $[e]$ bezeichnen und zumindest σ_1 und σ_3 nicht leer sind. Dann gibt es aber auch den Ablauf

$$M_0 \xrightarrow{\lambda(\sigma_1)} Mark(BL(C)) \xrightarrow{\lambda(\sigma_2)} Mark(C) \xrightarrow{\lambda(\sigma_3)} Mark([e])$$

in $\Upsilon_{S\times\neg\varphi}$. Nach Proposition 6.3.2 (i) auf der vorherigen Seite repräsentiert das letzte Ereignis von σ_1 eine Schaltinstanz für eine Transition aus T_L, d.h., es gibt eine Zerlegung $\sigma_1 = \sigma e$ mit $\lambda(e) \in \{t^\varsigma \in \mathcal{I}_T \mid t \in T_L\}$. Daraus folgt unmittelbar

$$M_0 \xrightarrow{\lambda(\sigma)t^\varsigma} Mark(BL(C)) \xrightarrow{\lambda(\sigma_2)} Mark(C) \xrightarrow{\lambda(\sigma_3)} Mark([e])$$

mit $t \in T_L$. Aus Gleichung 6.3.6 folgt die Existenz des unendlichen Ablaufs

$$M_0 \xrightarrow{\lambda(\sigma)t^\varsigma} Mark(BL(C)) \xrightarrow{\lambda(\sigma_2)(\lambda(\sigma_3))^\omega} .$$

Mit Proposition 6.3.2 (ii) auf der vorherigen Seite ergibt sich, daß die Schaltfolgen $\lambda(\sigma_2)$ und $\lambda(\sigma_3)$ ausschließlich aus Schaltinstanzen unsichtbarer Transitionen bestehen und somit ein illegaler Livelock in $\Upsilon_{S\times\neg\varphi}$ nachgewiesen wurde. ■

Für den Nachweis der Vollständigkeit des Tableau-Systems $\mathcal{T}$ bezüglich illegaler Livelocks werden zunächst noch die Definition 6.3.5 und die Proposition 6.3.3 benötigt.

Definition 6.3.5 (L-Konfiguration einer Entfaltung [EH01, Hel02])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ und dessen Entfaltung β. Eine Konfiguration C von β heißt *L-Konfiguration*, falls $BL(C) \neq \emptyset$ und $C \setminus BL(C)$ mindestens $|\mathcal{R}(M_0)| + 1$ Ereignisse enthält.

Proposition 6.3.3 (Eigenschaften von L-Konfigurationen [EH01, Hel02])
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ und dessen Entfaltung $\beta = (N', \lambda, Min(N'))$.

(i) $\Upsilon_{S\times\neg\varphi}$ *weist genau dann einen illegalen Livelock* $M_0 \xrightarrow{\sigma} M \xrightarrow{\sigma_1}$ *auf, wenn es in* β *eine L-Konfiguration* C *gibt, sodaß* σ' *eine Linearisierung von* $BL(C)$ *bezeichnet, für die* $\lambda(\sigma') = \sigma$ *gilt.*

(ii) Eine L-Konfiguration von β *enthält mindestens ein erfolgreiches Terminal vom Typ (ii)(b).*

Beweis: Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$ und dessen Entfaltung $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$.

(i) ($\Rightarrow$) Wenn $M_0 \xrightarrow{\sigma} M \xrightarrow{\sigma_1}$ einen illegalen Livelock von $\Upsilon_{S\times\neg\varphi}$ beschreibt, dann gibt es die Zerlegung $\sigma = \sigma' t^\varsigma$ mit $t \in T_L$, und es gilt $\sigma_1 \in \{t^\varsigma \in \mathcal{I}_T \mid t \in T_H\}^\omega$. Nun bezeichne C eine unendliche Konfiguration von β derart, daß $\sigma_2\sigma_3$ eine Linearisierung von C beschreibt, welche die Eigenschaft $\lambda(\sigma_2) = \sigma$ und $\lambda(\sigma_3) = \sigma_1$ aufweist. Nach Proposition 6.3.2 (iii) auf Seite 208 bezeichnet dann σ_2 eine Linearisierung von $BL(C)$. Da C unendlich ist, enthält die Menge $C \setminus BL(C)$ ebenfalls unendlich viele Ereignisse. Nun werde eine Erweiterung $BL(C) \oplus E'$ von $BL(C)$ derart gewählt, sodaß $|\mathcal{R}(M_0)| + 1$ Ereignisse in E' vorkommen. Dann beschreibt $BL(C) \oplus E'$ eine L-Konfiguration von β.

($\Leftarrow$) Gegeben seien eine L-Konfiguration C von β und eine Linearisierung σ von $BL(C)$. Daraus folgt unmittelbar die Existenz des Ablaufs

$$M_0 \xrightarrow{\lambda(\sigma)} Mark(BL(C))$$

in $\Upsilon_{S\times\neg\varphi}$. Mit Proposition 6.3.2 (i) auf Seite 208 ergibt sich die Zerlegung $\sigma = \sigma' e$, wobei $\lambda(e) \in \{t^\varsigma \in \mathcal{I}_T \mid t \in T_L\}$. Daraus folgt unmittelbar

$$M_0 \xrightarrow{\lambda(\sigma')t^\varsigma} Mark(BL(C)),$$

wobei $t \in T_L$. Da die Menge $C \setminus BL(C)$ mindestens $|\mathcal{R}(M_0)|+1$ Ereignisse enthält, folgt mit dem Schubfachprinzip die Existenz zweier Ereignisse $e_1, e_2 \in C \setminus BL(C)$ mit

$$Mark([e_1]) = Mark([e_2]). \tag{6.3.8}$$

Aufgrund der strengen Sicherheit von $\Upsilon_{S\times\neg\varphi}$ können die Ereignisse e_1 und e_2 nicht nebenläufig sein, sondern müssen kausal angeordnet sein. Ohne Beschränkung der Allgemeinheit soll $e_1 < e_2$ angenommen werden. Dann gibt es aber den Ablauf

$$M_0 \xrightarrow{\lambda(\sigma')t^\varsigma} Mark(BL(C)) \xrightarrow{\lambda(\sigma_1)} Mark([e_1]) \xrightarrow{\lambda(\sigma_2)} Mark([e_2])$$

in $\Upsilon_{S\times\neg\varphi}$, wobei $\sigma\sigma_1$ und $\sigma\sigma_1\sigma_2$ Linearisierungen von $[e_1]$ bzw. $[e_2]$ beschreiben und zumindest σ_2 nicht leer ist. Mit Gleichung 6.3.8 folgt dann unmittelbar

$$M_0 \xrightarrow{\lambda(\sigma')t^\varsigma} Mark(BL(C)) \xrightarrow{\lambda(\sigma_1)} Mark([e_1]) \xrightarrow{(\lambda(\sigma_2))^\omega}.$$

Da σ_1 und σ_2 nach Proposition 6.3.2 (ii) auf Seite 208 ausschließlich aus Ereignissen bestehen, die Schaltinstanzen unsichtbarer Transitionen repräsentieren, wurde ein illegaler Livelock nachgewiesen.

(ii) Gegeben sei eine L-Konfiguration C von β. Da $C \setminus BL(C)$ mindestens $|\mathcal{R}(M_0)|+1$ Ereignisse enthält, gibt es nach dem Schubfachprinzip zwei Ereignisse $e_1, e_2 \in C \setminus BL(C)$ mit

$$Mark([e_1]) = Mark([e_2]).$$

Aus $e_1, e_2 \in C \setminus BL(C)$ und Proposition 6.3.2 (i) auf Seite 208 folgen unmittelbar $e_1, e_2 \in E_{II}$ (siehe Definition 6.3.2 auf Seite 200) und

$$BL([e_1]) = BL(C) = BL([e_2]).$$

Da $\prec_{LTL-X}$ eine Ordnung bildet, kann ohne Beschränkung der Allgemeinheit $[e_1] \prec_{LTL-X} [e_2]$ angenommen werden. Aufgrund der strengen Sicherheit von $\Upsilon_{S\times\neg\varphi}$ können sich die Ereignisse e_1 und e_2 nicht nebenläufig verhalten, sondern müssen kausal angeordnet sein, d.h., $e_1 < e_2$. Daraus folgt $[e_1] \subset [e_2]$, und somit wurde e_2 als erfolgreiches Terminal vom Typ (ii)(b) identifiziert.

■

Satz 6.3.5 (Vollständigkeit von $\mathcal{T}$ bezüglich illegaler Livelocks)
Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi}$ und dessen Tableau-System $\mathcal{T}$. Wenn $\Upsilon_{S\times\neg\varphi}$ einen illegalen Livelock aufweist, dann gibt es in $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (ii)(b).

Beweis: Gegeben seien ein Produktnetzsystem $\Upsilon_{S\times\neg\varphi} = (N, M_0)$ mit $N = (S, T, F, \iota)$, dessen Entfaltung $\beta = (N', \lambda, Min(N'))$ mit $N' = (B, E, F')$ und dessen Tableau-System $\mathcal{T}$.
Nach Proposition 6.3.3 (i) genügt es zu zeigen, daß $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (ii)(b) enthält, falls es in β eine L-Konfiguration gibt.
Dabei wird die folgende Beweisstrategie verfolgt: Für eine L-Konfiguration C von β wird gezeigt, daß entweder C ein erfolgreiches Terminal vom Typ (ii)(b) enthält, das auch in $\mathcal{T}$ vorkommt, oder es eine L-Konfiguration C' von β mit $C' \prec_{LTL-X} C$ gibt. Da $\prec_{LTL-X}$ nach Definition 6.3.1 auf Seite 199 eine adäquate Ordnung darstellt und somit nach Definition 3.1.8 auf Seite 44 fundiert ist, enthält $\mathcal{T}$ ein erfolgreiches Terminal vom Typ (ii)(b).
Nach Proposition 6.3.3 (ii) auf Seite 209 enthält die L-Konfiguration C von β ein erfolgreiches Terminal e vom Typ (ii)(b). Dabei kann e ohne Einschränkung der Allgemeinheit derart gewählt werden, daß es das bezüglich der Kausalrelation $<$ minimale erfolgreiche Terminal vom Typ (ii)(b) in C darstellt. Falls e auch in $\mathcal{T}$ vorkommt, ist der Beweis beendet. Andernfalls muß es in C ein Terminal $e_1 < e$ geben, das aufgrund der Minimalität von e nicht erfolgreich sein kann.
Im folgenden wird nun eine L-Konfiguration C' mit der Eigenschaft $C' \prec_{LTL-X} C$ konstruiert. Dazu werden die drei Fälle betrachtet, deretwegen das Terminal e_1 als nicht erfolgreich eingestuft wurde.

- e_1 bezeichnet ein Terminal vom Typ (i)(a) oder (i)(b).
  Nun bezeichne C_1 die Konfiguration von β, deretwegen e_1 als Terminal identifiziert

wurde. Wegen

$$Mark(C_1) = Mark([e_1]) \tag{6.3.9}$$

gibt es einen Isomorphismus f zwischen $\uparrow C_1$ und $\uparrow[e_1]$. Als nächstes wird die Konfiguration

$$C' = C_1 \oplus f(C \setminus [e_1])$$

definiert. Es bleibt zu zeigen, daß es sich bei C' um eine L-Konfiguration von β handelt, die der Eigenschaft $C' \prec_{LTL-X} C$ genügt. Da $e_1 \in E_I$ (siehe Definition 6.3.2 auf Seite 200) gilt und e_1 insbesondere keine Schaltinstanz einer Transition aus T_L repräsentiert, folgt

$$BL(C) = [e_1] \oplus E'$$

für eine nicht leere Menge E' von Ereignissen. Desweiteren gelten

$$BL(C') = C_1 \oplus f(E') \tag{6.3.10}$$

$$C' \setminus BL(C') = f(C \setminus BL(C)) \tag{6.3.11}$$

Aus $E \neq \emptyset$ folgt unmittelbar $BL(C') \neq \emptyset$. Aus Gleichung 6.3.11 folgt

$$|C' \setminus BL(C')| = |C \setminus BL(C)|.$$

Dann enthält die Menge $C' \setminus BL(C')$ auch mindestens $|\mathcal{R}(M_0)|+1$ Ereignisse. Somit beschreibt C' eine L-Konfiguration von β. Aus $C_1 \prec_{LTL-X} [e_1]$, Gleichung 6.3.9 und der dritten Bedingung von Definition 3.1.8 auf Seite 44 folgt

$$BL(C') \prec_{LTL-X} BL(C).$$

Daraus folgt dann mit der ersten Bedingung von Definition 6.3.1 auf Seite 199 die Eigenschaft $C' \prec_{LTL-X} C$.

- e_1 bezeichnet ein Terminal vom Typ (ii)(a).
  Bezeichne C_1 die lokale Konfiguration, deretwegen e_1 als Terminal identifiziert wurde. Aus

  $$Mark(C_1) = Mark([e_1]) \tag{6.3.12}$$

  und $[e_1] \in E_{II}$ (siehe Definition 6.3.2 auf Seite 200) folgt unmittelbar

  $$C_1 \cap E_{II} \neq \emptyset. \tag{6.3.13}$$

  Nun bezeichne σ eine Linearisierung von $BL(C)$. Daraus folgt unmittelbar die Existenz des Ablaufs

  $$M_0 \xrightarrow{\lambda(\sigma)} Mark(BL(C))$$

  in $\Upsilon_{S \times \neg\varphi}$. Mit Proposition 6.3.2 (i) auf Seite 208 ergibt sich die Zerlegung $\sigma = \sigma' e$, wobei $\lambda(e) \in \{t^s \in \mathcal{I}_T \mid t \in T_L\}$. Daraus folgt unmittelbar

  $$M_0 \xrightarrow{\lambda(\sigma')t^s} Mark(BL(C)),$$

wobei $t \in T_L$. Da das Ereignis e ein Terminal vom Typ (ii)(b) darstellt, gibt es in β eine Konfiguration C_2, sodaß

$$\begin{aligned} Mark(C_2) &= Mark([e]) \\ C_2 &\prec_{LTL-X} [e] \\ C_2 &\subset [e] \\ C_2 \cap E_{II} &\neq \emptyset \end{aligned} \tag{6.3.14}$$

Dann gibt es aber den Ablauf

$$M_0 \xrightarrow{\lambda(\sigma')t^\varsigma} Mark(BL(C)) \xrightarrow{\lambda(\sigma_1)} Mark(C_2) \xrightarrow{\lambda(\sigma_2)} Mark([e])$$

in $\Upsilon_{S\times\neg\varphi}$, wobei $\sigma\sigma_1$ und $\sigma\sigma_1\sigma_2$ Linearisierungen von C_2 bzw. $[e]$ beschreiben und zumindest σ_2 nicht leer ist. Mit Gleichung 6.3.14 folgt dann unmittelbar

$$M_0 \xrightarrow{\lambda(\sigma')t^\varsigma} Mark(BL(C)) \xrightarrow{\lambda(\sigma_1)} Mark(C_2) \xrightarrow{(\lambda(\sigma_2))^\omega} .$$

Da σ_1 und σ_2 nach Proposition 6.3.2 (ii) auf Seite 208 ausschließlich aus Ereignissen bestehen, die Schaltinstanzen unsichtbarer Transitionen repräsentieren, wurde ein illegaler Livelock nachgewiesen.
Da $e_1 < e$ gibt es mit Gleichung 6.3.14 auch den illegalen Livelock

$$M_0 \xrightarrow{\lambda(\sigma')t^\varsigma} Mark(BL(C)) \xrightarrow{\lambda(\sigma_3)} Mark([e_1]) \xrightarrow{\lambda(\sigma_4)} Mark([e]) \xrightarrow{(\lambda(\sigma_2))^\omega} \tag{6.3.15}$$

in $\Upsilon_{S\times\neg\varphi}$, wobei $\sigma\sigma_3$ und $\sigma\sigma_3\sigma_4$ Linearisierungen von $[e_1]$ bzw. $[e]$ beschreiben.
Da C_1 eine lokale Konfiguration von β beschreibt, gibt es einen Ablauf

$$M_0 \xrightarrow{\lambda(\sigma_5)} Mark(BL(C_1)) \xrightarrow{\lambda(\sigma_6)} Mark(C_1),$$

in $\Upsilon_{S\times\neg\varphi}$, wobei σ_5 und $\sigma_5\sigma_6$ Linearisierungen von $BL(C_1)$ bzw. C_1 beschreiben. Zusammen mit Gleichung 6.3.12 auf der vorherigen Seite und Zeile 6.3.15 folgt daraus

$$M_0 \xrightarrow{\lambda(\sigma_5)} Mark(BL(C_1)) \xrightarrow{\lambda(\sigma_6)} Mark(C_1) \xrightarrow{\lambda(\sigma_4)} Mark([e]) \xrightarrow{(\lambda(\sigma_2))^\omega}$$

in $\Upsilon_{S\times\neg\varphi}$. Aus Gleichung 6.3.13 auf der vorherigen Seite und Proposition 6.3.2 (i) auf Seite 208 folgt, daß es eine Zerlegung $\sigma_5 = \sigma_5' e_5$ mit $\lambda(e_5) \in \{t^\varsigma \in \mathcal{I}_T \mid t \in T_L\}$ gibt. Da zudem die Linearisierungen σ_2, σ_4 und σ_6 nach Proposition 6.3.2 (ii) auf Seite 208 ausschließlich Ereignisse enthalten, die Schaltinstanzen unsichtbarer Transitionen repräsentieren, wurde erneut ein illegaler Livelock nachgewiesen.
Nach Proposition 6.3.3 (i) auf Seite 209 gibt es eine L-Konfiguration C' von β, sodaß σ_5 eine Linearisierung von $BL(C')$ beschreibt.
Als letzter Schritt muß noch die Eigenschaft $C' \prec_{LTL-X} C$ nachgewiesen werden.

Da σ_5 einen Präfix von $\sigma_5\sigma_6$ beschreibt, folgt daraus unmittelbar $BL(C') \subseteq C_1$. Somit gilt

$$\begin{aligned} BL(C') &= BL(C_1) && (\text{da } BL(C') \subseteq C_1) \\ &\prec_{LTL-X} BL([e_1]) && (\text{da } e_1 \text{ Terminal vom Typ (ii)(a)}) \\ &= BL(C) && (\text{da } BL(C) \subseteq [e_1]) \end{aligned}$$

Daraus folgt nach Definition 6.3.1 auf Seite 199 unmittelbar $C' \prec_{LTL-X} C$.

- e_1 bezeichnet ein Terminal vom Typ (ii)(c).
  Bezeichne C_1 die Konfiguration von β, deretwegen e_1 als Terminal identifiziert wurde. Wegen
  $$Mark(C_1) = Mark([e_1]) \tag{6.3.16}$$
  gibt es einen Isomorphismus f zwischen $\uparrow C_1$ und $\uparrow[e_1]$. Als nächstes wird die Konfiguration
  $$C' = C_1 \oplus f(C \setminus [e_1]) \tag{6.3.17}$$
  definiert. Es bleibt zu zeigen, daß es sich bei C' um eine L-Konfiguration von β handelt, die der Eigenschaft $C' \prec_{LTL-X} C$ genügt.
  Da e_1 ein Terminal vom Typ (ii)(c) bezeichnet, gilt $e_1 \in E_{II}$ (siehe Definition 6.3.2 auf Seite 200). Mit Gleichung 6.3.16 ergibt sich daraus $C_1 \cap E_{II} \neq \emptyset$. Dann gilt
  $$BL(C) = BL([e_1]) \tag{6.3.18}$$
  $$BL(C') = BL(C_1) \tag{6.3.19}$$
  Da e_1 ein Terminal vom Typ (ii)(c) bezeichnet, gilt $BL(C_1) = BL([e_1])$, woraus mit den Gleichungen 6.3.18 und 6.3.19 unmittelbar
  $$BL(C') = BL(C) \tag{6.3.20}$$
  folgt. Da C eine L-Konfiguration von β darstellt, folgt aus $BL(C) \neq \emptyset$ auch $BL(C') \neq \emptyset$. Da e_1 ein Terminal vom Typ (ii)(c) bezeichnet, gilt $|C_1| \geq |[e_1]|$, woraus mit
  $$C = [e_1] \oplus (C \setminus [e_1]), \tag{6.3.21}$$
  den Gleichungen 6.3.17 und 6.3.20 und der Tatsache, daß C eine L-Konfiguration von β beschreibt, auch unmittelbar
  $$|C' \setminus BL(C')| \geq |C \setminus BL(C)| \geq |\mathcal{R}(M_0) + 1|$$
  folgt. Somit bezeichnet C' eine L-Konfiguration von β. Schließlich folgt mit den Gleichungen 6.3.17 und 6.3.21 und der dritten Bedingung von Definition 3.1.8 auf Seite 44 aus $C_1 \prec_{LTL-X} [e_1]$ unmittelbar $C' \prec_{LTL-X} C$.

■

6.4 Experimentelle Ergebnisse

In diesem Abschnitt werden experimentelle Ergebnisse für die Verifikation von LTL-X-Eigenschaften von M-Netzsystemen präsentiert und diskutiert. Dabei wird das in den vorangegangenen Abschnitten beschriebene Verfahren PUNF [SK04] mit dem Model-Checker SPIN [Hol03] und dem in [EH01, Hel02] implementierten LTL-X Model-Checker UNFSMODELS für S/T-Netzsysteme verglichen.
Alle Testreihen wurden auf einem Linux PC durchgeführt, der mit einem 2,4 GHz Intel(R) Xeon(TM) Prozessor und 4 GByte Hauptspeicher ausgestattet war.
Der Model-Checker PUNF wurde in der Version 7.03 (parallel) verwendet. Damit jedoch ein fairer Vergleich mit SPIN und UNFSMODELS vorgenommen werden konnte, wurde PUNF mit der Option *-N=1* ausgeführt, was zur Folge hatte, daß die Entfaltungsroutine auf nur einem Prozessor ablaufen und somit keine Parallelisierung stattfinden konnte.
Der Model-Checker UNFSMODELS wurde in der Version 0.9 vom 22. Oktober 2003 verwendet und in allen Testreihen mit der Option *-l* (für LTL-X Model-Checking) ausgeführt.
Der Model-Checker SPIN wurde in der Version 4.0.7 vom 01. August 2003 verwendet und in allen Testreihen mit den Optionen *-DMEMCNT=32* (ermöglicht die Ausnutzung der gesamten 4 GByte des Hauptspeichers) und *-DNOFAIR* (es werden keine Fairneß-Annahmen getroffen) ausgeführt. Darüberhinaus wurden die Halbordnungsreduktionstechniken von SPIN standardmäßig in allen Testreihen aktiviert.
Da LTL-X auf ω-Wörtern (also unendlichen Abläufen) interpretiert wird, wurden in den Testreihen ausschließlich verklemmungsfreie Systeme verwendet. Eine Beschreibung aller getesteten Systeme kann Anhang A auf Seite 223 entnommen werden.
Um einen fairen Vergleich mit dem Model-Checker SPIN durchführen zu können, wurden alle Systeme (außer BUFFER(n)) in dessen Spezifikationssprache *Promela* modelliert. Die Promela-Modellierungen aus [Cor94] wurden unverändert übernommen, jedoch teilweise skaliert.
Die M-Netzsysteme, welche dem Model-Checker PUNF als Eingabe dienten, wurden mittels des Programms SPIN2CFA aus der von SPIN erzeugten Automatendarstellung der Systeme ohne weitere Optimierung hergeleitet, indem die von SPIN generierten Automaten in die Automatendarstellung CFA (*communicating finite automata*) des PEP-Tools [GRT+95] überführt und anschließend mit der PEP-Tool-Routine CFA2MNET in streng sichere M-Netzsysteme transformiert wurden. Die Automatenrepräsentation der Promela-Modelle wurde erzeugt, indem SPIN mit den Optionen *-a* (erzeugt das Verifikationsprogramm PAN) und *-o3* (verbietet die Verschmelzung mehrerer Anweisungen zu einer einzelnen Anweisung) ausgeführt wurde. Anschließend ermöglichte ein Aufruf von PAN mit der Option *-d* den Zugriff auf die Zustandstabellen (die interne Automatendarstellung) der Promela-Modelle.
Die sicheren S/T-Netzsysteme, welche dem Model-Checker UNFSMODELS als Eingabe dienten, wurden aus den M-Netzsystemen mittels der PEP-Tool-Routine HL2LL generiert.
Innerhalb der Testreihen wurden die folgenden LTL-X-Eigenschaften verwendet:

Tabelle 6.1 *Ergebnisse für die Verifikation von LTL-X-Eigenschaften*

System	LTL-X-Eigenschaft	Ergebnis	PUNF	UNFSMODELS	SPIN
ABP	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	0.08	0.19	0.01
BDS	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	8.47	199.22	0.71
DPD(7)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	7.25	506.51	2.14
FURNACE(3)	$\Diamond\Box\pi$	**wahr**	26.90	1056.62	1.00
GASNQ(4)	$\Diamond\Box\pi$	**wahr**	8.46	239.60	0.14
RW(12)	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	47.67	2770.07	0.44
FTP(1)	$\Diamond\Box\pi$	**wahr**	836.07	> 12000.00	3.99
OVER(5)	$\Diamond\Box\pi$	**wahr**	0.12	66.01	0.44
CYCLIC(12)	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	0.08	0.38	11.25
RING(9)	$\Diamond\Box\pi$	**wahr**	0.13	2.13	1.64
DP(12)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.36	13.05	116.53
PH(12)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.02	0.04	0.61
COM(15)(0)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.02	-	3.11
PAR(5)(10)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.02	-	3.60

- $\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$
  („*irgendwann werden* π_1, π_2 *und* π_3 *nie mehr gleichzeitig gelten*“)
- $\Diamond\Box\pi$
  („*irgendwann wird* π *immer gelten*“)
- $\Box(\pi_1 \rightarrow \Diamond\pi_2)$
  („*immer, wenn* π_1 *gilt, wird auch irgendwann* π_2 *gelten*“)
- $\Box(\pi_1 \rightarrow \Box\neg\pi_2)$
  („*immer, wenn* π_1 *gilt, wird* π_2 *nie mehr gelten*“)

Die Ergebnisse sind in Tabelle 6.1 dargestellt, wobei die Zeiten jeweils in Sekunden angegeben sind. Sie zeigen, daß der Model-Checker PUNF für M-Netzsysteme dem Model-Checker UNFSMODELS für S/T-Netzsysteme in allen Testreihen überlegen ist, teilweise mit einem Geschwindigkeitszuwachs bis zu einem Faktor von 550 (OVER(5)). Die Systeme COM(10)(0) und PAR(5)(10) konnten nicht mit UNFSMODELS verifiziert werden, da im Falle von COM(10)(0) der die bezüglich des S/T-Netzsystems negierte Formel repräsentierende Büchiautomat und im Falle von PAR(5)(10) dessen S/T-Netzsystem nicht mehr generiert werden konnten. Der Vergleich zwischen PUNF und SPIN verdeutlicht, daß SPIN bei den Beispielen in der oberen Tabellenhälfte eine deutlich bessere Performance aufweist. Im Gegensatz dazu scheint PUNF bei den Modellen in der unteren Tabellenhälfte einen Vorteil zu besitzen. Um dieses genauer zu untersuchen, wurden diese Systeme skaliert und erneut Testreihen unterzogen.

Tabelle 6.2 *Ergebnisse für die Verifikation von LTL-X-Eigenschaften, wobei die Systemmodelle skaliert wurden*

System	LTL-X-Eigenschaft	Ergebnis	PUNF	SPIN
CYCLIC(12)	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	0.08	11.25
CYCLIC(15)	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	0.08	168.40
CYCLIC(16)	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	0.07	477.65
CYCLIC(17)	$\Box(\pi_1 \rightarrow \Diamond\pi_2)$	**wahr**	0.10	1601.00
RING(9)	$\Diamond\Box\pi$	**wahr**	0.13	1.64
RING(12)	$\Diamond\Box\pi$	**wahr**	0.30	75.38
RING(13)	$\Diamond\Box\pi$	**wahr**	0.50	274.13
RING(14)	$\Diamond\Box\pi$	**wahr**	0.85	1267.36
DP(12)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.36	116.53
DP(13)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.53	559.27
DP(14)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.75	2123.41
PH(12)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.02	0.61
PH(15)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.01	16.69
PH(18)	$\Diamond\Box\neg(\pi_1 \wedge \pi_2 \wedge \pi_3)$	**wahr**	0.01	1570.24
COM(15)(0)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.02	3.11
COM(20)(0)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.02	232.44
COM(21)(0)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.03	685.90
COM(22)(0)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.02	2279.14
PAR(5)(10)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.02	3.60
PAR(6)(10)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.02	160.63
PAR(7)(10)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	0.04	*mem*

Die Ergebnisse in Tabelle 6.2 bestätigen, daß die LTL-X-Eigenschaften für diese Systeme mit PUNF deutlich schneller als mit SPIN verifiziert werden können. Während die Verifikationszeiten für SPIN exponentiell anzusteigen scheinen, können die LTL-X-Eigenschaften für alle Systeme mit PUNF in weniger als einer Sekunde nachgewiesen werden. Alle Systeme weisen einen hohen Grad an Nebenläufigkeit auf, und die Ergebnisse bestätigen, daß der entfaltungsbasierte Ansatz für die Verifikation solcher Systeme sehr geeignet ist. Zugegebenermaßen weisen die Promela-Modelle von COM(N)(M) und PAR(N)(M) einen eher künstlichen Charakter auf, jedoch sollte anhand dieser Systeme demonstriert werden, daß entfaltungsbasierte Techniken ausgezeichnet Systeme handhaben können, die einen hohen Grad an Nebenläufigkeit aufweisen. Im Gegensatz dazu scheinen sich die Halbordnungsreduktionstechniken von SPIN in diesen Beispielen weniger effizient auszuwirken. Ein möglicher Erklärungsansatz könnte in der Reduktionsbedingung für LTL Model-Checking gefunden werden, die in der Literatur als *cyclic condition* [CGP99] bzw. *reduction proviso* [Pel93] bekannt ist. Diese Bedingung kann beispielsweise bei Systemen, die einen eher zyklischen Charakter aufweisen, bewirken,

Tabelle 6.3 *Ergebnisse für die Verifikation von LTL-X-Eigenschaften, wobei* PUNF *im parallelen Modus ausgeführt wurde*

System	LTL-X-Eigenschaft	Ergebnis	PUNF(1)	PUNF(2)	SPIN
COM(20)(3)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	8.58	6.01	*mem*
COM(22)(3)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	11.51	8.51	*mem*
COM(25)(3)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	17.29	12.84	*mem*
PAR(20)(100)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	8.60	4.84	*mem*
PAR(20)(150)	$\Box(\pi_1 \rightarrow \Box\neg\pi_2)$	**wahr**	31.98	18.28	*mem*
BUFFER(20)	$\Diamond\Box\pi$	**wahr**	22.70	16.95	-
BUFFER(25)	$\Diamond\Box\pi$	**wahr**	142.72	89.40	-

daß der gesamte Zustandsraum aufgespannt werden muß.
In [HKK02] wurde die Entfaltungsroutine für S/T-Netzsysteme parallelisiert. Dieser Ansatz ist ebenfalls in die Entfaltungsroutine für M-Netzsysteme integriert worden, und um die Auswirkungen zu demonstrieren, wurden erneut Testreihen für Systeme mit einem hohen Grad an Nebenläufigkeit auf einem Rechner mit 2 Prozessoren durchgeführt. Die Ergebnisse sind in Tabelle 6.3 dargestellt, wobei PUNF(n) bedeutet, daß PUNF während der Entfaltung n Prozessoren zur Verfügung standen. Die Ergebnisse verdeutlichen, daß die innerhalb der Systeme vorhandene Nebenläufigkeit von PUNF ausgenutzt werden kann, indem die Entfaltungsroutine, soweit möglich, parallel auf den zur Verfügung stehenden Prozessoren abläuft. Somit könnte die Verifikationszeit bei Bereitstellung von n Prozessoren im Idealfall, der aber in der Praxis mit hoher Wahrscheinlichkeit nicht eintreten wird, um das n-fache verringert werden. Desweiteren zeigen die Ergebnisse, daß 4 GByte Hauptspeicher nicht ausreichen, um diese Systeme mit SPIN zu verifizieren.[4]

[4]Die BUFFER(n)-Systme konnten nicht mit SPIN verifiziert werden, da sie nicht als Promela-, sondern nur als M-Netz-Modelle vorlagen.

Kapitel 7

Abschließende Bemerkungen

Formale Verifikationsmethoden, und hier insbesondere Model-Checking-Techniken, stellen für Entwickler effiziente Werkzeuge dar, die unbedingt bei der System- und Softwareentwicklung begleitend als qualitätssichernde Maßnahmen eingesetzt werden sollten. In dieser Arbeit wurden mehrere Model-Checking-Verfahren vorgestellt und anhand von Fallbeispielen bezüglich ihrer praxisorientierten Anwendbarkeit analysiert. Dabei wurden sowohl exakte Verfahren und Halbentscheidungsverfahren betrachtet, welche die Erreichbarkeit von Systemzuständen untersuchen, als auch Reduktions- und Entscheidungsmethoden für den Nachweis temporallogischer Eigenschaften vorgestellt.
Im einzelnen wurden die folgenden Ergebnisse erzielt:

1. Es wurde nachgewiesen, daß das PSPACE-vollständige Erreichbarkeitsproblem für sichere S/T-Netzsysteme unter Verwendung von Netzentfaltungen in ein NP-vollständiges Problem überführt werden kann. Desweiteren wurden zwei Verfahren zur Erreichbarkeitsanalyse unter Verwendung von Netzentfaltungen erweitert und implementiert. Anhand von experimentellen Ergebnissen konnte verdeutlicht werden, daß die beiden Algorithmen bei der Verifikation von mehreren Systemen eine bessere Performance aufweisen als andere vergleichbare Methoden.

2. Es wurden zwei Halbentscheidungsverfahren erweitert und implementiert, die das Erreichbarkeitsproblem für sichere S/T-Netzsysteme mittels Techniken der linearen Programmierung anhand der strukturellen Konzepte von Fallen und Schaltnetzen zu lösen versuchen. Anhand von experimentellen Ergebnissen konnte gezeigt werden, daß beide Verfahren eine praktische Relevanz aufweisen, obwohl sie die Erreichbarkeitsfrage nicht exakt in dem Sinne beantworten, daß immer eine „Ja/Nein“-Entscheidung getroffen werden kann.

3. Es wurden Reduktionstechniken erweitert und implementiert, mittels derer sichere

S/T-Netzsysteme in dem Sinne „verhaltensäquivalent“ reduziert werden können, daß deren Verhalten bezüglich temporallogischer Eigenschaften bewahrt bleibt. Anhand von experimentellen Ergebnissen konnte nachgewiesen werden, daß die reduzierten Netzsysteme kleinere Präfixe erzeugen und dadurch die Verifikation von temporallogischen Eigenschaften mittels des in [EH01, Hel02] diskutierten Verfahrens erheblich beschleunigt werden kann.

4. Der in [EH01, Hel02] vorgestellte Ansatz zum Nachweis von LTL-X-Eigenschaften für sichere S/T-Netzsysteme wurde auf M-Netzsysteme erweitert. Anhand von experimentellen Ergebnissen wurde gezeigt, daß mit dieser Methode viele Systeme effizienter verifiziert werden können als beispielsweise mit SPIN [Hol03], der zu den bekanntesten und angesehensten Model-Checkern in dem Bereich der automatischen Verifikation verteilter Systeme gehört.

Als Gesamtergebnis bleibt festzuhalten, daß es unter den vorgestellten Verfahren, sofern sie denn vergleichbar sind, kein „bestes“ gibt, das universell einsetzbar ist und für alle betrachteten Systeme immer die besten Ergebnisse erzielt. Vielmehr haben die experimentellen Analysen gezeigt, daß es sowohl bei den Algorithmen für Erreichbarkeitsfragen als auch bei den LTL-X Model-Checkern vorkommt, daß mal die eine und dann wiederum eine andere Methode die bessere Performance aufweist. Jedoch haben die Ergebnisse auch verdeutlicht, daß es für jedes in dieser Arbeit betrachte Fallbeispiel mindestens ein Verfahren gibt, mit dessen Hilfe dieses effizient verifiziert werden kann.
Dennoch konnten im Rahmen dieser Arbeit nicht alle interessanten Fragestellungen behandel werden, und deshalb soll im folgenden ein Denkanstoß für weiterführende Arbeiten gegeben werden.
In Kapitel 5 wurden strukturelle Reduktionstechniken für S/T-Netzsysteme vorgestellt, die das Verhalten der Netzsysteme bezüglich LTL-X-Eigenschaften bewahren. Die experimentellen Ergebnisse haben gezeigt, daß sich die Netzreduktionen äußerst positiv auf den in [EH01, Hel02] beschriebenen LTL-X Model-Checking-Algorithmus auswirken, da die reduzierten Netzsysteme kleinere Präfixe aufweisen und der Verifikationsprozeß somit beschleunigt wird. Daraus ergibt sich die interessante Fragestellung, inwiefern die für S/T-Netzsysteme vorgeschlagenen Reduktionstechniken auf die Ebene der M-Netzsysteme übertragen werden können, da sich diese Vorgehensweise ebenfalls positiv auf den in Kapitel 6 diskutierten LTL-X Model-Checker für M-Netzsysteme auswirken dürfte. Ein Überblick bezüglich Reduktionstheorien für gefärbte Petrinetzsysteme findet sich in [Had90].
Eine weitere interessante Aufgabenstellung könnte darin bestehen, den LTL-X Model-Checker für M-Netzsysteme durch den Einsatz von Abstraktionstechniken zu optimieren. Jeder Stelle eines M-Netzsystems ist ein Datentyp zugeordnet. Soll nun aber beispielsweise die Eigenschaft überprüft werden, ob eine Stelle immer einen Wert größer als Null aufweist, ist es für die Verifikation einer solchen Eigenschaft unerheblich, welchen Wert genau die Stelle zu einem bestimmten Zeitpunkt aufweist, sondern es ist lediglich interessant, ob der Wert größer als Null ist. Folglich kann von den Datentypen der Stellen derart abstrahiert werden, daß eine Stelle keinen konkreten Wert ihres Datentyps mehr

zugewiesen bekommt, sondern lediglich mit den Prädikaten $(s > 0)$ und $(s \leq 0)$ als „Werten“ gerechnet wird. Diese Technik wird als *Prädikatenabstraktion* bezeichnet und wurde erstmals in [GS97] beschrieben und später beispielsweise innerhalb des SLAM-Projektes von Microsoft-Research zur automatischen Abstraktion von C-Programmen eingesetzt [BMMR01]. Die grundlegende Idee besteht darin, konkrete Systemzustände unter Evaluierung einer endlichen Menge von Prädikaten auf abstrakte Zustände abzubilden und somit den zu betrachtenden Zustandsraum einzuschränken. Angewendet auf die Datentypen der M-Netzsysteme dürfte diese Vorgehensweise zu kleineren Präfixen führen und müßte die Verifikation mit dem in Kapitel 6 beschriebenen LTL-X Model-Checker beschleunigen.

Anhang A

Fallbeispiele

In diesem Abschnitt werden alle in dieser Arbeit verwendeten Fallbeispiele aufgelistet und beschrieben.

- ABP: Modellierung eines alternierenden Bit-Protokolls mit 6 Prozessen, die zwei Teilnehmer, einen Sender, einen Empfänger und zwei verlustreiche Kanäle repräsentieren [Cor94].

- BDS: Modellierung des Kommunikationsgerüsts eines realen Ada-Programms, das ein Grenzgebietverteidigungssystem simuliert. Das System besteht aus 15 Prozessen [Cor94].

- BUFFER(n): Modellierung eines Speichers der Kapazität n, in dem n verschiedene Werte abgelegt werden können. Das System für BUFFER(2) ist in Abbildung 6.4 (a) auf Seite 188 dargestellt (Schröter).

- COM(n)(m): Modellierung von n endlos kreisenden Prozessen, die in einer Kette angeordnet sind. Jeder Prozeß kommuniziert mit seinen Nachbarn durch Kanäle der Kapazität m (Schröter).

- CYCLIC(n): Modellierung von Milners zyklischem Scheduler mit n Scheduler- und n Kunden-Prozessen [Cor94].

- DAC(n): Modellierung einer „teile-und-herrsche“-Berechnung mit n parallelen Prozessen [Cor94].

- DIJKSTRA(n): Modellierung von Dijkstras Algorithmus zur Garantierung des wechselseitigen Ausschlußses für n Prozesse [Ray86].

- DME(n): Modellierung eines Distributed-Mutual-Exclusion-Elements, das in ungetakteten Hardwareschaltungen zur Garantierung des wechselseitigen Ausschlusses für n konkurrierende Zugriffe eingesetzt wird [Mar85].

- DP(n): Modellierung n speisender Philosophen [Cor94]. Dabei wurde das Standardsystem, welches eine Verklemmung beinhaltet, in ein verklemmungsfreies System überführt, indem für einen Philosophen die Reihenfolge geändert wurde, in der er die Gabeln aufnimmt.

- DPD(n): Modellierung n speisender Philosophen, bei der die Verklemmungsfreiheit dadurch garantiert wird, daß ein Buch zwischen den Philosophen zirkuliert und der jeweils lesende Philosoph keine Gabeln aufnehmen darf [Cor94].

- DPFM(n): Modellierung n speisender Philosophen, bei der die Verklemmungsfreiheit dadaurch garantiert wird, daß ein Gabelmanager zur Zuteilung der Gabeln eingesetzt wird [Cor94].

- DPH(n): Modellierung n speisender Philosophen, die von einem Butler koordiniert werden [Cor94].

- EISENBAHN: Modellierung eines Eisenbahnnetzes (Schmidt).

- ELEVATOR(n): Modellierung einer Steuerung für n Fahrstühle. Das System mit der Eingangsgröße n besteht aus $n + 3$ Prozessen [Cor94].

- FIFO(n): Modellierung eines 1-Bit FIFO-Kanals der Kapazität n [Mar86, RCP95].

- FTP(n): Modellierung eines Programms, das Aufträge von n Teilnehmern zur Übertragung von Dateien über ein Netzwerk bearbeitet. Das System mit der Eingangsgröße n besteht aus $n + 8$ Prozessen [Cor94].

- FURNACE(n): Modellierung einer Temperatursteuerung für n Hochöfen. Das System mit der Eingangsgröße n besteht aus $2n + 6$ Prozessen [Cor94].

- GASNQ(n): Modellierung einer Selbstbedienungstankstelle mit einem Tankwart, zwei Zapfsäulen und n Kunden [Cor94].

- GASQ(n): Modellierung einer Selbstbedienungstankstelle mit einem Tankwart, zwei Zapfsäulen und n Kunden, wobei die Kunden in einer FIFO-Warteschlange eingereiht werden. [Cor94].

- HARTSTONE(n): Modellierung eines Ada-Programms, bei dem ein Prozeß beginnt und dann n Prozesse stoppt [Cor94].

- KEY(n): Modellierung eines Programms, das die Interaktion zwischen Tastatur und Monitor für einen Fenstermanager mit n Benutzeranwendungen steuert. Das System mit der Eingangsgröße n besteht aus $n + 5$ Prozessen. [Cor94].

- MMGT(n): Modellierung eines verteilten Speichermanagers für n Benutzer. Das System mit der Eingangsgröße n besteht aus $n + 4$ Prozessen [Cor94].

- OVER(n): Modellierung der Ada-Version eines automatischen Überholsystems für n Autos auf Autobahnen. Das System mit der Eingangsgröße n besteht aus $2n+1$ Prozessen [Cor94].
- PAR(n)(m): Modellierung von n nebenläufigen Prozessen, wobei jeder Prozeß eine lokale Variable modulo m hochzählt (Schröter).
- PH(n): Modellierung n speisender Philosophen, wobei $n-1$ Linkshänder und ein Rechtshänder am Tisch sitzen und dadurch die Verklemmungsfreiheit des Systems garantiert ist (Schröter).
- PRODCELL(n): Modellierung einer betrieblichen Produktionszelle, die bis zu n Metallplatten gleichzeitig bearbeiten kann [LL95, HD95].
- RING(n): Modellierung eines verteilten Algorithmus' zur Gewährleistung des wechselseitigen Ausschlusses von n Teilnehmern, die auf dieselbe Ressource zugreifen wollen. Dazu zirkuliert zwischen n Serverprozessen ein Token auf einem Ring [Cor94].
- RW(n): Modellierung einer Datenbank, die gleichzeitigen Zugriff einer beliebigen Anzahl von Lesern oder aber eines Schreibers erlaubt. Dabei werden die n lesenden und n schreibenden Teilnehmer über einen Controller synchronisiert, der deren Aktivitäten überwacht [Cor94].
- RW(n)(m): Modellierung eines skalierbaren Algorithmus' zur Synchronisation von n lesenden und m schreibenden Zugriffen paralleler Rechner auf einen gemeinsamen Speicherbereich [Hel93].
- SENTEST(n): Modellierung eines Programms zum Testen von n Sensoren. Das System mit der Eingangsgröße n besteht aus $n+4$ Prozessen [Cor94].
- SLOTRING(n): Modellierung des Slotted-Ring-Protokolls mit n Knoten [PRCB94].
- SPEED: Modellierung eines Programms mit 10 Prozessen, das die Geschwindigkeit eines Autos überwacht und reguliert [Cor94].
- TELEPHON(n): Modellierung eines Telefonprotokolls für n Telefone [Gra95].

Anhang B

Programmbeschreibungen

In diesem Abschnitt werden alle im Rahmen dieser Arbeit implementierten Programme aufgelistet, und es wird kurz deren Funktionsweise beschrieben.

B.1 Programme zur Erreichbarkeitsanalyse

Im folgenden werden alle Programme beschrieben, die im Rahmen der Kapitel 3 und 4 für die Erreichbarkeitsanalyse von Teilmarkierungen in sicheren S/T-Netzsystemen implementiert wurden.

- CheckCo
  Dieser Algorithmus löst das Erreichbarkeitsproblem für Teilmarkierungen sicherer S/T-Netzsysteme unter Verwendung der Co-Relation und wurde ausführlich in Abschnitt 3.3 auf Seite 54 diskutiert.
  Das Programm wird wie folgt aufgerufen:

  ```
  checkco <mci-file> <formula-file>
  ```

  Das MCI-Format stellt eine kompakte Beschreibung des entfalteten Netzsystems dar und kann mit nahezu jedem beliebigen Netzentfalter erzeugt werden, siehe beispielsweise [Röm00, KK01]. Allerdings sollte bei der Wahl des Entfalters darauf geachtet werden, daß dieser zusätzlich zu der MCI-Datei auch die Co-Relation der Entfaltung berechnet und explizit in eine Datei ausgibt, auf die dann wiederum CheckCo zurückgreift. Momentan wird diese Möglichkeit von dem in [Röm00] beschriebenen Entfalter angeboten.
  Die Formel-Datei enthält die auf Erreichbarkeit zu testende Teilmarkierung in Form einer durch Leerzeichen getrennten Liste von Stellennamen. Falls die Abwesenheit einer Marke von einer Stelle nachgewiesen werden soll, so wird dieses durch das dem Stellennamen vorangestellte „!“-Zeichen angezeigt.

- OnTheFly
  Dieser Algorithmus löst das Erreichbarkeitsproblem für Teilmarkierungen sicherer

S/T-Netzsysteme „on-the-fly“ während der Entfaltung der S/T-Netzsysteme und wurde ausführlich in Abschnitt 3.4.3 auf Seite 76 diskutiert.
Das Programm wird wie folgt aufgerufen:

```
onthefly <ll_net-file> <formula-file>
```

Das ll_net-Format entspricht dem Format des PEP-Tools [GRT+95] für S/T-Netzsysteme und kann mit diesem auch erzeugt werden.
Das Format der Formel-Datei ist identisch mit dem für CHECKCO beschriebenen Format.

- TRAPCHECK
  Dieser Algorithmus versucht, das Erreichbarkeitsproblem für Teilmarkierungen sicherer S/T-Netzsysteme unter Verwendung von Fallen und mittels Techniken der linearen Programmierung zu lösen und wurde ausführlich in Abschnitt 4.1.2 auf Seite 86 diskutiert.
  Das Programm wird wie folgt aufgerufen:

```
trapcheck <ll_net-file> <formula-file> [options]

options:
  -v: verbose output
  -s: save LPs in files
```

  Die Dateiformate sind identisch mit den für ONTHEFLY beschriebenen Formaten.
  Die Angabe der Option `-v` bewirkt eine Ausgabe der Lösungen aller betrachteten Ungleichungssysteme, und mit der Option `-s` werden alle Ungleichungssysteme in Dateien abgespeichert.
  Zur Lösung der linearen Ungleichungssysteme wird LPSOLVE in der Version 5.5 verwendet.

- TRAPSCHALTCHECK
  Dieser Algorithmus versucht, das Erreichbarkeitsproblem für Teilmarkierungen sicherer S/T-Netzsysteme unter Verwendung von Schaltnetzen und mittels Techniken der linearen Programmierung zu lösen und wurde ausführlich in Abschnitt 4.2.1 auf Seite 93 diskutiert.
  Das Programm wird wie folgt aufgerufen:

```
trapschaltcheck <ll_net-file> <formula-file> [options]

options:
  -v: verbose output
  -s: save LPs in files
```

Die Dateiformate und Optionen sind identisch mit den für TrapCheck beschriebenen Formaten und Optionen.
Zur Lösung der linearen Ungleichungssysteme wird LpSolve in der Version 5.5 verwendet.

B.2 Programme zur Netzreduktion

In diesem Abschnitt werden alle Programme beschrieben, die im Rahmen von Kapitel 5 zur Reduktion von sicheren S/T-Netzsystemen für die Verifikation von LTL-X-Eigenschaften implementiert wurden.

- Reduce
  Dieser Algorithmus reduziert S/T-Netzsysteme unter Verwendung von Techniken der linearen Programmierung und lokalen Netzreduktionen derart, daß LTL-X-Eigenschaften bewahrt bleiben. Die Reduktionstechniken wurden ausführlich in Abschnitt 5.2 auf Seite 130 diskutiert.
  Das Programm wird wie folgt aufgerufen:

  ```
  reduce <prod_net-file>
  ```

  Als Eingabe dient das in Abschnitt 5.1.4 auf Seite 119 beschriebene Produktnetzsystem, das für die Verifikation von LTL-X-Eigenschaften verwendet wird und beispielsweise mit dem in [Wal98] beschriebenen Model-Checker erzeugt werden kann.
  Zur Lösund der linearen Ungleichungssysteme wird Cplex^{TM} [Cpl97] in der Version 6.5 verwendet.

B.3 Programme zum LTL-X Model-Checking

In diesem Abschnitt werden alle Programme beschrieben, die im Rahmen von Kapitel 6 zur Verifikation von LTL-X-Eigenschaften von M-Netzsystem implementiert wurden.

- cfa2mnet
  Dieser Algorithmus erzeugt ein M-Netzsystem aus einem in Automatendarstellung (*Communicating Finite Automata*) vorliegenden Systemmodell.
  Das Programm wird wie folgt aufgerufen:

  ```
  cfa2mnet <prj-file> -o <output-file>
  ```

  Das Programm basiert auf der pfa2mnet-Routine des Pep-Tools [GRT+95], wurde jedoch um die Handhabung von Feldern und Kanälen mit Tupeln als Datenpaketen erweitert.

Die Projekt-Datei enthält Informationen über die globalen Variablen sowie eine Auflistung der parallelen endlichen Automaten, die in ein M-Netzsystem überführt werden sollen. Das Format ist identisch mit dem PFA-Format (*Parallel Finite Automata*) des PEP-Tools [GRT+95].
Das aus der Automatendarstellung erzeugte M-Netzsystem wird in die Ausgabedatei ausgegeben, wobei das Format mit dem Format des PEP-Tools [GRT+95] für M-Netzsysteme identisch ist.

- LTL2BNET
  Dieser Algorithmus erzeugt aus einer LTL-X-Formel und einem M-Netzsystem das in Definition 6.2.1 auf Seite 190 beschriebene Produktnetzsystem, welches für die Verifikation der LTL-X-Eigenschaft verwendet wird.
  Das Programm wird wie folgt aufgerufen:

```
ltl2bnet <options> <m_net-file> <formula-file>

options:
  -l: use LBT
  -s: use SPIN (default is LTL2BA)
```

Durch Angabe der Optionen `-l` oder `-s` können alternativ zu LTL2BA [GO01] auch LBT (basiert auf [GPVW96]) bzw. SPIN [Hol03] für die Umwandlung einer LTL-X-Formel in einen Büchiautomaten verwendet werden.
Das M-Netz-Format ist identisch mit dem im PEP-Tool [GRT+95] verwendeten Format für M-Netze.
Die Formel-Datei enthält die Beschreibung der zu verifizierenden LTL-X-Eigenschaft, wobei die folgende Syntax verwendet wird:

ltlform ::
„TRUE“ | „FALSE“
| *proposition*
| „(“ *ltlform* „)“
| *ltlform* „&“ *ltlform*
| *ltlform* „|“ *ltlform*
| *ltlform* „⇒“ *ltlform*
| „!“ *ltlform*
| „G“ *ltlform*
| „F“ *ltlform*
| *ltlform* „U“ *ltlform*

Dabei bezeichnen „G“ und „F“ die temporalen Operatoren „□“ und „◊“.
Die atomaren Propositionen einer LTL-X-Formel ergeben sich nach der folgenden

Syntax:

proposition ::
„(“ *proposition* „)“
| *aprop* „=“ *aprop*
| *aprop* „#“ *aprop*
| *aprop* „<“ *aprop*
| *aprop* „>“ *aprop*
| *aprop* „<=“ *aprop*
| *aprop* „>=“ *aprop*

aprop ::
name
| *wert*
| „*dot*“
| „(“ *aprop* „)“
| *aprop* „+“ *aprop*
| *aprop* „−“ *aprop*
| *aprop* „⋆“ *aprop*
| *aprop* „/“ *aprop*
| *aprop* „%“ *aprop*

- PUNF
  Dieser Algorithmus führt die Verifikation von LTL-X-Eigenschaften für M-Netzsysteme nach dem in den Abschnitten 6.2 auf Seite 187 und 6.3 auf Seite 199 beschriebenen Verfahren durch.
  Das Programm basiert auf einem im Rahmen von [Kho03, KK03] implementierten Entfalter für M-Netzsysteme und wurde entsprechend erweitert.
  Das Programm wird wie folgt aufgerufen:

```
punf [options] <prod_net-file>

options:
  -s: print statistics
  -N=n: specifies the number of working threads
```

  Als Eingabe dient das nach Definition 6.2.1 auf Seite 190 erstellte Produktnetzsystem, das mit LTL2BNET erzeugt werden kann.
  Durch Angabe der Option `-s` können Informationen wie beispielsweise die Größe

des erzeugten Tableau-Systems ermittelt werden. Mit der Option `-N` kann die Anzahl von Prozessoren reguliert werden, auf denen die Entfaltungsroutine nach Möglichkeit parallel rechnen soll. PUNF besitzt noch weitere Optionen, die jedoch an dieser Stelle unerwähnt bleiben, da sie im Rahmen der vorliegenden Arbeit nicht verwendet wurden.

- SPIN2CFA
  Dieser Algorithmus überführt ein Promela-Modell in Automatendarstellung.
  Das Programm wird wie folgt aufgerufen:

  ```
  spin2cfa <promela-file>
  ```

  Als Eingabe dient eine Datei, die eine Spezifikation in der dem Model-Checker SPIN [Hol03] zugrundeliegenden Eingabesprache *Promela* enthält.
  Das Programm ruft SPIN auf und erzeugt unter Verwendung dessen interner Automatenrepräsentation eine Automatendarstellung, dessen Format eng an das PFA-Format des PEP-Tools [GRT+95] angelehnt ist. Die Ausgabe kann direkt als Eingabe für CFA2MNET dienen.
  Der Sprachumfang von Promela wird ausführlich in [Hol03] beschrieben. Es konnte zwar lediglich eine Teilsprache von Promela bei der Übersetzung in die Automatendarstellung berücksichtigt werden, jedoch wird an dieser Stelle auf deren Vorstellung verzichtet, da dieses den Rahmen des Anhangs sprengen würde.

Literaturverzeichnis

[AT85] H. Alaiwan and J. F. Toudic. Recherche des Semi-Flots, des Verroux et des Trappes dans les Réseaux de Petri. *TSI*, 4(1):103–112, 1985.

[Bau90] B. Baumgarten. *Petri-Netze. Grundlagen und Anwendungen.* BI-Wissenschaftsverlag, 1990.

[Ber85] G. Berthelot. Checking Properties of Nets Using Transformations. In G. Rozenberg, editor, *Advances in Petri Nets (covers the 6th European Workshop on Applications and Theory in Petri Nets)*, volume 222 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 1985.

[BF88] E. Best and C. Fernández. Nonsequential Processes - A Petri Net View. *EATCS Monographs in Theoretical Computer Science*, 13, 1988.

[BFF+95a] E. Best, H. Fleischhack, W. Fraczak, R. P. Hopkins, H. Klaudel, and E. Pelz. A Class of Composable High Level Petri Nets with an Application to the Semantics of B(PN)^2. In G. D. Michelis and M. Diaz, editors, *Proceedings of the 16th International Conference on Application and Theory of Petri Nets*, volume 935 of *Lecture Notes in Computer Science*, pages 103–120. Springer-Verlag, 1995.

[BFF+95b] E. Best, H. Fleischhack, W. Fraczak, R. P. Hopkins, H. Klaudel, and E. Pelz. An M-net Semantics of B(PN)^2. In J. Desel, editor, *Proceedings of the International Workshop on Structures in Concurrency Theory*, pages 85–100. Springer-Verlag, 1995.

[BH93] E. Best and R. P. Hopkins. B(PN)^2 - A Basic Petri Net Programming Notation. In A. Bode, M. Reeve, and G. Wolf, editors, *Proceedings of the 5th International Conference on Parallel Architectures and Languages Europe*, volume 694 of *Lecture Notes in Computer Science*, pages 379–390. Springer-Verlag, 1993.

[BMMR01] T. Ball, R. Majumdar, T. Millstein, and S. K. Rajamani. Automatic Predicate Abstraction of C Programs. In *Proceedings of the ACM SIGPLAN 2001 Conference on Programming Language, Design, and Implementation*, pages 203–213. ACM Press, 2001.

[Bry86] R. E. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[BS89] I. N. Bronstein and K. A. Semendjajew. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 24 edition, 1989.

[Büc62] J. R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Proceedings of the International Congress on Logic, Methodology and Philosophy Science 1960*, pages 1–12. Stanford University Press, 1962.

[CEP95] A. Cheng, J. Esparza, and J. Palsberg. Complexity Results for 1-safe Nets. *Theoretical Computer Science*, 147:117–136, 1995.

[CGL94] E. M. Clarke, O. Grumberg, and D. E. Long. Model Checking and Abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[CGP99] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT-Press, 1999.

[Cor94] J. C. Corbett. Evaluating Deadlock Detection Methods for Concurrent Software. In T. J. Ostrand, editor, *Proceedings of the 1994 International Symposium on Software Testing and Analysis*, Software Engineering Notes (Special Issue), pages 204–215. ACM-Press, 1994.

[Cpl97] CPLEX Optimization Inc. *Using the* CPLEX^{TM} *Callable Library and* CPLEX^{TM} *Mixed Integer Library*, 1997.

[CS90] J. M. Colom and M. Silva. Improving the Linearly Based Characterization of P/T Nets. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 483 of *Lecture Notes in Computer Science*, pages 113–145. Springer-Verlag, 1990.

[DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

[DE00] J. Desel and T. Erwin. Modeling, Simulation and Analysis of Business Processes. In W. M. P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 129–141. Springer-Verlag, 2000.

[Des98] J. Desel. Basic Linear Algebraic Techniques for Place or Transition Nets. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 257–308. Springer-Verlag, 1998.

[Des00] J. Desel. Validation of Process Models by Construction of Process Nets. In W. M. P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 110–128. Springer-Verlag, 2000.

[DGG97] D. Dams, R. Gerth, and O. Grumberg. Abstract Interpretation of Reactive Systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253–291, 1997.

[DGV99] M. Daniele, F. Giunchiglia, and M. Y. Vardi. Improved Automata Generation for Linear Temporal Logic. In N. Halbwachs and D. A. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 249–260. Springer-Verlag, 1999.

[DJLN03] J. Desel, G. Juhás, R. Lorenz, and C. Neumair. Modelling and Validation with VipTool. In W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, editors, *Proceedings of the International Conference on Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 380–389. Springer-Verlag, 2003.

[DJN04] J. Desel, G. Juhás, and C. Neumair. Finite Unfoldings of Unbounded Petri Nets. In J. Cortadedella and W. Reisig, editors, *Proceedings of the 25th International Conference on Applications and Theory of Petri Nets*, volume 3099 of *Lecture Notes in Computer Science*, pages 157–176. Springer-Verlag, 2004.

[DMN04] J. Desel, V. Milijic, and C. Neumair. Model Validation in Controller Design. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 467–495. Springer-Verlag, 2004.

[DNR96] J. Desel, K. P. Neuendorf, and M. D. Radola. Proving Nonreachability by Modulo-Invariants. *Theoretical Computer Science*, 153(1–2):49–64, 1996.

[ECS93] J. Ezpeleta, J. M. Couvreur, and M. Silva. A New Technique for Finding a Generating Family of Siphons, Traps and ST-Components. Application to Colored Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*, pages 126–147. Springer-Verlag, 1993.

[EH00a] J. Esparza and K. Heljanko. A new Unfolding Approach to LTL Model Checking. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, volume 1853 of *Lecture Notes in Computer Science*, pages 475–486. Springer-Verlag, 2000.

[EH00b] K. Etessami and G. J. Holzmann. Optimizing Büchi Automata. In C. Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 153–167. Springer-Verlag, 2000.

[EH01] J. Esparza and K. Heljanko. Implementing LTL Model Checking with Net Unfoldings. In M. B. Dwyer, editor, *Proceedings of the 8th International SPIN Workshop on Model Checking of Software*, volume 2057 of *Lecture Notes in Computer Science*, pages 37–56. Springer-Verlag, 2001.

[Eme90] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 995–1072. Elsevier, 1990.

[Eng91] J. Engelfriet. Branching Processes of Petri Nets. *Acta Informatica*, 28:575–591, 1991.

[ER99] J. Esparza and S. Römer. An Unfolding Algorithm for Synchronous Products of Transition Systems. In J. C. M. Baeten and S. Mauw, editors, *Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 2–20. Springer-Verlag, 1999.

[ERV96] J. Esparza, S. Römer, and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. In T. Margaria and B. Steffen, editors, *Proceedings of the 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 87–106. Springer-Verlag, 1996.

[ERV02] J. Esparza, S. Römer, and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design*, 20:285–310, 2002.

[ES00] J. Esparza and C. Schröter. Reachability Analysis Using Net Unfoldings. In H. D. Burkhard, L. Czaja, A. Skowron, and P. Starke, editors, *Workshop of Concurrency, Specification & Programming*, volume II of *Informatik-Bericht 140*, pages 255–270. Humboldt-Universität zu Berlin, 2000.

[ES01a] J. Esparza and C. Schröter. Net Reductions for LTL Model-Checking. In T. Margaria and T. Melham, editors, *Proceedings of the 11th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, volume 2144 of *Lecture Notes in Computer Science*, pages 310–324. Springer-Verlag, 2001.

[ES01b] J. Esparza and C. Schröter. Unfolding Based Algorithms for the Reachability Problem. *Fundamenta Informaticae*, 47(3,4):231–245, 2001.

[Esp93] J. Esparza. A Partial Order Approach to Model Checking. Habilitationsschrift, Universität Hildesheim, 1993.

[Esp94] J. Esparza. Model Checking Using Net Unfoldings. *Science of Computer Programming*, 23:151–195, 1994.

[FG96] H. Fleischhack and B. Grahlmann. A Petri Net Semantics for B(PN)^2 with Procedures which Allows Verification. Technical Report 21, Universität Hildesheim, 1996.

[FG97] H. Fleischhack and B. Grahlmann. A Petri Net Semantics for B(PN)^2 with Procedures. In *Proceedings of the 2nd International Workshop on Software Engineering for Parallel and Distributed Systems*, pages 15–27. IEEE Computer Society, 1997.

[Gab00] H. N. Gabow. Path-Based Depth-First Search for Strong and Biconnected Components. *Information Processing Letters*, 74(3–4):107–114, 2000.

[Gen87] H. Genrich. Predicate-Transition Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, 1987.

[GL88] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. A. Kowalski and K. Bowen, editors, *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080. MIT-Press, 1988.

[GO01] P. Gastin and D. Oddoux. Fast LTL to Büchi Automata Translation. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer-Verlag, 2001.

[God90] P. Godefroid. Using Partial Orders to Improve Automatic Verification Methods. In E. M. Clarke and R. P. Kurshan, editors, *Proceedings of the 2nd International Workshop on Computer Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 176–185. Springer-Verlag, 1990.

[GPVW96] R. Gerth, D. A. Peled, M. Y. Vardi, and P. Wolper. Simple On-the-fly Automatic Verification of Linear Temporal Logic. In P. Dembinski and M. Sredniawa, editors, *Proceedings of the 15th IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification 1995*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1996.

[Gra95] B. Grahlmann. Verifying Telecommunication Protocols with PEP. In *Proceedings of the 9th Symposium on Quality and Reliability in Electronics*, pages 251–256. Scientific Society for Telecommunications, 1995.

[GRT+95] B. Grahlmann, S. Römer, T. Thielke, B. Graves, M. Damm, R. Riemann, L. Jenner, S. Melzer, and A. Gronewold. PEP: Programming Environment Based on Petri Nets. Technical Report 14, Universität Hildesheim, 1995.

[GS97] S. Graf and H. Saïdi. Construction of Abstract State Graphs with PVS. In O. Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer-Verlag, 1997.

[Had90] S. Haddad. A Reduction Theory for Coloured Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1989 (covers the 9th European Workshop on Applications and Theory of Petri Nets)*, volume 424 of *Lecture Notes in Computer Science*, pages 209–235. Springer-Verlag, 1990.

[HD95] M. Heiner and P. Deusen. Petri Net Based Qualitative Analysis - A Case Study. Technical Report I-08/1995, Brandenburg Technische Universität Cottbus, 1995.

[Hel93] H. Hellwagner. Scalable Readers/Writers Synchronization on Shared-Memory Machines. Esprit P5404 (GP MIMD), Arbeitspapier, 1993.

[Hel99] K. Heljanko. Using Logic Programs with Stable Model Semantics to Solve Deadlock and Reachability Problems for 1–Safe Petri Nets. In R. Cleaveland, editor, *Proceedings of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579 of *Lecture Notes in Computer Science*, pages 240–254. Springer-Verlag, 1999.

[Hel02] K. Heljanko. *Combining Symbolic and Partial Order Methods for Model Checking 1-safe Petri Nets*. PhD thesis, Helsinki University of Technology, 2002.

[HKK02] K. Heljanko, V. Khomenko, and M. Koutny. Parallelisation of the Petri Net Unfolding Algorithm. In J. P. Katoen and P. Stevens, editors, *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *Lecture Notes in Computer Science*, pages 371–385. Springer-Verlag, 2002.

[Hol03] G. J. Holzmann. *The SPIN Model Checker*. Addison-Wesley, 2003.

[Jen91] K. Jensen. Coloured Petri Nets: A High-level Language for System Design and Analysis. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 342–416. Springer-Verlag, 1991.

[Jen92] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1: Basic Concepts*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1992.

[Jen94] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2: Analysis Methods.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1994.

[Jen97] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 3: Practical Use.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1997.

[Jen98] K. Jensen. An Introduction to the Practical Use of Coloured Petri Nets. In W. Reisig and G. Rozenberg, editors, *Advances in Petri Nets – Lectures on Petri Nets II: Applications*, volume 1492 of *Lecture Notes in Computer Science*, pages 237–292. Springer-Verlag, 1998.

[Kho03] V. Khomenko. *Model Checking Based on Prefixes of Petri Net Unfoldings.* PhD thesis, University of Newcastle upon Tyne, 2003.

[KK00] V. Khomenko and M. Koutny. LP Deadlock Checking Using Partial Order Dependencies. In C. Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 410–425. Springer-Verlag, 2000.

[KK01] V. Khomenko and M. Koutny. Towards an Efficient Algorithm for Unfolding Petri Nets. In K. G. Larsen and M. Nielsen, editors, *Proceedings of the 12th International Conference on Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 366–380. Springer-Verlag, 2001.

[KK03] V. Khomenko and M. Koutny. Branching Processes of High-Level Petri Nets. In H. Garavel and J. Hatcliff, editors, *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2619 of *Lecture Notes in Computer Science*, pages 458–472. Springer-Verlag, 2003.

[KKTT96] A. Kondratyev, M. Kishinevsky, A. Taubin, and S. Ten. A Structural Approach for the Analysis of Petri Nets by Reduced Unfoldings. In J. Billington and W. Reisig, editors, *Proceedings of the 17th International Conference on Applications and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pages 346–365. Springer-Verlag, 1996.

[KKV02] V. Khomenko, M. Koutny, and W. Vogler. Canonical Prefixes of Petri Net Unfoldings. In E. Brinksma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 582–595. Springer-Verlag, 2002.

[LL95] C. Lewerentz and T. Lindner. Formal Development of Reactive Systems - Case Study Production Cell. volume 891 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[Mar85] A. J. Martin. The Design of a Self-Timed Circuit of Distributed Mutual Exclusion. In H. Fuchs, editor, *Proceedings of the 1985 Chapel Hill Conference on VLSI*, pages 245–260. Computer Science Press, 1985.

[Mar86] A. J. Martin. Self-Timed FIFO: An Exercise in Compiling Programs Into VLSI Circuits. In D. Borrione, editor, *From HDL Descriptions to Guaranteed Correct Circuit Designs (Proceedings of the IFIP Wg10.2 Working Conference)*, pages 133–153. Elsevier Science & Technology Books, 1986.

[McM92] K. L. McMillan. Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits. In G. van Bochmann and D. K. Probst, editors, *Proceedings of the 4th International Conference on Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 164–174. Springer-Verlag, 1992.

[McM93] K. L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic Publishers, 1993.

[Mel98] S. Melzer. *Verifikation verteilter Systeme mittels linearer - und Constraint-Programmierung*. PhD thesis, Technische Universität München, 1998.

[Mil80] R. Milner. A Calculus of Communicating Systems. volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MR97] S. Melzer and S. Römer. Deadlock Checking Using Net Unfoldings. In O. Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 352–363. Springer-Verlag, 1997.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[Nie99] I. Niemelä. Logic Programming with Stable Model Semantics as a Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3,4):241–273, 1999.

[NPW80] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Events Structures and Domains. *Theoretical Computer Science*, 13(1):85–108, 1980.

[NS97] I. Niemelä and P. Simons. Smodels - An Implementation of the Stable Model and Well-Founded Semantics for Normal Logic Programs. In J. Dix, U. Furbach, and A. Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning*, volume 1265 of *Lecture Notes in Computer Science*, pages 420–429. Springer-Verlag, 1997.

[Pel93] D. A. Peled. All from One, One for All – On Model Checking Using Representatives. In C. Courcoubetis, editor, *Proceedings of the 5th International Conference on Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*, pages 409–423. Springer-Verlag, 1993.

[Pel94] D. A. Peled. Combining Partial Order Reductions with On-the-fly Model-Checking. In D. L. Dill, editor, *Proceedings of the 6th International Conference on Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 377–390. Springer-Verlag, 1994.

[Pet62] C. A. Petri. Kommunikation mit Automaten. Schriften des Instituts für instrumentelle Mathematik, Universität Bonn, 1962.

[Pnu77] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[PPP00] D. Poitrenaud and J. F. Pradat-Peyre. Pre- and Post-Agglomerations for LTL Model Checking. In M. Nielsen and D. Simpson, editors, *Proceedings of the 21st International Conference on Applications and Theory of Petri Nets*, volume 1825 of *Lecture Notes in Computer Science*, pages 387–408. Springer-Verlag, 2000.

[PRCB94] E. Pastor, O. Roig, J. Cortadella, and R. M. Badia. Petri Net Analysis Using Boolean Manipulation. In R. Valette, editor, *Proccedings of the 15th International Conference on Applications and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 416–435. Springer-Verlag, 1994.

[Ray86] M. Raynal. Algorithms For Mutual Exclusion. North Oxford Academic Publisher Ltd., 1986.

[RCP95] O. Roig, J. Cortadella, and E. Pastor. Verification of Asynchronous Circuits by BDD-based Model Checking of Petri Nets. In G. De Michelis and M. Diaz, editors, *Proceedings of the 16th International Conference on Applications and Theory of Petri Nets*, volume 935 of *Lecture Notes in Computer Science*, pages 374–391. Springer-Verlag, 1995.

[Rei85] W. Reisig. Petri Nets - An Introduction. volume 4 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag, 1985.

[Rei91] W. Reisig. Petri Nets and Algebraic Specifications. *Theoretical Computer Science*, 80(1):1–34, 1991.

[Ren88] J. Renegar. A Polynomial-Time Algorithm, Based on Newton's Method for Linear Programming. *Mathematical Programming*, 40:59–93, 1988.

[Röm00] S. Römer. *Theorie und Praxis der Netzentfaltungen als Grundlage für die Verifikation nebenläufiger Systeme.* PhD thesis, Techniche Universität München, 2000.

[RR98a] W. Reisig and G. Rozenberg. Lectures on Petri Nets I: Basic Models. In *Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[RR98b] W. Reisig and G. Rozenberg. Lectures on Petri Nets II: Applications. In *Advances in Petri Nets*, volume 1492 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[Sch92] U. Schöning. *Theoretische Informatik - kurz gefaßt.* BI-Wissenschaftsverlag, 1992.

[Sha81] M. Sharir. A Strong-Connectivity Algorithm and its Applications in Data Flow Analysis. *Computers and Mathematics with Applications*, 7(1):67–72, 1981.

[Sim00] P. Simons. *Extending and Implementing the Stable Model Semantics.* PhD thesis, Helsinki University of Technology, 2000.

[SK04] C. Schröter and V. Khomenko. Parallel LTL-X Model Checking of High-Level Petri Nets Based on Unfoldings. In R. Alur and D. A. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 109–121. Springer-Verlag, 2004.

[SSE03] C. Schröter, S. Schwoon, and J. Esparza. The Model-Checking Kit. In W. van der Aalst and E. Best, editors, *Proceedings of the 24th International Conference on Applications and Theory of Petri Nets*, volume 2679 of *Lecture Notes in Computer Science*, pages 463–472. Springer-Verlag, 2003.

[Sta90] P. Starke. *Analyse von Petri-Netz-Modellen.* B. G. Teubner, 1990.

[Tar72] R. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Tra93] H. Trauboth. *Software-Qualitätssicherung: Konstruktive und analytische Maßnahmen.* Oldenbourg-Verlag, 1993.

[Val90] A. Valmari. A Stubborn Attack on State Explosion. In E. M. Clarke and R. P. Kurshan, editors, *Proceedings of the 2nd International Workshop on Computer Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 156–165. Springer-Verlag, 1990.

[Var96] M. Y. Vardi. An Automata-Theoretic Approach to Linear Temporal Logic. In F. Moller and G. M. Birtwistle, editors, *Logics for Concurrency - Structure versus Automata (Proceedings of the 8th Banff Higher Order Workshop)*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, 1996.

[VW86a] M. Y. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *Proceedings of the 1st Symposium on Logics in Computer Science*, pages 332–344. IEEE Computer Society, 1986.

[VW86b] M. Y. Vardi and P. Wolper. Automata-Theoretic Techniques for Modal Logics of Programs. *Computer and System Sciences*, 32:183–221, 1986.

[Wal90] E. Wallmüller. *Software-Qualitätssicherung in der Praxis.* Hanser-Fachbuchverlag, 1990.

[Wal98] F. Wallner. Model Checking LTL Using Net Unfoldings. In A. J. Hu and M. Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 207–218. Springer-Verlag, 1998.