

Diplomarbeit

# Synthese offener Workflownetze aus Serviceautomaten

Patrick Köhnen

15. Januar 2008



Humboldt-Universität zu Berlin  
Mathematisch-Naturwissenschaftliche Fakultät II  
Institut für Informatik

Gutachter:  
Prof. Dr. Karsten Wolf  
Prof. Dr. Wolfgang Reisig



---

## Danksagung

Ich danke allen Personen, die mich in irgendeiner Form bei der Erstellung dieser Arbeit unterstützt haben. Für die Unterstützung bei Fragen den inhaltlichen Teil betreffend danke ich allen Mitarbeitern des Lehrstuhls Theorie der Programmierung, einer Gruppe von sehr fähigen, aber vor allem sehr freundlichen Menschen. Ganz besonders hervorzuheben ist hierbei mein Betreuer Peter Massuthe, der sich immer die Zeit nahm, um meine Fragen, auch mal in längeren und konstruktiven Diskussionen, zu klären, und immer mit Rat und Tat zur Seite stand.

Meinen Freunden und meiner Familie, ganz besonders meinen Eltern, danke ich für die Unterstützung während meines Studiums. Zudem danke ich für das Verständnis, besonders in den letzten Wochen der Erstellung dieser Arbeit, aus Zeitgründen auf mich verzichten zu müssen.

Vielen herzlichen Dank!



---

## Zusammenfassung

Das Paradigma der service-orientierten Architektur beschreibt eine Kapselung einzelner Funktionalitäten von Softwaresystemen in Services. Ein Service besitzt somit eine bestimmte Funktionalität und zudem eine definierte Schnittstelle. Über diese Schnittstelle kann die Funktionalität des Services genutzt werden. Durch diese Trennung von Funktionalität und Schnittstelle ist ein Service unabhängig von der Plattform und der verwendeten Programmiersprache. Änderungen und Erweiterungen service-basierter Softwaresysteme können durch Anpassungen oder das Austauschen des betreffenden Services erreicht werden und sind daher im Vergleich zu anderen Systemen einfacher, schneller und mit einem geringeren Risiko umsetzbar.

Web Services sind eine spezielle und weit verbreitete Form von Services. Ein Web Service ist ein eigenständiges Softwaremodul, dessen Funktionalität über das Internet angeboten wird. Mit der Web Service Business Process Execution Language (WS-BPEL) kann ein Web Service definiert werden, der den Geschäftsprozess eines Unternehmens abbildet. WS-BPEL besitzt hierfür Kontrollstrukturen und Funktionalität zur Behandlung von Fehlern und Ausnahmen.

WS-BPEL besitzt keine formale Semantik und kann daher nicht formal analysiert werden. Für eine formale Analyse kann ein WS-BPEL Prozess in ein offenes Workflownetz (oWFN) übersetzt werden. Existiert für ein oWFN ein Partner, der verklemmungsfrei mit diesem oWFN interagiert, kann ein Serviceautomat (SVA) berechnet werden, der das Verhalten dieses Partners beschreibt. Bisher war es möglich, ein oWFN nach WS-BPEL zu übersetzen, aber nicht, ein oWFN aus einem SVA zu synthetisieren. Ein berechneter Partner für einen WS-BPEL Prozess in Form eines SVAs konnte somit bisher nicht zurück in WS-BPEL übersetzt werden. Das Ziel dieser Arbeit ist es, die Synthese eines oWFNS aus einem SVA zu definieren und somit den Vorgang der Berechnung eines Partners für einen WS-BPEL Prozess zu vervollständigen.



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Hintergrund</b>	<b>5</b>
2.1	Web Services Business Process Execution Language . . . . .	5
2.2	Offene Workflownetze . . . . .	6
2.2.1	Petrinetze . . . . .	7
2.2.2	Interaktion offener Workflownetze . . . . .	9
2.3	Verhaltensbeschreibung offener Workflownetze mit Serviceautomaten . .	14
2.3.1	Serviceautomat . . . . .	14
2.3.2	Interaktionsgraph . . . . .	16
2.4	Synthese von Petrinetzen aus Serviceautomaten . . . . .	17
2.4.1	Regionentheorie . . . . .	18
2.4.2	Elementarer Serviceautomat . . . . .	20
2.4.3	Synthetisiertes Petrinetz . . . . .	21
<b>3</b>	<b>Synthese offener Workflownetze aus Serviceautomaten</b>	<b>25</b>
3.1	Transformation von Serviceautomaten in elementare Serviceautomaten .	25
3.2	Synthetisiertes offenes Workflownetz . . . . .	26
3.3	Notwendige Eigenschaften von oWFNs zur Übersetzung in WS-BPEL . .	27
3.3.1	Empfangskonflikt . . . . .	28
3.3.2	Soundness . . . . .	31
3.3.3	Zyklen in offenen Workflownetzen . . . . .	36
3.4	Verhaltensanalyse transformierter Serviceautomaten . . . . .	38
<b>4</b>	<b>Implementierung der Synthese offener Workflownetze</b>	<b>41</b>
4.1	Das Tool SVA2oWFN . . . . .	42
4.2	Ein Beispiel für die Synthese eines offenen Workflownetzes . . . . .	43
<b>5</b>	<b>Zusammenfassung und Ausblick</b>	<b>47</b>
5.1	Zusammenfassung . . . . .	47
5.2	Ausblick . . . . .	48
	<b>Literaturverzeichnis</b>	<b>49</b>



# Abbildungsverzeichnis

1.1	Schematische Darstellung der service-orientierten Architektur. . . . .	2
2.1	WS-BPEL Prozess 'Stromanbieter'. . . . .	6
2.2	Ein Petrinetz mit der Anfangsmarkierung $m_0 = [p0]$ . . . . .	8
2.3	Das Petrinetz aus Abbildung 2.2 nach dem aufeinanderfolgenden Schalten der Transitionen mit der Beschriftung <code>anmelden</code> (a) und der Beschriftung <code>adresse_neu</code> (b). . . . .	10
2.4	Zwei oWFNs $N$ und $M$ , wobei $N$ dem WS-BPEL Prozess aus Abbildung 2.1 entspricht. . . . .	12
2.5	Die Komposition der beiden Partner oWFNs $N$ und $M$ aus Abbildung 2.4. . . . .	13
2.6	Ein SVA $A$ , der das Verhalten des oWFNs $M$ aus Abbildung 2.4(b) beschreibt. . . . .	15
2.7	Ein nicht vollständig terminierungsfähiger Serviceautomat. . . . .	17
2.8	Interaktionsgraph des oWFNs $N$ aus Abbildung 2.4(a). . . . .	18
2.9	Ein Serviceautomat $A$ . . . . .	19
2.10	Ein elementarer SVA $A'$ , der aus dem nicht elementaren SVA $A$ in Abbildung 2.9, durch das Aufsplitten der Beschriftung <code>!anmelden</code> , entstanden ist. . . . .	21
2.11	Synthetisiertes Petrinetz $N(A')$ zum ESVA aus Abbildung 2.10. . . . .	22
2.12	Erreichbarkeitsgraph $EG(N(A'))$ des SPNs $N(A')$ aus Abbildung 2.11. . . . .	23
2.13	Bisimulation zwischen dem ESVA $A'$ aus Abbildung 2.10 und dem Erreichbarkeitsgraph $EG(N(A'))$ aus Abbildung 2.12. . . . .	24
3.1	Transformation des SVAs $A$ in den SVA $A'$ ohne Selbstschleifen. . . . .	26
3.2	Das SPN $N(A')$ aus Abbildung 2.11 zu dem oWFN $N'(A')$ erweitert. . . . .	28
3.3	Ein ESVA $A$ und das daraus synthetisierte oWFN $N(A)$ mit einem Empfangskonflikt. . . . .	28
3.4	Ein aus dem ESVA $A'$ (a) synthetisiertes oWFN $N'(A')$ (b) ohne Empfangskonflikt. . . . .	29
3.5	Ein ESVA $A$ mit Empfangskonflikt. . . . .	30
3.6	Ein aus dem ESVA $A'$ synthetisiertes oWFN kann keinen Empfangskonflikt besitzen. . . . .	31
3.7	Das aus dem ESVA $A$ (a) synthetisierte oWFN $N(A)$ ist nicht sound. . . . .	33
3.8	Das aus dem ESVA $A'$ (a) synthetisierte oWFN $N'(A')$ ist sound. . . . .	36
3.9	Vermeidung von unerlaubten Zyklen. . . . .	37

4.1	Interaktionsgraph $I$ des oWFNs $N$ aus Abbildung 2.4(a). . . . .	41
4.2	Der dem IG aus Abbildung 4.1 entsprechende SVA $A$ . . . . .	44
4.3	Der aus dem SVA $A$ in Abbildung 4.2 entstandene SVA $A'$ . . . . .	44
4.4	Das aus dem SVA $A$ in Abbildung 4.3 synthetisierte oWFN $N(A')$ . . . . .	45

# 1 Einleitung

Traditionelle informations-technologische Systemlandschaften in Unternehmen bestehen aus monolithischen Systemen. Solche Systeme bestehen aus Komponenten, die sehr eng miteinander verknüpft sind. Änderungen einer einzelnen Komponente können daher häufig zu Problemen und Fehlern bei der Interaktion mit anderen Komponenten führen und beinhalten somit ein zeitliches und finanzielles Risiko.

Die service-orientierte Architektur (SOA) [Got00] ist ein Paradigma, das die oben beschriebenen Probleme minimiert. Bei der SOA wird die Funktionalität eines Systems in einzelne Services gekapselt. Ein Service ist ein Softwaremodul zur Ausführung einer festgelegten Funktionalität. Die Funktionalität eines Services kann über eine definierte Schnittstelle genutzt werden. Nutzer eines Services kann sowohl ein Anwender, als auch ein anderer Service sein.

Ein großer Vorteil der SOA, gegenüber monolithischen Systemen, ist der modulare Aufbau von Systemen durch Services. Muss ein Vorgang in einem Geschäftsprozess geändert werden, reicht es oftmals aus, den entsprechenden Service anzupassen bzw. auszutauschen. Zudem ist eine dynamische Bindung von Services während der Laufzeit möglich. Da die Kommunikation mit einem Service lediglich über dessen Schnittstelle abläuft, ist die Implementation eines Services unabhängig von der Programmiersprache und der Plattform des Services.

Innerhalb der SOA wird zwischen den drei verschiedenen Rollen Serviceanbieter, Servicebroker und Servicenutzer unterschieden. Diese drei Rollen sind in Abbildung 1.1 dargestellt. Ein Serviceanbieter macht seinen Service sowie die Art und Weise, wie dieser Service genutzt werden kann, bei einem Servicebroker bekannt. Ein Servicenutzer kann bei dem Servicebroker nach einem benötigten Service suchen. Findet der Servicenutzer einen passenden Service, erhält er vom Servicebroker alle notwendigen Informationen, um diesen Service zu nutzen.

Eine spezielle und weit verbreitete Form von Services sind Web Services [Got00]. Ein Web Service ist ein eigenständiges Softwaremodul, dessen Funktionalität über ein Netzwerk (im Allgemeinen das Internet) angeboten wird.

Mit Hilfe der Web Service Business Process Execution Language (WS-BPEL) [AAA+07] können verschiedene Web Services für die Abbildung eines Geschäftsprozesses genutzt werden. WS-BPEL verfügt hierfür über Kontrollstrukturen (if-then-else, while, etc.) und

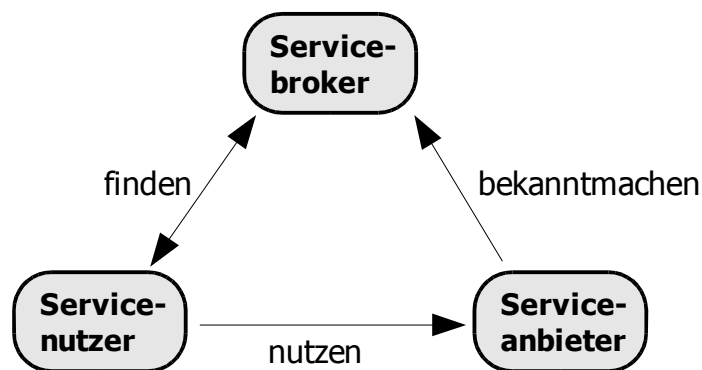


Abbildung 1.1: Schematische Darstellung der service-orientierten Architektur.

zusätzliche Funktionalität zur Behandlung von Fehlern und Ausnahmen. Eine formale Semantik besitzt WS-BPEL nicht, kann jedoch in ein Modell mit formaler Semantik übersetzt werden [LMSW06]. Bei diesem Modell handelt es sich um offene Workflownetze (oWFN) [MRS05], die eine spezielle Form von Petrinetzen [Rei85] darstellen.

Für einen, in ein oWFN übersetzten, WS-BPEL Prozess kann berechnet werden, ob andere Prozesse existieren, die mit diesem Prozess verklebungsfrei interagieren können [Sch05]. Existiert für ein oWFN ein solcher Partner-Prozess, kann ein Serviceautomat (SVA) [MS05] berechnet werden, der das Verhalten dieses Partner-Prozesses beschreibt [Wei06].

Es existiert eine formale Übersetzung von oWFNs nach WS-BPEL [Kle07], aber keine formale Beschreibung der Synthese von oWFNs aus SVAs. Ein berechneter Partner eines WS-BPEL Prozesses, in Form eines SVAs, konnte bisher nicht formal nach WS-BPEL übersetzt werden. Das erste Ziel dieser Arbeit ist es, die Synthese von oWFNs aus SVAs formal zu definieren und somit die formal definierte Berechnung eines Partners für einen WS-BPEL Prozesses, in Form eines anderen WS-BPEL Prozesses, zu vervollständigen.

Für die Übersetzung von oWFNs in WS-BPEL werden in [Kle07] bestimmte Eigenschaften des oWFNs verlangt. Die Zusicherung dieser Eigenschaften, bereits bei der Synthese eines oWFNs aus einem SVA zu gewährleisten, ist das zweite Ziel dieser Arbeit.

Der Aufbau dieser Arbeit gliedert sich in vier weitere Kapitel. Das zweite Kapitel beschreibt die für diese Arbeit notwendigen Grundlagen. Die ersten beiden Abschnitte behandeln die Sprache WS-BPEL und oWFNs. Im dritten Abschnitt wird die Verhaltensbeschreibung von oWFNs mit SVAs beschrieben. Die Synthese von Petrinetzen aus Serviceautomaten behandelt der dritte Abschnitt.

Das dritte Kapitel widmet sich der Synthese offener Workflownetze aus SVAs. Ein SVA muss für die Synthese in ein oWFN bestimmte Eigenschaften besitzen. Sind diese Eigen-

---

schaften nicht vorhanden, muss der SVA zunächst transformiert werden. Diesen Vorgang beschreibt der erste Abschnitt dieses Kapitels. Im zweiten Abschnitt werden synthetisierte oWFNs beschrieben und definiert. Der dritte Abschnitt behandelt die notwendigen Eigenschaften für eine Übersetzung von oWFNs in WS-BPEL und zeigt wie diese Eigenschaften zugesichert werden können. Im vierten Abschnitt wird analysiert, ob das Verhalten eines SVAs durch Transformationen verändert wird.

Im Rahmen dieser Arbeit wurde die Berechnung synthetisierter oWFNs aus SVAs in dem Tool SVA2oWFN umgesetzt. Die Beschreibung und Arbeitsweise dieses Tools erfolgt anhand eines Beispiels im vierten Kapitel.

Das fünfte Kapitel liefert eine Zusammenfassung dieser Arbeit, beschreibt die offenen Punkte und analysiert in einem Ausblick die Möglichkeiten, die aus den Ergebnissen dieser Arbeit resultieren.



## 2 Hintergrund

In diesem Kapitel werden die grundlegenden Begriffe und Definitionen erläutert und definiert, die in den nachfolgenden Kapiteln verwendet werden. Abschnitt 2.1 behandelt mit der Web Services Business Process Execution Language eine Sprache zur Beschreibung von Geschäftsprozessen. Um einen Geschäftsprozess zu analysieren ist es sinnvoll ihn formal zu beschreiben. Hierfür können offene Workflownetze verwendet werden, die in Abschnitt 2.2 eingeführt werden. Das Verhalten eines offenen Workflownetzes kann mit einem Serviceautomat dargestellt werden. Serviceautomaten sind das Thema in Abschnitt 2.3. Um aus einem Serviceautomaten ein offenes Workflownetz mit äquivalenten Verhalten zu erstellen, wird zunächst ein synthetisiertes Petrinetz erstellt. Dieser Vorgang basiert auf der Regionentheorie und wird in Abschnitt 2.4 beschrieben.

### 2.1 Web Services Business Process Execution Language

Die *Web Services Business Process Execution Language* (WS-BPEL) [AAA<sup>+</sup>07] ist eine Sprache zur Beschreibung von Geschäftsprozessen, die auf Web Services basieren. Zur Beschreibung der Geschäftsprozesse werden in WS-BPEL Aktivitäten verwendet. Diese unterscheiden sich in Basisaktivitäten und strukturierte Aktivitäten. Basisaktivitäten dienen zur Manipulation von Daten (*assign*), Kommunikation mit einem Partner (*invoke*, *receive*, *reply*), dem Warten (*wait*), dem nichts tun (*empty*), oder dem signalisieren von Fehlern (*throw*).

Die strukturierten Aktivitäten beinhalten Basisaktivitäten und ordnen deren Ausführungsreihenfolge. Eine strukturierte Aktivität kann auch Teil einer anderen strukturierten Aktivität sein und ein Geschäftsprozess kann daher geschachtelt definiert werden. Aktivitäten innerhalb einer strukturierten Aktivität werden sequentiell (*sequence*), parallel (*flow*), wiederholt (*while*), oder abhängig von internen Bedingungen (*switch*) und externen Bedingungen (*pick*) ausgeführt. Ein *scope* ist eine Aktivität, die das Verhalten einer anderen Aktivität kapselt. Jeder Aktivität können hierfür Fehlerhandler (*fault handler*), Ereignishandler (*event handler*) und Kompensationshandler (*compensation handler*) zugeteilt werden. Mit einem Fehlerhandler können Fehler, die in Aktivitäten innerhalb des Fehlerhandlers auftreten, gefangen und behandelt werden. Ein

Ereignishandler dient zur Behandlung eingehender Nachrichten und zeitbasierter Ereignisse. Kompensationshandler haben den Zweck, bestimmtes Verhalten rückgängig zu machen.

Abbildung 2.1 zeigt den WS-BPEL Prozess eines Stromanbieters. Der Prozess beschreibt, wie ein Kunde des Stromanbieters entweder seine Adresse oder seinen Zählerstand aktualisieren kann. Mit dem Empfang der Anmeldung eines Kunden beginnt der Prozess. Im nächsten Schritt erwartet der Prozess entweder eine neue Adresse, oder den aktuellen Zählerstand des Kunden. Der Prozess bestätigt den Erhalt der entsprechenden Daten und endet mit dem Empfangen einer Abmeldung.

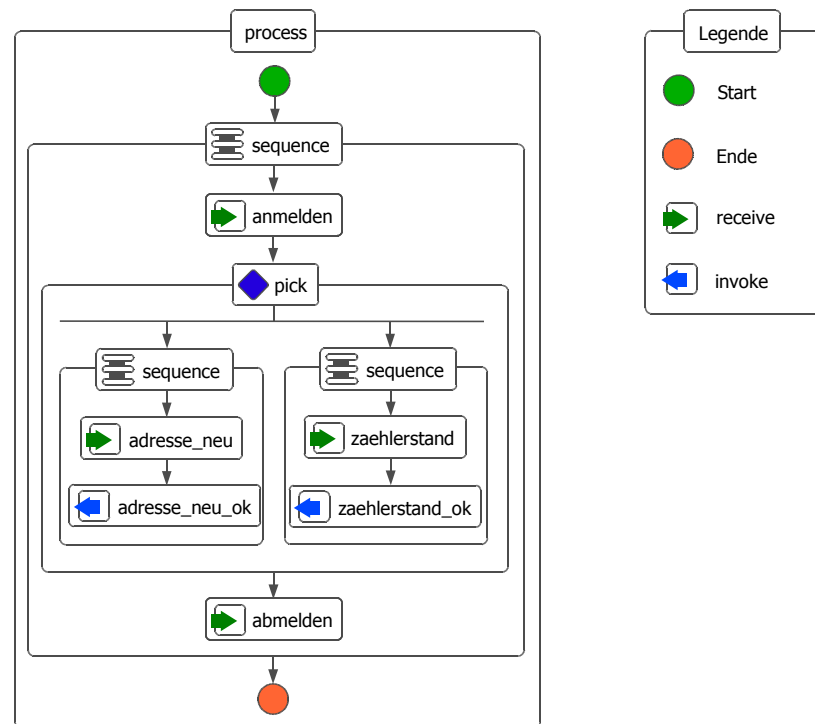


Abbildung 2.1: WS-BPEL Prozess 'Stromanbieter'.

## 2.2 Offene Workflownetze

Ein mit WS-BPEL beschriebener Prozess besitzt keine formale Semantik. Für die formale Analyse eines solchen Prozesses ist es daher notwendig ihn in ein Modell mit einer formalen Semantik zu überführen. Ein solches Modell stellen die offenen Workflownetze (oWFNs) [MRS05] dar. OWFNs beruhen auf Petrinetzen und sind eine erweiterte Variante von Workflownetzen [Aal97]. Die Übersetzung von WS-BPEL Prozessen in oWFNs wurde in [LMSW06] beschrieben und beruht auf der Semantik aus [Sta05].

### 2.2.1 Petrinetze

Petrinetze sind ein mathematischer Formalismus zur Modellierung von Systemen in Form eines gerichteten Graphen. Die Knotenmenge besteht aus den beiden disjunkten Mengen der Plätze (graphisch dargestellt als Kreise) und der Transitionen (graphisch dargestellt als Rechtecke). Eine Kante führt immer von einem Platz zu einer Transition, oder von einer Transition zu einem Platz. Die folgenden Definitionen beziehen sich auf einfache low-level Petrinetze entsprechend [Rei85].

**Definition 2.2.1 (Petrinetz)**

$N = (P, T, F, m_0)$  ist ein Petrinetz, wenn gilt:

- $P$  ist eine endliche Menge von Plätzen,
- $T$  ist eine endliche Menge von Transitionen ( $T \cap P = \emptyset$ ),
- $F$  ist eine Menge von Kanten (Flussrelation)  $F \subseteq (P \times T) \cup (T \times P)$ ,
- $m_0$  ist die Anfangsmarkierung.

Für einige Definitionen dieser Arbeit benötigt jede Transition eine Beschriftung. Eine Transitionsbeschriftung ist eine Funktion, die jeder Transition eine Beschriftung zuordnet.

**Definition 2.2.2 (Transitionsbeschriftung für ein Petrinetz)**

Sei  $N = (P, T, F, m_0)$  ein Petrinetz und  $L$  eine Menge von Beschriftungen. Die Funktion  $\lambda : T \rightarrow L$  ist eine Transitionsbeschriftung für  $N$ .

Abbildung 2.2 zeigt ein Petrinetz, wobei die Beschriftungen der Transitionen den Nachrichten des WS-BPEL Prozesses aus Abbildung 2.1 entsprechen.

Eine Markierung  $m$  ist eine Abbildung, die jedem Platz eine Anzahl von Marken (graphisch dargestellt als schwarze Token) zuordnet.

**Definition 2.2.3 (Markierung)**

Sei  $N = (P, T, F, m_0)$  ein Petrinetz. Eine Funktion  $m : P \rightarrow \mathbb{N}$  heißt Markierung von  $N$ . Für zwei Markierungen  $m$  und  $m'$  ist  $m \geq m'$ , wenn für alle Plätze  $p \in P$  gilt  $m(p) \geq m'(p)$ .

Dargestellt wird eine Markierung in dieser Arbeit als Multimenge, also eine Menge bei der Elemente mehr als einmal enthalten sein können. Für jede Marke, die ein Platz

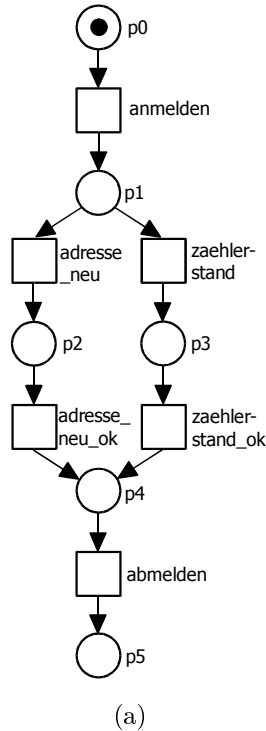


Abbildung 2.2: Ein Petrinetz mit der Anfangsmarkierung  $m_0 = [p_0]$ .

bei einer Markierung besitzt, ist der Platz jeweils einmal als Element in der Multimenge vorhanden. Das Petrinetz in Abbildung 2.2 besitzt bei der Anfangsmarkierung  $m_0 = [p_0]$  eine Marke auf dem Platz  $p_0$  und keine Marke auf allen anderen Plätzen. Angenommen die Anfangsmarkierung wäre  $m_0 = [p_0, p_0, p_2]$ , dann befänden sich auf dem Platz  $p_0$  zwei Marken und eine Marke auf dem Platz  $p_2$ .

Zur Beschreibung der Dynamik in Petrinetzen werden im Folgenden die Begriffe Vorbereich und Nachbereich von Knoten benötigt. Der Vorbereich eines Knotens besteht aus allen Knoten, von denen eine Kante zu diesem Knoten führt. Alle Knoten, zu denen von einem Knoten eine Kante führt, bilden den Nachbereich dieses Knotens.

**Definition 2.2.4 (Vor- und Nachbereich eines Knotens)**

Sei  $x \in P \cup T$  ein Knoten eines Petrinetzes  $N = (P, T, F, m_0)$ .

1. Die Menge  $\bullet x = \{y \mid (y, x) \in F\}$  wird als Vorbereich von  $x$ ,
2. die Menge  $x \bullet = \{y \mid (x, y) \in F\}$  wird als Nachbereich von  $x$  bezeichnet.

Der Vorbereich des Platzes  $p_1$  in Abbildung 2.2 besteht aus der Transition mit der Beschriftung *anmelden*. Der Nachbereich dieses Platzes wird von den zwei Transitionen mit den Beschriftungen *adresse\_neu* und *zaehlerstand* gebildet.

Eine Transition wird aktiviert genannt, wenn sich auf jedem Platz im Vorbereich der Transition mindestens eine Marke befindet.

**Definition 2.2.5 (Aktivierte Transition)**

Sei  $t \in T$  eine Transition eines Petrinetzes  $N = (P, T, F, m_0)$ . Die Transition  $t$  ist bei einer Markierung  $m$  aktiviert, wenn für alle Plätze  $p \in \bullet t$  gilt:  $m(p) \geq 1$ .

In Abbildung 2.2 ist die Transition mit der Beschriftung `anmelden` aktiviert, da der Platz `p0` eine Marke besitzt und sich keine weiteren Plätze im Vorbereich der Transition `anmelden` befinden. Alle anderen Transitionen des Netzes sind nicht aktiviert.

Eine aktivierte Transition eines Petrinetzes kann schalten. Hierbei wird von jedem Platz aus dem Vorbereich der Transition eine Marke entfernt und auf jeden Platz aus dem Nachbereich eine Marke hinzugefügt. Dargestellt wird das Schalten einer Transition mit  $m \xrightarrow{t} m'$ . Eine (möglicherweise leere) Schaltsequenz wird mit  $m \xrightarrow{*} m'$  dargestellt.

**Definition 2.2.6 (Schalten einer Transition)**

Sei  $N = (P, T, F, m_0)$  ein Petrinetz und  $t \in T$  eine Transition, die bei der Markierung  $m$  aktiviert ist. Die Transition  $t$  kann schalten und hierdurch die neue Markierung  $m'$  herbeiführen. Für alle  $p \in P$  muss gelten:

$$m'(p) = m(p) + \begin{cases} -1, & \text{falls } (p, t) \in F \text{ und } (t, p) \notin F, \\ 1, & \text{falls } (t, p) \in F \text{ und } (p, t) \notin F, \\ 0, & \text{sonst.} \end{cases}$$

Abbildung 2.3(a) zeigt das Petrinetz aus Abbildung 2.2 nach dem Schalten der Transition `anmelden` mit der Markierung  $m_1 = [p1]$ . In Abbildung 2.3(a) sind die Transitionen mit der Beschriftung `adresse_neu` und `zaehlerstand` aktiviert. Schalten der Transition mit der Beschriftung `adresse_neu` führt zu der Markierung  $m_2 = [p2]$  in Abbildung 2.3(b).

## 2.2.2 Interaktion offener Workflownetze

Workflownetze [Aal97] sind spezielle Petrinetze, die einen Arbeitsablauf beschreiben. Bei oWFNs handelt es sich um modifizierte Workflownetze. Der entscheidende Unterschied besteht darin, dass oWFNs explizite Plätze zur Kommunikation mit einem Partner besitzen. Des weiteren haben Workflownetze genau einen Anfangsplatz und genau einen Endplatz. Ein oWFN besitzt dagegen eine Anfangsmarkierung, die aus mehreren Plätzen bestehen kann und eine Menge von Endmarkierungen. Die folgenden Definitionen orientieren sich an [MRS05].

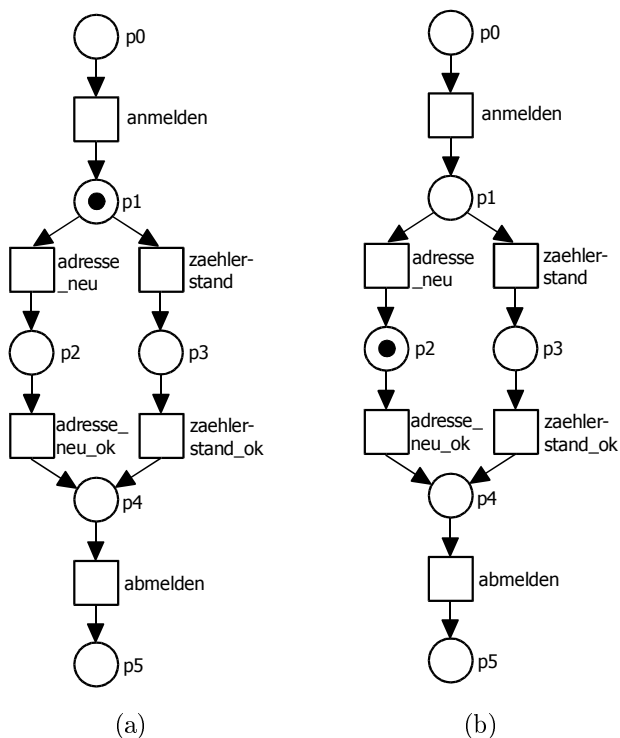


Abbildung 2.3: Das Petrinetz aus Abbildung 2.2 nach dem aufeinanderfolgenden Schalten der Transitionen mit der Beschriftung `anmelden` (a) und der Beschriftung `adresse_neu` (b).

**Definition 2.2.7 (Offenes Workflownetz)**

Ein offenes Workflownetz  $N = (P, P_i, P_o, T, F, m_0, \Omega)$  ist ein Petrinetz und besteht somit aus inneren Plätzen  $P$ , Transitionen  $T$ , einer Kantenmenge  $F$  und einer Anfangsmarkierung  $m_0$ . Zusätzlich besitzt ein offenes Workflownetz:

- Zwei disjunkte Mengen  $P_i, P_o \subseteq P$ , genannt Inputplätze und Outputplätze, so dass gilt:  $F \cap (T \times P_i) = F \cap (P_o \times T) = \emptyset$  und  $m_0(p) = 0$ , für alle  $p \in P_i \cup P_o$ ,
- eine Menge  $\Omega$  von Endmarkierungen, wobei  $m_f(p) = 0$ , für alle  $m_f \in \Omega, p \in P_i \cup P_o$ . In einer Markierung  $m_f \in \Omega$  darf keine Transition aktiviert sein.

Abbildung 2.4(a) zeigt ein oWFN  $N$ , dessen Verhalten dem WS-BPEL Prozess aus Abbildung 2.1 entspricht. Es besteht aus dem Petrinetz aus Abbildung 2.2 mit zusätzlichen Inputplätzen für die eingehenden Nachrichten und zusätzlichen Outputplätzen für ausgehende Nachrichten. Die Anfangsmarkierung ist  $m_0 = [p0]$ .  $N$  besitzt mit  $m_f = [p5]$  genau eine Endmarkierung.

Die Input- und Outputplätze eines oWFNs dienen zur Kommunikation mit anderen oWFNs. Zwei oWFNs  $N$  und  $M$  sind Partner oWFNs, wenn die Inputplätze von  $N$  genau den Outputplätzen von  $M$  entsprechen und die Outputplätze von  $M$  genau den Inputplätzen von  $N$  entsprechen. Alle Elemente von  $N$  und  $M$  außer den Input- und Outputplätzen müssen disjunkt sein.

**Definition 2.2.8 (Partner oWFNs)**

Seien  $N = (P_N, P_{i_N}, P_{o_N}, T_N, F_N, m_{0_N}, \Omega_N)$  und  $M = (P_M, P_{i_M}, P_{o_M}, T_M, F_M, m_{0_M}, \Omega_M)$  zwei oWFNs.  $N$  und  $M$  sind Partner oWFNs, wenn gilt:

- $P_{i_N} = P_{o_M}$  und  $P_{o_N} = P_{i_M}$ ,
- $(P_N \setminus (P_{i_N} \cup P_{o_N})) \cap (P_M \setminus (P_{o_M} \cup P_{i_M})) = \emptyset$ ,
- $T_N \cap T_M = \emptyset$ .

Abbildung 2.4(b) zeigt ein Partner oWFN  $M$  für das oWFN  $N$  aus Abbildung 2.4(a). Die Anfangsmarkierung von  $M$  ist  $m_{0_M} = [p6]$ .  $M$  besitzt mit  $m_{f_M} = [p11]$  genau eine Endmarkierung. Die Outputplätze von  $M$  anmelden, zaehlerstand, adresse\_neu und abmelden sind die Inputplätze von  $N$ . Die Inputplätze zaehlerstand\_ok und adresse\_neu\_ok von  $M$  sind die Outputplätze von  $N$ .

Zwei Partner oWFNs können zu einem neuen oWFN kombiniert werden. Hierfür wird die folgende Definition zur Komposition von Markierungen benötigt.

**Definition 2.2.9 (Komposition von Markierungen)**

Seien  $N = (P_N, P_{i_N}, P_{o_N}, T_N, F_N, m_{0_N}, \Omega_N)$  und  $M = (P_M, P_{i_M}, P_{o_M}, T_M, F_M, m_{0_M}, \Omega_M)$  zwei Partner oWFNs. Die Komposition  $m_N \oplus m_M : P_N \cup P_M \rightarrow \mathbb{N}$  zweier Markierungen von  $N$  und  $M$  ist definiert als:

$$(m_N \oplus m_M)(p) = \begin{cases} m_N(p), & \text{falls } p \in P_N \setminus (P_{i_N} \cup P_{o_N}), \\ m_M(p), & \text{falls } p \in P_M \setminus (P_{i_M} \cup P_{o_M}), \\ 0, & \text{sonst.} \end{cases}$$

Sind zwei oWFNs  $N$  und  $M$  Partner oWFNs, können  $N$  und  $M$  zu einem neuen oWFN kombiniert werden. Die Input- und Outputplätze von  $N$  werden zusammen mit den entsprechenden Output- und Inputplätzen von  $M$  zu inneren Plätzen.

**Definition 2.2.10 (Komposition von oWFNs)**

Seien  $N = (P_N, P_{i_N}, P_{o_N}, T_N, F_N, m_{0_N}, \Omega_N)$  und  $M = (P_M, P_{i_M}, P_{o_M}, T_M, F_M, m_{0_M}, \Omega_M)$  zwei Partner oWFNs. Die Komposition  $N \oplus M$  dieser oWFNs ist definiert als:

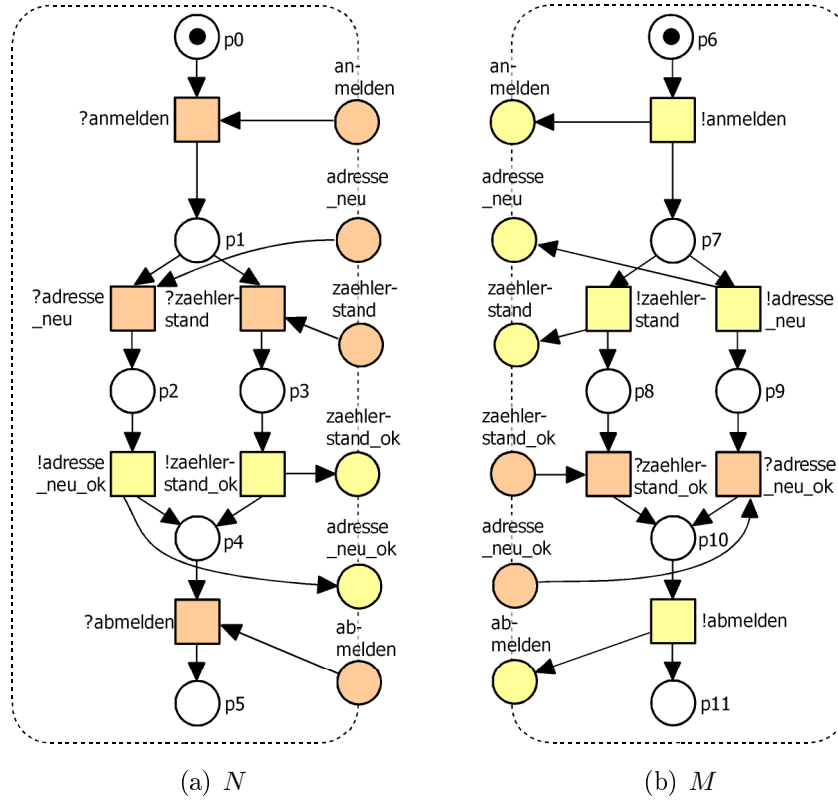


Abbildung 2.4: Zwei oWFNs  $N$  und  $M$ , wobei  $N$  dem WS-BPEL Prozess aus Abbildung 2.1 entspricht.

- $P_{N \oplus M} = P_N \cup P_M$ ,
- $T_{N \oplus M} = T_N \cup T_M$ ,
- $F_{N \oplus M} = F_N \cup F_M$ ,
- $m_{0_{N \oplus M}} = m_{0_N} \oplus m_{0_M}$ ,
- $P_{i_{N \oplus M}} = P_{o_{N \oplus M}} = \emptyset$ ,
- $\Omega_{N \oplus M} = \{m_N \oplus m_M \mid m_N \in \Omega_N, m_M \in \Omega_M\}$ .

Abbildung 2.5 zeigt die Komposition der beiden oWFNs aus Abbildung 2.4(a) und 2.4(b). Die Anfangsmarkierung der Komposition ist  $m_{N \oplus M} = [p0, p6]$ . Es existiert genau eine Endmarkierung  $m_{f_{N \oplus M}} = [p5, p11]$ .

Ein oWFN heißt schwach terminierend, wenn von jeder Markierung, die von der Anfangsmarkierung erreichbar ist, eine Endmarkierung erreichbar ist.

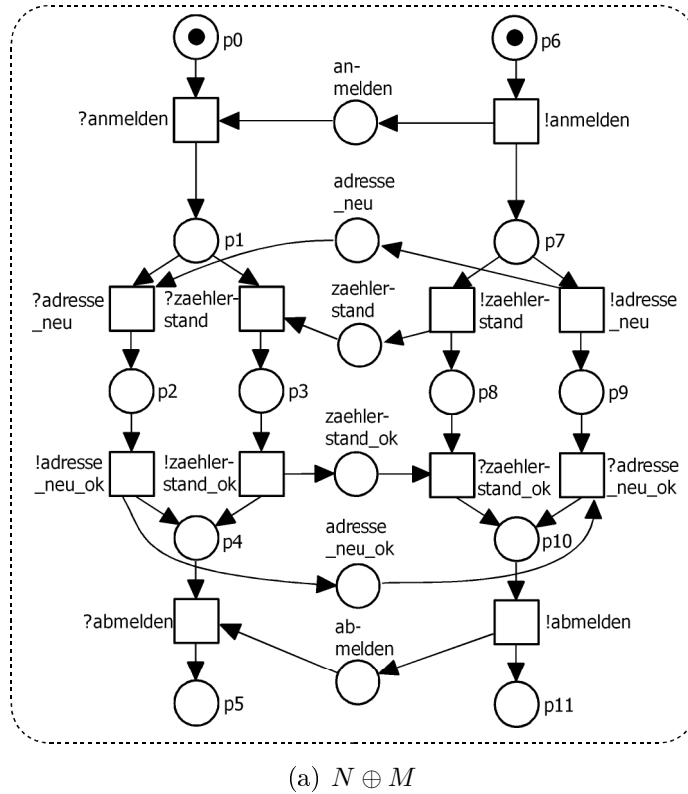


Abbildung 2.5: Die Komposition der beiden Partner oWFNs  $N$  und  $M$  aus Abbildung 2.4.

**Definition 2.2.11 (Schwach terminierend)**

Sei  $N = (P, P_i, P_o, T, F, \Omega)$  ein oWFN.  $N$  heißt schwach terminierend, wenn für alle  $m$  mit  $m_0 \xrightarrow{*} m$ , ein  $m_f \in \Omega$  existiert mit  $m \xrightarrow{*} m_f$ .

Das oWFN in Abbildung 2.4(a) ist nicht schwach terminierend, da von der Anfangsmarkierung keine Endmarkierung erreichbar ist. Das oWFN in Abbildung 2.5 ist dagegen schwach terminierend.

Ein Partner oWFN ist eine Strategie für ein gegebenes oWFN, wenn die Komposition der beiden oWFNs schwach terminiert.

**Definition 2.2.12 (Strategie)**

Seien  $N = (P_N, P_{i_N}, P_{o_N}, T_N, F_N, m_{0_N}, \Omega_N)$  und  $S = (P_S, P_{i_S}, P_{o_S}, T_S, F_S, m_{0_S}, \Omega_S)$  zwei Partner oWFNs.  $S$  ist eine Strategie für  $N$ , gdw  $N \oplus S$  schwach terminiert.

Das oWFN  $M$  in Abbildung 2.4(b) ist eine Strategie für das oWFN  $N$  in Abbildung 2.4(a).

## 2.3 Verhaltensbeschreibung offener Workflownetze mit Serviceautomaten

Serviceautomaten sind spezielle Automaten, die für die Verhaltensbeschreibung von Geschäftsprozessen verwendet werden können. Existiert für ein oWFN  $N$  eine Strategie, dann wird  $N$  bedienbar [Sch05] genannt. Bedienbarkeit eines oWFNs kann mit dem in [Sch05] vorgestellten Algorithmus entschieden werden, welcher in [Wei06] erweitert wurde. Dieser Algorithmus berechnet für ein bedienbares oWFN das Verhalten einer Strategie in Form eines Serviceautomaten. Hierzu wird zunächst ein Interaktionsgraph erstellt. Ist ein oWFN nicht bedienbar, kann dementsprechend keine Strategie berechnet werden. Mit einer Bedienungsanleitung können in einer Struktur alle Strategien eines oWFNs dargestellt werden. Ein Ziel dieser Arbeit ist es, zu einem gegebenen Serviceautomaten ein verhaltensäquivalentes oWFN zu berechnen.

### 2.3.1 Serviceautomat

Ein Serviceautomat (SVA) [MS05] dient dazu, das Verhalten eines Service zu beschreiben. Jeder SVA verfügt über eine endliche Menge von Kanälen, über die er kommunizieren kann. Es existieren Inputkanäle zum Empfangen von Nachrichten und Outputkanäle zum Senden von Nachrichten. Jeder SVA besitzt einen Anfangszustand und eine Menge von Endzuständen.

Wie bei oWFNs ist die Kommunikation zwischen zwei SVAs asynchron. Senden und Empfangen von Nachrichten sind demnach getrennt voneinander. Sendet ein oWFN eine Nachricht, landet auf dem entsprechenden Outputplatz eine Marke. Ein solches Konzept besitzt ein einzelner SVA nicht. Bei der Komposition zweier SVAs wird eine Nachricht, die von einem SVA gesendet, aber von dem anderen SVA noch nicht empfangen wurde, im jeweiligen Zustand repräsentiert.

#### Definition 2.3.1 (Serviceautomat)

Sei  $C$  eine endliche Menge von Nachrichtenkanälen und  $\tau \notin C$  eine Bezeichnung für einen internen Schritt des Automaten. Das Tupel  $A = (Q, I, O, \delta, q_0, F)$  ist ein Serviceautomat, mit:

- $Q$  ist eine Menge von Zuständen,
- $I \subseteq C$  ist eine Menge von Inputkanälen,
- $O \subseteq C$  ist eine Menge von Outputkanälen, wobei  $I \cap O = \emptyset$
- $\delta \subseteq Q \times (I \cup O \cup \{\tau\}) \times Q$  ist eine nichtdeterministische Transitionsrelation,

- $q_0 \in Q$  ist ein Anfangszustand,
- $F \subseteq \{q \mid x \in I, \text{ für alle } (q, x, q') \in \delta\}$  ist eine Menge von Endzuständen.

Ein SVA ist endlich, wenn die Menge der Zustände endlich ist. Eine Transition  $(q, x, q') \in \delta$  wird auch mit  $q \xrightarrow{x} q'$  dargestellt. Eine (möglicherweise leere) Transitionsfolge wird mit  $q \xrightarrow{*} q'$  dargestellt. In dieser Arbeit wird verlangt, daß jeder Zustand vom Anfangszustand erreichbar ist, also  $q_0 \xrightarrow{*} q$  für alle  $q \in Q$ .

Abbildung 2.6 zeigt den SVA  $A$ , der das Verhalten des oWFNs  $M$  aus Abbildung 2.4(b) beschreibt. Der Anfangszustand  $q_0$  wird durch einen Pfeil ohne Startzustand angezeigt. Der Endzustand  $q_5$  ist durch einen doppelten Kreis dargestellt.

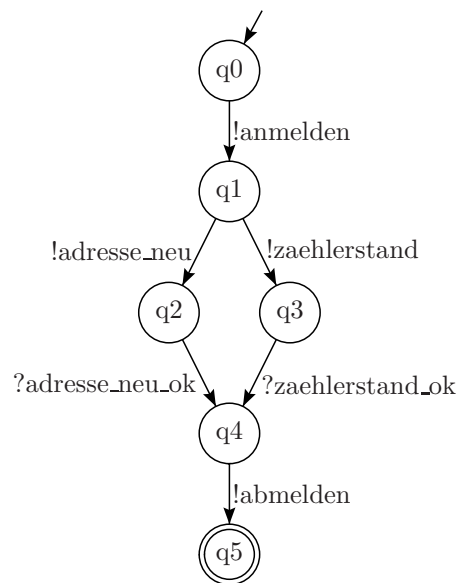


Abbildung 2.6: Ein SVA  $A$ , der das Verhalten des oWFNs  $M$  aus Abbildung 2.4(b) beschreibt.

**Definition 2.3.2 (Aktionen)**

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein SVA. Die Elemente der Menge  $I \cup O \cup \tau$  werden Aktionen genannt.

Eine Transitionsbeschriftung ordnet jeder Transition eine Beschriftung zu. Die Menge der Beschriftungen besteht aus den Aktionen mit einem zusätzlichen Index. Durch den Index können Transitionen mit einer identischen Aktion unterschiedlich behandelt werden. Bei der graphischen Darstellung wird der Index einer Aktion bei der Beschriftung weggelassen, wenn alle Beschriftungen der Transitionen mit dieser Aktion den gleichen Index haben. Aktionen aus der Menge der Inputkanäle erhalten bei der Beschriftung

ein vorangestelltes '?' und Aktionen aus der Menge der Outputkanäle erhalten bei der Beschriftung ein vorangestelltes '!'.

**Definition 2.3.3 (Transitionsbeschriftung eines Serviceautomaten)**

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein SVA. Die Funktion  $\lambda$  mit  $(q, x, q') \in \delta$  und  $i \in \mathbb{N}$  ist eine Transitionsbeschriftung für  $A$ , wenn gilt:

$$\lambda((q, x, q')) = \begin{cases} ?x_i, & \text{falls } x \in I, \\ !x_i, & \text{falls } x \in O, \\ \tau_i, & \text{sonst.} \end{cases}$$

Die Menge der Beschriftungen eines SVAs  $A$  besteht aus den Beschriftungen der einzelnen Transitionen von  $A$ .

**Definition 2.3.4 (Menge der Beschriftungen)**

Die Menge der Beschriftungen  $L_A$  eines SVAs  $A = (Q, I, O, \delta, q_0, F)$ , ist definiert als:  $L_A = \{l \mid \text{es existiert ein } (q, x, q') \in \delta, \text{ mit } \lambda((q, x, q')) = l\}$ .

Ein SVA wird vollständig terminierungsfähig genannt, wenn von jedem Zustand eine Folge von Transitionen zu einem Endzustand existiert.

**Definition 2.3.5 (Vollständig terminierungsfähiger Serviceautomat)**

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein SVA.  $A$  ist vollständig terminierungsfähig, wenn für alle  $q \in Q$  gilt: Es existiert ein  $q_f \in F$  mit  $q \xrightarrow{*} q_f$ .

Der SVA in Abbildung 2.6 ist vollständig terminierungsfähig. Der SVA in Abbildung 2.7 ist dagegen nicht vollständig terminierungsfähig, da von den Zuständen  $q_2$  und  $q_4$  keine Transitionsfolge zu einem Endzustand existiert.

## 2.3.2 Interaktionsgraph

Um zu entscheiden, ob ein gegebenes oWFN  $N$  bedienbar ist, kann ein Interaktionsgraph (IG) [Wei06] konstruiert werden. Ein IG ist ein gerichteter und beschrifteter Graph mit einem ausgezeichneten Wurzelknoten. Bei der Konstruktion eines IGs werden für  $N$  Mengen von Markierungen gebildet. Zwei Markierungen gehören einer solchen Menge an, wenn eine der Markierungen von der anderen Markierung durch Schalten aktivierter Transitionen von  $N$  erreicht werden kann. Jede vollständige Markierungsmenge mit dieser Eigenschaft bildet im IG einen Knoten. Eine neue Markierungsmenge wird erreicht, wenn einem Inputplatz eine Marke hinzugefügt wird, die zur Aktivierung einer

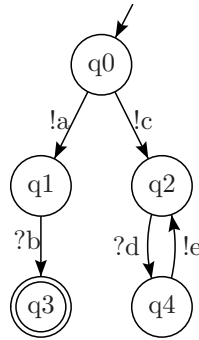


Abbildung 2.7: Ein nicht vollständig terminierungsfähiger Serviceautomat.

Transition führt, oder eine vorhandene Marke von einem Outputplatz entfernt wird. Das Hinzufügen und Entfernen von Marken auf die Input- und Outputplätze wird im IG durch die Kanten repräsentiert. Bei der graphischen Darstellung von IGs wird das Hinzufügen einer Marke auf einen Inputplatz  $p$  durch eine Kante mit der Beschriftung  $!p$  dargestellt. Das Entfernen einer Marke von einem Outputplatz  $p$  wird durch eine Kante mit der Beschriftung  $?p$  dargestellt. Ein IG eines oWFNs  $N$  beschreibt immer das Verhalten einer Strategie für  $N$ . Ist  $N$  nicht bedienbar, kann auch kein IG für  $N$  konstruiert werden.

Abbildung 2.8 zeigt den IG für das oWFN  $N$  aus Abbildung 2.4(a). Der erste Knoten besteht nur aus der Anfangsmarkierung  $[p0]$ , da bei der Anfangsmarkierung keine Transition von  $N$  aktiviert ist. Das Erreichen einer neuen Markierungsmenge ist nur möglich, wenn dem Inputplatz anmelden eine Marke hinzugefügt wird, da hierdurch die Transition mit der Beschriftung  $?anmelden$  aktiviert wird. Durch das Hinzufügen von Marken auf die anderen Inputplätze werden keine Transitions aktiviert. Das Hinzufügen einer Marke auf den Inputplatz anmelden entspricht im IG der Kante  $!anmelden$  und führt zu der Markierung  $[p0, anmelden]$ . Bei dieser Markierung ist die Transition  $?anmelden$  aktiviert. Durch das Schalten dieser Transition entsteht die Markierung  $[p1]$ . Bei dieser Markierung ist keine Transition aktiviert. Die Markierungen  $[p0, anmelden]$  und  $[p1]$  bilden daher im IG einen weiteren Knoten.

## 2.4 Synthese von Petrinetzen aus Serviceautomaten

Mit Hilfe der Regionentheorie [ER90] kann zu einem Transitionssystem, sofern dieses bestimmte Eigenschaften erfüllt, ein Petrinetz mit äquivalenten Verhalten erstellt werden. Im Rahmen dieser Arbeit wird die Regionentheorie nicht auf Transitionssysteme, sondern auf SVAs angewendet. Die folgenden Definitionen aus [DR96] und [CKLY98] wurden entsprechend angepasst.

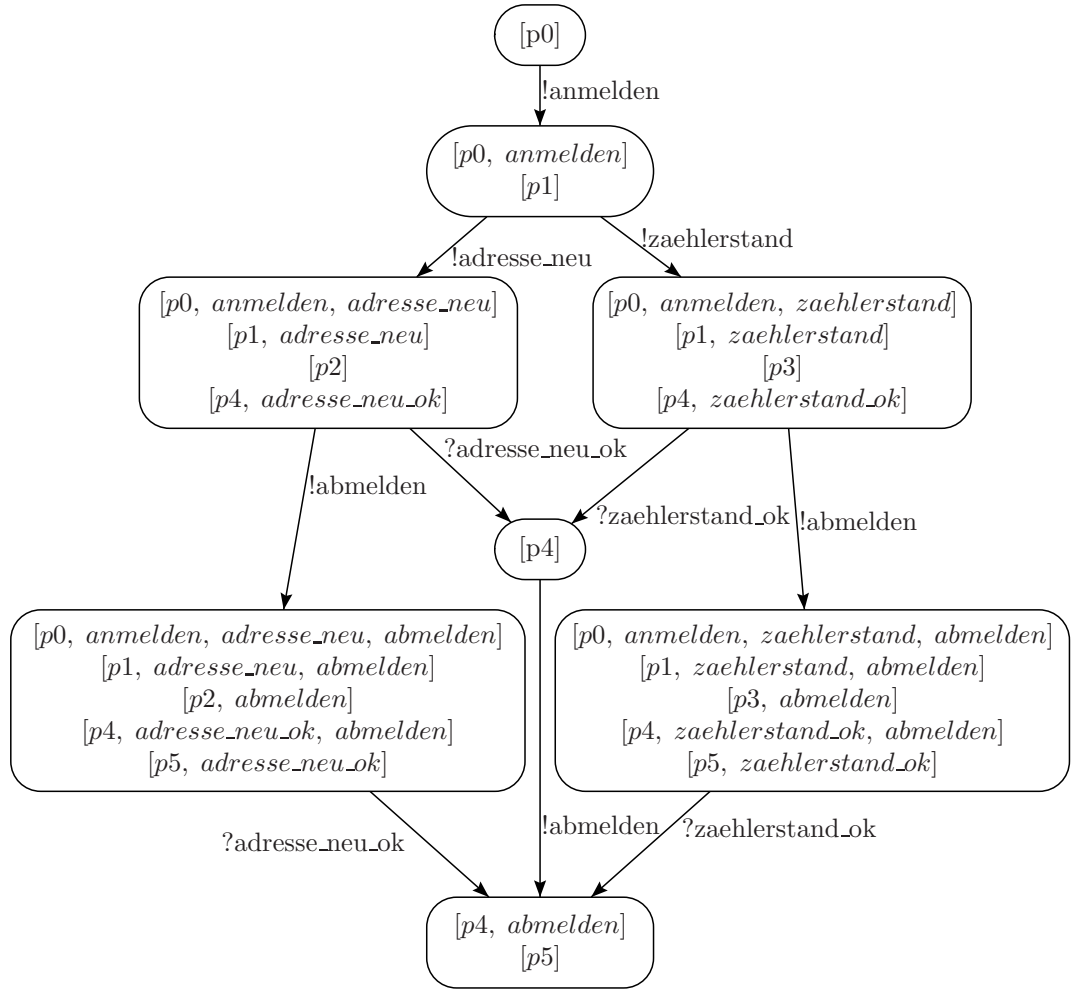


Abbildung 2.8: Interaktionsgraph des oWFNs  $N$  aus Abbildung 2.4(a).

### 2.4.1 Regionentheorie

Eine Region ist eine Teilmenge der Zustände eines SVAs. Diese Teilmenge muss bestimmte Eigenschaften bezüglich der Transitionen des SVAs erfüllen. Existiert in einem SVA eine Transition  $t = (q, x, q')$ , wobei  $q$  zur Region gehört und  $q'$  nicht zur Region gehört, muss dies für alle Transitionen gelten, die eine identische Beschriftung wie  $t$  besitzen. Gleiches gilt, falls  $q$  zur Region gehört und  $q'$  nicht zur Region gehört.

**Definition 2.4.1 (Region)**

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein SVA. Die Menge  $R, R \subseteq Q$ , ist eine Region von  $A$ , wenn für alle  $(q_1, x, q'_1) \in \delta$  und  $(q_2, x, q'_2) \in \delta$  mit  $\lambda((q_1, x, q'_1)) = \lambda((q_2, x, q'_2))$  gilt:

- $q_1 \in R$  und  $q'_1 \notin R$ , gdw.  $q_2 \in R$  und  $q'_2 \notin R$ ,

- $q_1 \notin R$  und  $q'_1 \in R$ , gdw.  $q_2 \notin R$  und  $q'_2 \in R$ .

Die Zustandsmenge  $\{q_0, q_1, q_2\}$  des SVAs  $A$  in Abbildung 2.9 ist keine Region. Es existieren zwei Transitionen mit der Beschriftung  $!anmelden$ . Bei der Transition  $(q_0, anmelden, q_1)$  befinden sich der Startknoten  $q_0$  und der Zielknoten  $q_1$  innerhalb der Zustandsmenge. Bei der Transition  $(q_2, anmelden, q_4)$  ist dagegen nur der Startknoten innerhalb der Zustandsmenge. Die Zustandsmenge  $\{q_0, q_2, q_5\}$  ist dagegen eine Region von  $A$ .

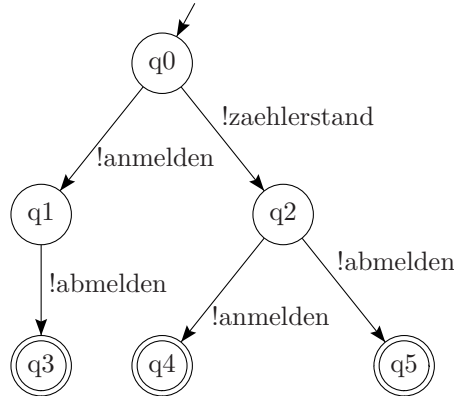


Abbildung 2.9: Ein Serviceautomat  $A$ .

Die Mengen  $\emptyset$  und  $Q$  werden *triviale Regionen* genannt. Alle anderen Regionen gehören zu den *nicht trivialen Regionen*. Die Menge der nicht trivialen Regionen eines SVAs  $A$  wird mit  $R_A$  bezeichnet. Die Menge  $R_q = \{R \mid q \in R, R \in R_A\}$  besteht aus allen nicht trivialen Regionen, in denen der Zustand  $q$  enthalten ist. Eine nicht triviale Region  $r$  ist *minimal*, wenn die Zustandsmenge keiner anderen nicht trivialen Region eine Teilmenge der Zustandsmenge von  $r$  ist. Die Menge der *minimalen nicht trivialen Regionen* eines SVAs  $A$  wird mit  $minR_A$  bezeichnet.

Die Menge der nicht trivialen Regionen des SVAs  $A$  in Abbildung 2.9 ist die Menge  $R_A = \{\{q_0, q_1, q_2, q_4\}, \{q_0, q_1, q_3\}, \{q_0, q_2, q_5\}, \{q_1, q_3, q_4\}, \{q_2, q_4, q_5\}, \{q_3, q_5\}\}$ . Für  $A$  gilt  $R_A = minR_A$ , da die nicht trivialen Regionen alle minimal sind. Die Menge  $R_{q_5} = \{\{q_0, q_2, q_5\}, \{q_2, q_4, q_5\}, \{q_3, q_5\}\}$  besteht aus den nicht trivialen Regionen, denen der Zustand  $q_5$  angehört.

Befinden sich alle Wurzelknoten von Transitionen mit gleicher Beschriftung innerhalb einer Region und die Zielknoten dieser Transitionen außerhalb einer Region, wird diese Region als *Vorregion* dieser Transitionen bezeichnet. Die Menge aller Vorregionen der Transitionen mit der Beschriftung  $l$  wird dargestellt mit  $ol$ .

#### Definition 2.4.2 (Menge der Vorregionen)

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein SVA. Die Menge  $ol = \{R \in R_A \mid \text{für alle } (q, x, q') \in \delta \text{ mit } \lambda((q, x, q')) = l, q \in R, q' \notin R\}$  wird als Menge der Vorregionen von  $l$  bezeichnet.

Die Menge der Vorregionen der Transitionen mit der Beschriftung  $!anmelden$  im SVA  $A$  aus Abbildung 2.9 ist  $\circ(!anmelden) = \{\{q0, q2, q5\}\}$ .

Die Menge der Nachregionen wird analog zu der Menge der Vorregionen definiert.

**Definition 2.4.3 (Menge der Nachregionen)**

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein SVA. Die Menge  $l_\circ = \{R \in R_A \mid \text{für alle } (q, x, q') \in \delta \text{ mit } \lambda((q, x, q')) = l, q \notin R, q' \in R\}$  wird als Menge der Nachregionen von  $l$  bezeichnet.

Die Menge der Nachregionen der Transitionen mit der Beschriftung  $!anmelden$  im SVA  $A$  aus Abbildung 2.9 ist  $(!anmelden)_\circ = \{\{q1, q3, q4\}\}$ .

## 2.4.2 Elementarer Serviceautomat

Zu einem SVA kann nur dann ein Petrinetz mit äquivalenten Verhalten erstellt werden, wenn es sich um einen elementaren Serviceautomaten (ESVA) handelt. Ein SVA  $A$  ist elementar, wenn  $A$  bestimmte Eigenschaften erfüllt. Die folgende Definition basiert auf der Definition für elementare Automaten aus [CKLY98] und ist an die in dieser Arbeit verwendeten SVAs angepasst.

**Definition 2.4.4 (Elementarer Serviceautomat)**

Ein SVA  $A = (Q, I, O, \delta, q_0, F)$  ist ein elementarer Serviceautomat, wenn die folgenden Axiome erfüllt sind:

*A1 Keine Selbstschleifen:*  $q \neq q'$ , für alle  $(q, x, q') \in \delta$ .

*A2 Jede Beschriftung wird verwendet:* Für alle  $l \in L_A$  existiert ein  $(q, x, q') \in \delta$  mit  $\lambda((q, x, q')) = l$ .

*A3 Jeder Zustand ist vom Anfangszustand erreichbar:*  $q_0 \xrightarrow{*} q'$ , für alle  $q' \in Q$ .

*A4 Separate Zustände:* Wenn  $R_q = R_{q'}$ , dann  $q = q'$ , mit  $q, q' \in Q$ .

*A5 Abgeschlossene Vorregionen:* Wenn  $\circ x \subseteq R_q$ , dann existiert ein  $(q \xrightarrow{x} q') \in \delta$ .

Abbildung 2.9 zeigt einen SVA  $A$ , der nicht elementar ist. Für  $A$  gilt  $\circ(!anmelden) = \{\{q0, q2, q5\}\} \subset \{\{q0, q2, q5\}, \{q2, q4, q5\}, \{q3, q5\}\} = R_{q5}$ . Nach Axiom A5 muss  $A$  eine Transition  $!anmelden$  besitzen, die  $q5$  als Startknoten besitzt. Dies ist nicht der Fall und  $A$  ist somit kein ESVA. Der SVA  $A'$  in Abbildung 2.10 erfüllt dagegen alle geforderten Axiome und ist daher elementar.

Jeder SVA bei dem die ersten drei Axiome erfüllt sind, kann durch das Aufsplitten der Beschriftungen in einen splitmorphen ESVA umgewandelt werden. Zwei Beschriftungen werden aufgesplittet, indem ihnen ein unterschiedlicher Index zugeordnet wird. Ein Splitmorphismus ist ein Paar totaler Abbildungen. Existiert von einem SVA  $A_1$  ein Splitmorphismus zu einem SVA  $A_2$ , dann ist  $A_2$  splitmorph zu  $A_1$ .

**Definition 2.4.5 (Splitmorphismus)**

Seien  $A_1 = (Q_1, I_1, O_1, \delta_1, q_{0_1}, F_1)$  und  $A_2 = (Q_2, I_2, O_2, \delta_2, q_{0_2}, F_2)$  zwei SVAs. Ein Splitmorphismus  $h$  von  $A_1$  nach  $A_2$  ist ein Paar  $(h_Q, h_L)$  totaler Abbildungen für die gilt:

- $h_Q : Q_1 \rightarrow Q_2$  ist bijektiv,
- $h_L : L_1 \rightarrow L_2$  ist surjektiv,
- $(q, x, q') \in \delta_1$ , genau dann wenn  $(h_Q(q), x, h_Q(q')) \in \delta_2$ , mit  $h_L(\lambda_1((q, x, q'))) = \lambda_2(h_Q(q), x, h_Q(q'))$ .

Der nicht elementare SVA  $A$  in Abbildung 2.9, wird durch das Aufsplitten der Beschriftung !anmelden in die Beschriftungen !anmelden<sub>1</sub> und !anmelden<sub>2</sub> zu dem splitmorphen ESVA  $A'$  in Abbildung 2.10.

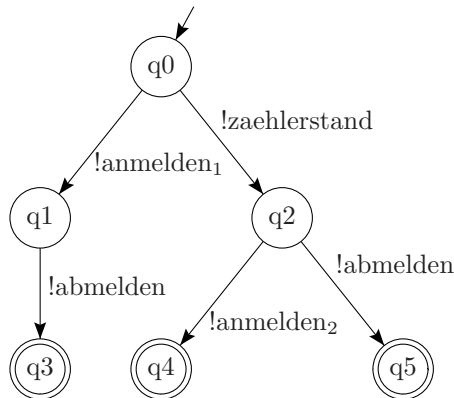


Abbildung 2.10: Ein elementarer SVA  $A'$ , der aus dem nicht elementaren SVA  $A$  in Abbildung 2.9, durch das Aufsplitten der Beschriftung !anmelden, entstanden ist.

**2.4.3 Synthetisiertes Petrinetz**

Zu einem ESVA  $A$  kann ein synthetisiertes Petrinetz  $N$  derart generiert werden, dass  $N$  zu  $A$  ein äquivalentes Verhalten besitzt. Jeder Platz von  $N$  entspricht einer minimalen

nicht trivialen Region von  $A$ . Für jedes Element aus der Menge der verwendeten Beschriftungen von  $A$ , besitzt  $N$  eine Transition mit einer identischen Beschriftung. Eine Kante von einem Platz zu einer Transition ist genau dann vorhanden, wenn die dem Platz entsprechende Region in  $A$  eine Vorregion der Transitionen mit dieser Beschriftung ist. Handelt es sich um eine Nachregion der Transitionen mit dieser Beschriftung, besitzt  $N$  eine Kante von der entsprechenden Transition zu diesem Platz. Die Anfangsmarkierung besteht aus allen Plätzen, die in den entsprechenden Regionen den Anfangszustand des SVAs besitzen.

**Definition 2.4.6 (Synthetisiertes Petrinetz)**

Sei  $A = (Q, I, O, \delta, q_0, F_A)$  ein ESVA. Ein Petrinetz  $N(A) = (P, T, F, m_0)$  ist ein aus  $A$  synthetisiertes Petrinetz (SPN), wenn gilt:

- $P = \text{min}R_A$ ,
- $T = \{t_i \mid \lambda(t_i) = l_i, l_i \in L_A = \{l_1, \dots, l_n\}, 1 \leq i \leq n\}$ ,
- $F = \{(r, t) \mid r \in \circ(\lambda(t))\} \cup \{(t, r) \mid r \in (\lambda(t))\circ\}$ , mit  $r \in \text{min}R_A$  und  $t \in T$ ,
- $m_0 = \{r \mid q_0 \in r, r \in \text{min}R_A\}$ .

Abbildung 2.11 zeigt das zum ESVA  $A'$  aus Abbildung 2.10 synthetisierte Petrinetz  $N(A')$ . Die Plätze entsprechen den minimalen nicht-trivialen Regionen  $r_0 = \{q_0\}$ ,  $r_1 = \{q_1, q_3\}$ ,  $r_2 = \{q_1, q_2\}$ ,  $r_3 = \{q_2, q_5\}$ ,  $r_4 = \{q_3, q_5\}$  und  $r_5 = \{q_4\}$ . Die Region  $r_0$  ist eine Vorregion der Beschriftung !anmelden<sub>1</sub>, daher führt eine Kante von dem Platz  $r_0$  zu der Transition mit der Beschriftung !anmelden<sub>1</sub>. Die Region  $r_1$  gehört zur Menge der Nachregionen der Beschriftung !anmelden<sub>1</sub>, daher führt eine Kante von der Transition mit der Beschriftung !anmelden<sub>1</sub>, zu dem Platz  $r_1$ . Die Anfangsmarkierung ist  $m_0 = [r_0]$ , da die diesem Platz entsprechende Region  $r_0$ , als einzige Region den Anfangszustand  $q_0$  enthält.

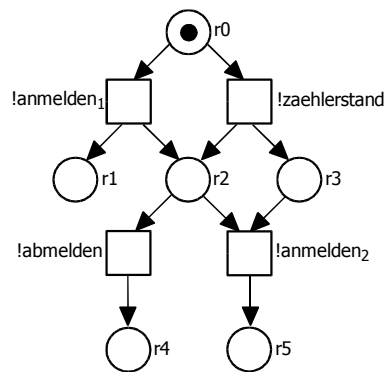


Abbildung 2.11: Synthetisiertes Petrinetz  $N(A')$  zum ESVA aus Abbildung 2.10.

Um die Verhaltensäquivalenz zwischen einem SVA  $A$  und dem zugehörigen SPN  $N(A)$  zu zeigen, wird im Folgenden der Erreichbarkeitsgraph eines Petrinetzes  $N$  benötigt. Der Erreichbarkeitsgraph von  $N(A)$  ist ein beschrifteter und gerichteter Graph, bei dem jeder Knoten eine erreichbare Markierung von  $N$  repräsentiert. Zwischen zwei Knoten existiert genau dann eine Kante, wenn in  $N$  von der einen Markierung in die andere Markierung geschaltet werden kann.

**Definition 2.4.7 (Erreichbarkeitsgraph)**

Sei  $N = (P, T, F, m_0)$  ein Petrinetz und  $M$  die Menge aller erreichbaren Markierungen von  $N$ . Das Tupel  $EG(N) = (L, V, E, v_0)$  ist ein Erreichbarkeitsgraph, mit:

- $L = \{l \mid l = \lambda(t), t \in T\}$  ist eine Menge von Beschriftungen,
- $V = M$  ist eine Menge von Knoten,
- $E = \{(m, l, m') \mid m, m' \in V, \lambda(t) = l, m \xrightarrow{t} m'\}$  ist eine Menge von Kanten,
- $v_0 = m_0$  ist der Wurzelknoten.

Abbildung 2.12 zeigt den Erreichbarkeitsgraphen des SPNs  $N(A')$  aus Abbildung 2.11. Der Wurzelknoten  $[r0]$  entspricht der Anfangsmarkierung  $m_0 = [r0]$  von  $N$ . Bei dieser Markierung sind in  $N$  die beiden Transitionen  $!anmelden_1$  und  $!zaehlerstand$  aktiviert. Schalten der Transition  $!anmelden_1$  führt zu der Markierung  $m = [r1, r2]$ . Der Erreichbarkeitsgraph besitzt daher eine Kante vom Wurzelknoten zum Knoten  $[r1, r2]$ .

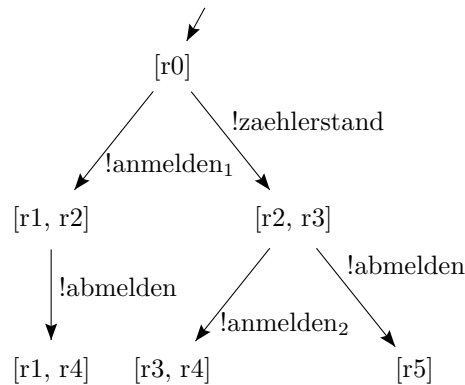


Abbildung 2.12: Erreichbarkeitsgraph  $EG(N(A'))$  des SPNs  $N(A')$  aus Abbildung 2.11.

Eine *Bisimulation* [Mil89] ist eine Relation zwischen zwei Transitionssystemen. In dieser Arbeit wird diese Relation verwendet um das Verhalten zwischen einem SVA und einem Erreichbarkeitsgraphen in Beziehung zu setzen und dementsprechend definiert. Ein SVA

$A$  und ein Erreichbarkeitsgraph  $EG$  sind *bisimilar* (geschrieben  $A \sim EG$ ), wenn zwischen  $A$  und  $EG$  eine Bisimulation existiert.

**Definition 2.4.8 (Bisimulation)**

Sei  $\sim$  eine zweistellige Relation zwischen einem SVA  $A = (Q, I, O, \delta, q_0, F)$  und einem Erreichbarkeitsgraphen  $EG = (L, V, E, v_0)$ . Die Relation  $\sim$  ist eine Bisimulation zwischen  $A$  und  $EG$ , wenn gilt:

- für jedes  $q \in Q$  existiert ein  $v \in V$ , mit  $q \sim v$ ,
- für jedes  $v \in V$  existiert ein  $q \in Q$ , mit  $q \sim v$ ,
- für jedes  $(q, x, q') \in \delta$  und jedes  $v \in V$ , mit  $q \sim v$ , existiert ein  $(v, l, v') \in E$  mit  $\lambda((q, x, q')) = l$ , so dass gilt  $q' \sim v'$ ,
- für jedes  $(v, l, v') \in E$  und jedes  $q \in Q$ , mit  $q \sim v$  existiert ein  $(q, x, q') \in \delta_A$  mit  $l = \lambda((q, x, q'))$ , so dass gilt  $q' \sim v'$ .

Der ESVA  $A'$  in Abbildung 2.10 und der Erreichbarkeitsgraph  $EG(N(A'))$  in Abbildung 2.12 sind bisimilar. Die Relation  $\{(q_0, [r_0]), (q_1, [r_1, r_2]), (q_2, [r_2, r_3]), (q_3, [r_1, r_4]), (q_4, [r_3, r_4]), (q_5, [r_5])\}$  ist eine Bisimulation zwischen  $A'$  und  $EG(N(A'))$ . In Abbildung 2.13 ist diese Relation zwischen  $A'$  und  $EG(N(A'))$  dargestellt. Die gestrichelten Pfeile zeigen an, welcher Zustand von  $A'$  mit welchem Knoten von  $EG(N(A'))$  in Relation steht.

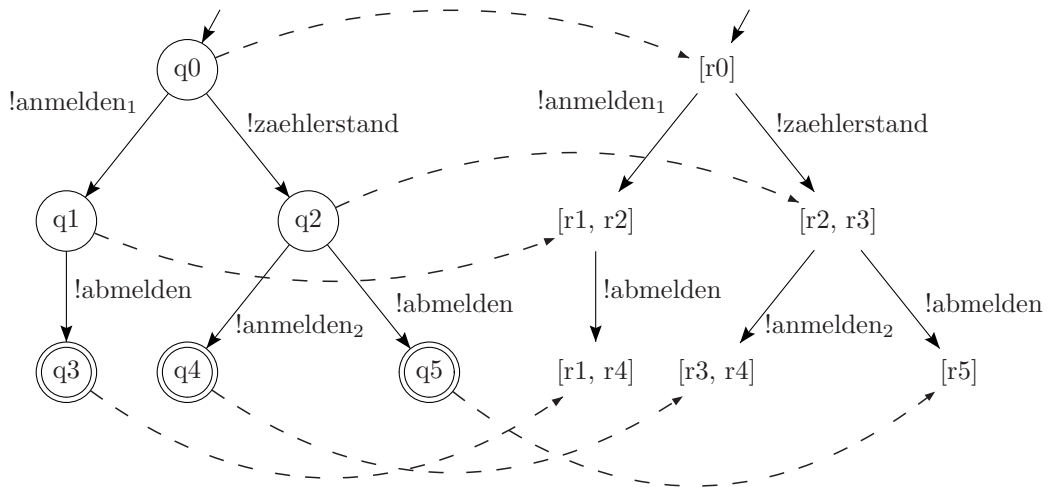


Abbildung 2.13: Bisimulation zwischen dem ESVA  $A'$  aus Abbildung 2.10 und dem Erreichbarkeitsgraph  $EG(N(A'))$  aus Abbildung 2.12.

Zu jedem ESVA  $A$  kann ein synthetisiertes Petrinetz  $N(A)$  generiert werden, so dass gilt  $A \sim EG(N(A))$  [CKLY98].

# 3 Synthese offener Workflownetze aus Serviceautomaten

Das Ziel dieser Arbeit besteht darin, einen Serviceautomat (SVA) in ein offenes Workflownetz (oWFN) zu übersetzen. Darüber hinaus soll das entstandene oWFN alle notwendigen Eigenschaften besitzen, um in die Web Services Business Process Execution Language (WS-BPEL) übersetzt werden zu können.

Im ersten Schritt der Übersetzung eines SVAs in ein oWFN muss der SVA in einen elementaren Serviceautomat (ESVA) transformiert werden. Abschnitt 3.1 zeigt, dass jeder SVA, bei dem jeder Zustand vom Anfangszustand erreichbar ist, in einen ESVA transformiert werden kann. Zu jedem ESVA kann ein synthetisiertes Petrinetz (SPN) berechnet werden (vgl. Abschnitt 2.4.3). Abschnitt 3.2 zeigt, dass jedes SPN zu einem oWFN erweitert werden kann. Die Beschreibung und Zusicherung der notwendigen Eigenschaften von oWFNs für die Übersetzung in WS-BPEL wird in Abschnitt 3.3 behandelt. Der vierte Abschnitt beschäftigt sich mit der Frage, wie sich die beschriebenen Transformation eines SVAs auf dessen Verhalten auswirken.

## 3.1 Transformation von Serviceautomaten in elementare Serviceautomaten

Um aus einem SVA  $A$  mit Hilfe der Regionentheorie (vgl. Abschnitt 2.4.1) ein Petrinetz synthetisieren zu können, muss es sich bei  $A$  um einen ESVA handeln. Die fünf Axiome eines ESVA (vgl. Abschnitt 2.4.2) müssen daher für  $A$  erfüllt sein. Die Axiome A4 und A5 können durch Aufsplitten von Transitionsbeschriftungen immer erfüllt werden. Im Extremfall erhält jede Transition eine eigene Beschriftung. Jeder Zustand des SVAs wird dann zu einer minimalen nicht-trivialen Region und die beiden Axiome sind immer erfüllt [CKLY98]. Die Erfüllung der ersten drei Axiome wird in [CKLY98] vorausgesetzt und muss daher hier noch zugesichert werden.

Axiom A1 erlaubt keine Selbstschleifen, d.h. der Startzustand und der Zielzustand einer Transition dürfen nicht identisch sein. Besitzt ein Zustand  $q$  eines SVAs eine Selbstschleife, kann diese durch hinzufügen eines neuen Zustandes  $q'$  und einer mit  $\tau$  beschrifteten

Transition umgewandelt werden. Die Transition mit identischem Startzustand und Endzustand behält  $q$  als Startzustand und erhält den neu hinzugefügten Zustand  $q'$  als Zielzustand. Die hinzugefügte  $\tau$ -beschriftete Transition besitzt  $q'$  als Startzustand und  $q$  als Zielzustand.

Der Zustand  $q_0$  des SVAs  $A$  in Abbildung 3.1(a) besitzt zwei Selbstschleifen. Durch das Hinzufügen des Zustands  $q_0'$  und einer  $\tau$ -Kante wurde  $A$  in den SVA  $A'$  in Abbildung 3.1(b) ohne Selbstschleifen transformiert.

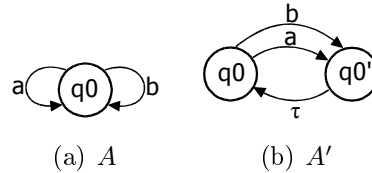


Abbildung 3.1: Transformation des SVAs  $A$  in den SVA  $A'$  ohne Selbstschleifen.

Das Axiom A2 verlangt, dass jede Beschriftung verwendet wird. Dies folgt unmittelbar aus der Definition für die Menge der Beschriftungen von SVAs (vgl. Abschnitt 2.3.1).

Die Forderung von Axiom A3, dass jeder Zustand vom Anfangszustand aus erreichbar ist, wird in dieser Arbeit für SVAs vorausgesetzt (vgl. Abschnitt 2.3.1). Jeder SVA, der diese Forderung erfüllt, kann somit in einen ESVA transformiert werden.

## 3.2 Synthetisiertes offenes Workflownetz

Um ein SPN  $N(A)$ , das aus einem ESVA  $A$  synthetisiert wurde, zu einem oWFN  $N'(A)$  zu erweitern, sind zwei Schritte notwendig. Die Input- und Outputplätze müssen mit den zugehörigen Kanten hinzugefügt werden und die Anfangs- und Endmarkierungen müssen angepasst bzw. bestimmt werden. Für jeden Input- und Outputkanal von  $A$ , wird  $N'(A)$  ein Input- und Outputplatz hinzugefügt und mit den entsprechenden Transitionen über eine neue Kante verbunden. Die Anfangsmarkierung von  $N'(A)$  entspricht der Anfangsmarkierung von  $N(A)$ . Endmarkierung von  $N'(A)$  ist jede Markierung, bei der in  $N(A)$  keine Transition aktiviert ist. Da  $N(A)$  keine Input- und Outputplätze besitzt, sind diese Plätze bei den Anfangs- und Endmarkierungen von  $N'(A)$  nicht markiert.

### Definition 3.2.1 (Synthetisiertes offenes Workflownetz)

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein ESVA, und  $N(A) = (P_{spn}, T_{spn}, F_{spn}, m_{0_{spn}})$  ein aus  $A$  synthetisiertes Petrinetz mit der Menge aller erreichbaren Markierungen  $M$ .  $N(A)$ , erweitert zu einem synthetisierten offenen Workflownetz  $N'(A) = (P, P_i, P_o, T, F, m_0, \Omega)$  mit  $m_{io} : P_i \cup P_o \rightarrow \{0\}$ , ist definiert als:

- $P_i = I,$
- $P_o = O,$
- $P = P_{spn} \cup P_i \cup P_o,$
- $T = T_{spn},$
- $F = F_{spn}$   
 $\cup \{(p, t) \mid p \in P_i, t \in T, (q, p, q') \in \delta, \lambda((q, p, q')) = \lambda(t)\}$   
 $\cup \{(t, p) \mid p \in P_o, t \in T, (q, p, q') \in \delta, \lambda((q, p, q')) = \lambda(t)\},$
- $m_0 = m_{0_{spn}} \oplus m_{io},$
- $\Omega = \{m_{spn} \oplus m_{io} \mid \text{bei } m_{spn} \in M \text{ ist in } N(A) \text{ keine Transition aktiviert}\}.$

Das SPN  $N(A')$  in Abbildung 2.11 wurde aus dem ESVA  $A'$  in Abbildung 2.10 synthetisiert. Abbildung 3.2 zeigt  $N(A')$  erweitert zu dem oWFN  $N'(A')$ . Für jeden Inputkanal und Outputkanal von  $A'$  erhält  $N'(A')$  einen zusätzlichen Platz. Da  $A'$  keine Inputkanäle besitzt wurden nur für die drei Outputkanäle *anmelden*, *zaehlerstand* und *abmelden* die drei entsprechenden Outputplätze hinzugefügt. Für jede Beschriftung einer Transition von  $A'$  existiert in  $N'(A')$  eine identisch beschriftete Transition. Nutzt eine Transition von  $A'$  einen Inputkanal bzw. Outputkanal, dann wird die identisch beschriftete Transition von  $N'(A')$  mit dem entsprechenden Inputplatz bzw. Outputplatz mit einer Kante verbunden. In  $A'$  existieren die beiden Transitionen  $(q0, \text{anmelden}, q1)$  und  $(q2, \text{anmelden}, q4)$ , die jeweils den Outputkanal *anmelden* nutzen. Die Beschriftungen dieser beiden Transitionen sind gesplittet in  $!anmelden_1$  und  $!anmelden_2$ . Die Transitionen von  $N'(A')$  mit der Beschriftung  $!anmelden_1$  und  $!anmelden_2$  besitzen daher beide eine Kante zu dem Outputplatz *anmelden*, der dem Outputkanal *anmelden* von  $A'$  entspricht. Die Anfangsmarkierung von  $N'(A')$  ist  $m_0 = [r0]$  und die Endmarkierungen sind  $m_{f1} = [r1, r4]$ ,  $m_{f2} = [r3, r4]$  und  $m_{f3} = [r5]$ .

### 3.3 Notwendige Eigenschaften von oWFNs zur Übersetzung in WS-BPEL

In den vorherigen Abschnitten wurde bereits gezeigt, wie zu einem SVA  $A$  ein oWFN  $N(A)$  synthetisiert werden kann.  $N(A)$  soll zudem in WS-BPEL übersetzt werden können. Hierfür muss  $N(A)$  bestimmte Eigenschaften besitzen. Bereits in  $A$  kann erkannt werden, ob  $N(A)$  die notwendigen Eigenschaften zur Übersetzung in WS-BPEL besitzen wird. Die Zusicherung dieser Eigenschaften von  $N(A)$ , durch Transformationen von  $A$ , ist das zweite Ziel dieser Arbeit und wird in diesem Abschnitt beschrieben.

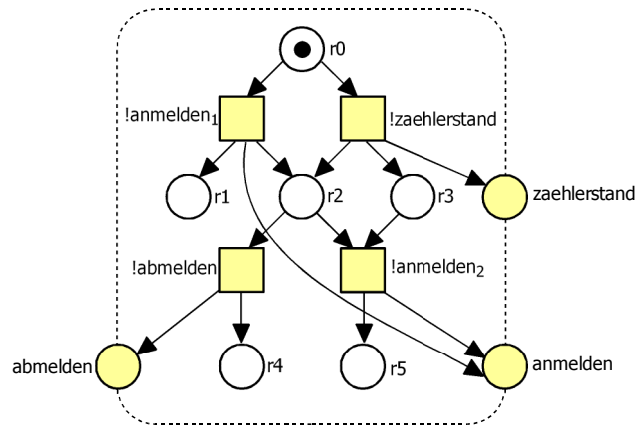


Abbildung 3.2: Das SPN  $N(A')$  aus Abbildung 2.11 zu dem oWFN  $N'(A')$  erweitert.

### 3.3.1 Empfangskonflikt

Ein oWFN besitzt einen Empfangskonflikt, wenn bei einer erreichbaren Markierung zwei Transitionen aktiviert sind, die im Vorbereich einen identischen Inputplatz besitzen. Ein oWFN mit einem Empfangskonflikt kann nicht in WS-BPEL übersetzt werden, da Empfangskonflikte in WS-BPEL nicht erlaubt sind.

Abbildung 3.3(b) zeigt ein oWFN  $N(A)$ , das aus dem ESVA  $A$  in Abbildung 3.3(a) erstellt wurde. Bei der dargestellten Markierung in  $N(A)$ , mit jeweils einer Marke auf den Plätzen  $p_0$  und  $a$ , entsteht ein Empfangskonflikt. Die beiden Transitionen mit den Beschriftungen  $?a_1$  und  $?a_2$  sind aktiviert und besitzen beide den Inputplatz  $a$  in ihrem Vorbereich.

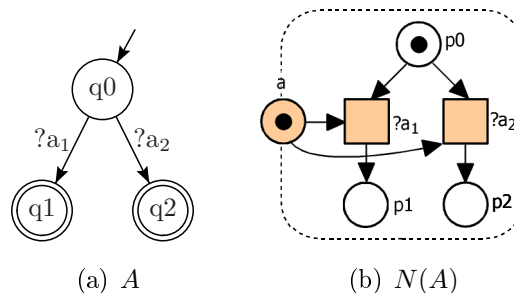


Abbildung 3.3: Ein ESVA  $A$  und das daraus synthetisierte oWFN  $N(A)$  mit einem Empfangskonflikt.

Im ESVA  $A$  in Abbildung 3.3 (a) kann der Empfangskonflikt durch eine Transformation von  $A$  verhindert werden. Der ESVA  $A'$  in Abbildung 3.4 (a) ist aus  $A$  entstanden. Es

wurde der neue Zustand  $q0'$  und die Transition mit der Beschriftung  $\tau$  hinzugefügt. Die Transition mit der Beschriftung  $?a_2$  beginnt nun im Zustand  $q0'$ . Das aus  $A'$  synthetisierte oWFN  $N'(A')$  in Abbildung 3.4(b) besitzt keinen Empfangskonflikt.

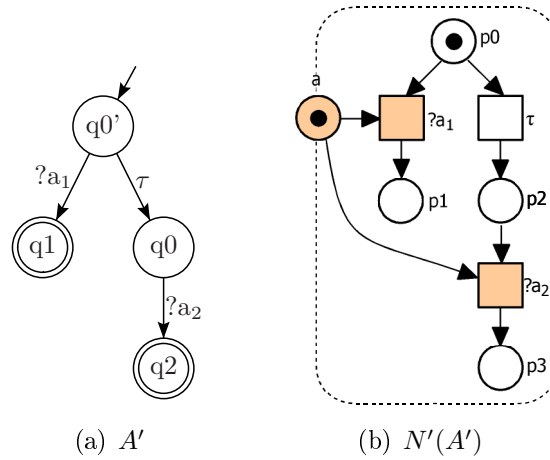


Abbildung 3.4: Ein aus dem ESVA  $A'$  (a) synthetisiertes oWFN  $N'(A')$  (b) ohne Empfangskonflikt.

Satz 3.3.2 besagt, dass ein Empfangskonflikt durch Hinzufügen von Zuständen und  $\tau$ -beschrifteten Kanten immer verhindert werden kann. Grundlage für diesen Satz ist Lemma 3.3.1. In diesem Lemma wird gezeigt, dass ein Empfangskonflikt nur auftreten kann, wenn ein Zustand zwei ausgehende Transitionen besitzt, die Nachrichten vom gleichen Inputkanal konsumieren.

### Lemma 3.3.1

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein ESVA und  $N(A) = (P, P_i, P_o, T, F, m_0, \Omega)$  ein aus  $A$  synthetisiertes oWFN. Ein Empfangskonflikt kann in  $N(A)$  nur auftreten, wenn  $A$  einen Zustand  $q$  besitzt, für den gilt:  $(q, x, q') \in \delta, (q, x, q'') \in \delta, q' \neq q''$  und  $x \in I$ .

**Beweis.** Angenommen  $N(A)$  besitzt einen Empfangskonflikt. Dann besitzt  $N(A)$  eine Markierung  $m$ , bei der zwei Transitionen  $t_1$  und  $t_2$  aktiviert sind, für die gilt:  $p \in \text{ot}_1, p \in \text{ot}_2$  und  $p \in P_i$ . Der dieser Markierung  $m$  entsprechende Knoten  $v$  im Erreichbarkeitsgraph von  $N(A)$  besitzt dann zwei ausgehende Kanten beschriftet mit  $\lambda(t_1)$  und  $\lambda(t_2)$ . Da der Erreichbarkeitsgraph bisimilar zu  $A$  ist, muss in  $A$  ein zu  $v$  in Relation stehender Knoten  $q$  mit  $(q, x, q') \in \delta, (q, x, q'') \in \delta$  existieren, so dass:  $\lambda((q, x, q')) = \lambda(t_1)$  und  $\lambda((q, x, q'')) = \lambda(t_2)$ . Wegen  $p \in \text{ot}_1, p \in \text{ot}_2$  und  $p \in P_i$  und Definition 3.2.1 muss  $x \in I$  gelten.  $\square$

### Satz 3.3.2

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein ESVA und  $N(A) = (P, P_i, P_o, T, F, m_0, \Omega)$  ein aus  $A$  synthetisiertes oWFN. Besitzt  $N(A)$  Empfangskonflikte kann  $A$  so transformiert werden, dass  $N(A)$  keine Empfangskonflikte mehr besitzt.

**Beweis.** Besitzt  $N(A)$  einen Empfangskonflikt muss wegen Lemma 3.3.1 ein Zustand  $q_1$  in  $N(A)$  existieren, der mindestens zwei ausgehende Transitionen besitzt, die Nachrichten vom gleichen Inputkanal konsumieren. Sei  $n \in \mathbb{N}$  die höchste Anzahl von ausgehenden Transitionen des Zustands  $q_1$ , die Nachrichten vom gleichen Inputkanal konsumieren.  $A$  erhält  $n - 1$  zusätzliche  $\tau$ -Transitionen und die zusätzlichen Zustände  $q_2, q_3, \dots, q_n$ . Von dem Zustand  $q_n$  führt eine Folge von  $\tau$ -Transitionen zum Zustand  $q_1$  ( $q_n \xrightarrow{\tau} q_{n-1} \xrightarrow{\tau} \dots \xrightarrow{\tau} q_1$ ). Alle Transitionen mit  $q_1$  als Zielzustand erhalten  $q_n$  als Zielzustand. Alle  $\tau$ -Transitionen und alle Transitionen, deren Aktion zu der Menge der Outputkanäle gehört, mit  $q_1$  als Startzustand bleiben unverändert. Sei  $t_1, \dots, t_m$ , mit  $1 \leq m \leq n$  eine Menge von Transitionen mit  $t_i = (q_1, x, q'_i)$ , die Nachrichten vom gleichen Inputkanal  $x$  konsumieren und  $q_1$  als Startzustand besitzen. Dann erhält jede dieser Transitionen einen neuen Startzustand nach dem Schema  $t_i = (q_i, x, q'_i)$ .

Wird diese Transformation für alle Zustände von  $A$  durchgeführt, die ausgehende Transitionen besitzen, die Nachrichten vom gleichen Inputkanal konsumieren, besitzen alle Zustände von  $A$  keine Transitionen, die Nachrichten vom gleichen Inputkanal konsumieren und  $N(A)$  somit keine Empfangskonflikte.  $\square$

Die im Beweis von Satz 3.3.2 beschriebene Transformation von einem ESVA soll noch einmal an einem Beispiel dargestellt werden. Der Zustand  $q_0$  des ESVAs  $A$  in Abbildung 3.5 besitzt drei ausgehende Transitionen, die Nachrichten vom Inputkanal  $a$  konsumieren und zwei ausgehende Transitionen, die Nachrichten vom Inputkanal  $b$  konsumieren.

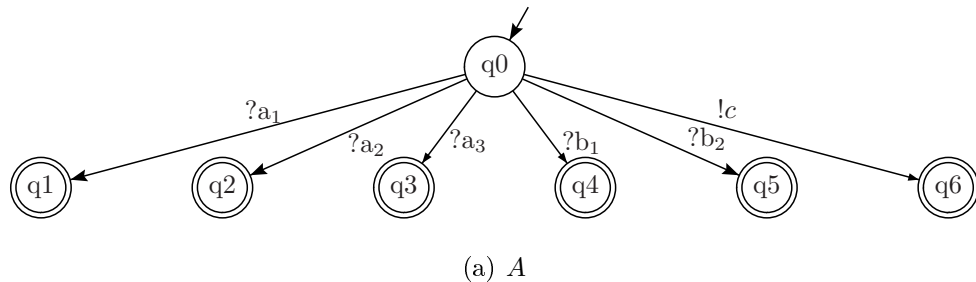


Abbildung 3.5: Ein ESVA  $A$  mit Empfangskonflikt.

Um Empfangskonflikte im aus  $A$  synthetisierten oWFN zu vermeiden, wird  $A$  in den ESVA  $A'$  transformiert. Der ESVA  $A'$  ist in Abbildung 3.6 dargestellt. Der einzige Zustand in  $A$ , der ausgehende Transitionen enthält, die Nachrichten vom gleichen Inputkanal konsumieren, ist der Zustand  $q_0$ . Dieser Zustand besitzt drei ausgehende Transitionen, die Nachrichten vom Inputkanal  $a$  konsumieren. Dies ist die höchste Anzahl von ausgehenden Transitionen dieses Zustands, die Nachrichten vom gleichen Inputkanal konsumieren.  $A'$  besitzt daher die beiden zusätzlichen Zustände  $q_0'$  und  $q_0''$ . Die eingefügten  $\tau$ -Transitionen sind  $(q_0'', \tau, q_0')$  und  $(q_0', \tau, q_0)$ . In  $A$  existieren keine Transitionen mit  $q_0$  als Zielzustand, daher existieren in  $A'$  keine Transitionen mit  $q_0''$  als Zielzustand. Der Zustand  $q_0''$  ist daher der Anfangszustand von  $A'$ . Die Transition mit der Beschriftung

$!c$  konsumiert keine Nachricht von einem Inputkanal und bleibt daher unverändert. Jede Transition mit einer der Beschriftungen  $?a_1$ ,  $?a_2$  und  $?a_3$  erhält einen eigenen Startzustand aus der Menge  $\{q_0, q_0', q_0''\}$ . Die Transitionen mit den Beschriftungen  $?b_1$  und  $?b_2$  erhalten jeweils einen eigenen Startzustand aus der Menge  $\{q_0, q_0'\}$ .

Im ESVA  $A'$  besitzt kein Zustand ausgehende Transitionen, die Nachrichten vom gleichen Inputkanal konsumieren. In einem aus  $A'$  synthetisierten oWFN  $N(A')$  kann somit kein Empfangskonflikt existieren.

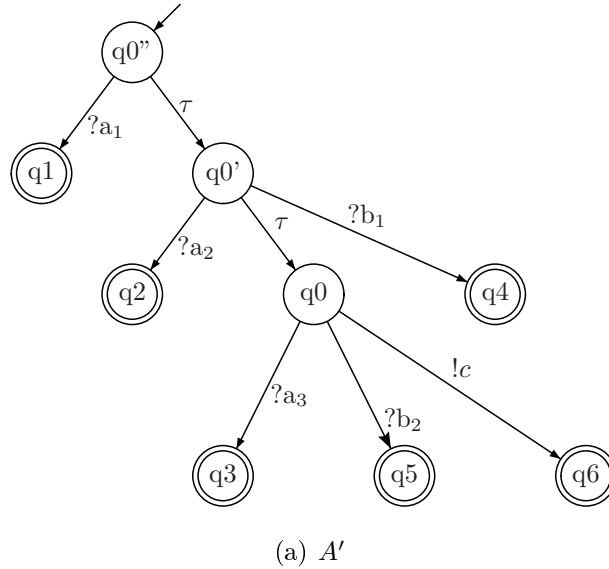


Abbildung 3.6: Ein aus dem ESVA  $A'$  synthetisiertes oWFN kann keinen Empfangskonflikt besitzen.

### 3.3.2 Soundness

In [Aal97] wurde für Workflownetze die *Soundness-Eigenschaft* definiert. Diese Eigenschaft wurde in [Kle07] für oWFNs angepasst und als eine Voraussetzung von oWFNs für die Übersetzung in WS-BPEL verlangt. Soundness stellt Anforderungen an die Struktur und das Verhalten eines oWFNs.

#### Definition 3.3.1 (Soundness)

Sei  $A = (Q, I, O, \delta, q_0, F)$  ein ESVA und  $N(A) = (P_{spn}, T_{spn}, F_{spn}, m_{0_{spn}})$  ein aus  $A$  synthetisiertes Petrietz mit der Menge aller erreichbaren Markierungen  $M$ . Das SPN  $N(A)$  erweitert zu einem oWFN  $N'(A) = (P, P_i, P_o, T, F, m_0, \Omega)$  mit  $m_{io} : P_i \cup P_o \rightarrow 0$  ist sound, wenn gilt:

1. *Es existiert ein Platz, der als einziger Platz eine Marke bei der Anfangsmarkierung besitzt:*  
*Es existiert ein  $p_0 \in P$ , mit  $m_0(p_0) = 1$  und  $m_0(p) = 0$ , für alle  $p \in P \setminus \{p_0\}$ .*
2. *Bei jeder Endmarkierung besitzt ein Platz genau eine oder keine Marke:*  
 *$m_f(p) \in \{0, 1\}$ , für alle  $m_f \in \Omega, p \in P$ .*
3. *Markierte Plätze einer Endmarkierung besitzen keine Transitionen im Nachbereich:*  
 *$p \bullet = \emptyset$ , falls ein  $m_f \in \Omega$  existiert mit  $m_f(p) = 1$ .*
4. *Von jeder erreichbaren Markierung in  $N(A)$ , existiert von der entsprechenden Markierung in  $N'(A)$  eine Schaltfolge zu einer Endmarkierung:*  
 *$(m_{spn} \oplus m_{io}) \xrightarrow{*} m_f, m_f \in \Omega$ , für alle  $m_{spn} \in M$ .*
5. *Wenn eine erreichbare Markierung in  $N(A)$  allen Plätzen einer Endmarkierung eine Marke zuordnet, dann ist diese Markierung die Endmarkierung selbst:*  
 *$(m_{spn} \oplus m_{io}) = m_f$ , falls  $(m_{spn} \oplus m_{io}) \geq m_f, m_f \in \Omega$ , für alle  $m_{spn} \in M$ .*
6. *Jede Transition in  $N(A)$  wird durch mindestens eine erreichbare Markierung aktiviert:*  
*Für alle  $t \in T$  existiert ein  $m_{spn}$  und  $m'_{spn}$ , mit  $m_{spn} \xrightarrow{t} m'_{spn}, m_{spn}, m'_{spn} \in M$ .*

Das aus dem ESVA  $A$  in Abbildung 3.7(a) synthetisierte oWFN  $N(A)$  in Abbildung 3.7(b) ist nicht sound. Die Anfangsmarkierung von  $N(A)$  ist  $m_0 = [p0, p1]$ , die erste Anforderung der Soundness Eigenschaft ist daher nicht erfüllt.  $N(A)$  besitzt die beiden Endmarkierungen  $m_{f1} = [p3]$  und  $m_{f2} = [p2, p4]$ . Der Platz  $p2$  besitzt die Transition  $\tau_3$  im Nachbereich. Die dritte Anforderung der Soundness Eigenschaft ist daher ebenfalls nicht erfüllt.

Jeder vollständig terminierungsfähige ESVA  $A$  (Abschnitt 2.3.1) kann so erweitert werden, dass das aus  $A$  synthetisierte oWFN  $N(A)$  sound ist. Diese Aussage wird in Satz 3.3.6 gezeigt. Bei dem Beweis dieses Satzes werden die folgenden drei Lemmata verwendet.

**Lemma 3.3.3**

*Sei  $A = (Q, I, O, \delta, q_0, F_A)$  ein ESVA und  $N(A) = (P, T, F, m_0)$  das aus  $A$  synthetisierte Petrinetz. Da der Erreichbarkeitsgraph von  $N(A)$  bisimilar zu  $A$  ist, entspricht jede erreichbare Markierung  $m$  von  $N(A)$  einem Zustand  $q_m$  von  $A$ . Bei einer erreichbaren Markierung von  $N(A)$  befinden sich genau auf den Plätzen von  $N(A)$  eine Marke, die einer Region entsprechen, in der  $q_m$  enthalten ist.*

**Beweis.** Per Induktion über die erreichbaren Markierungen  $m_i$  der Schaltfolgen  $m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \dots, m_{n-1} \xrightarrow{t_n} m_n$  von  $N(A)$  mit  $n \in \mathbb{N}, 0 \leq i \leq n$  und  $t_i \in T$ . Der Zustand  $q_{m_i} \in Q$  bezeichne den zu  $m_i$  bisimilaren Zustand.

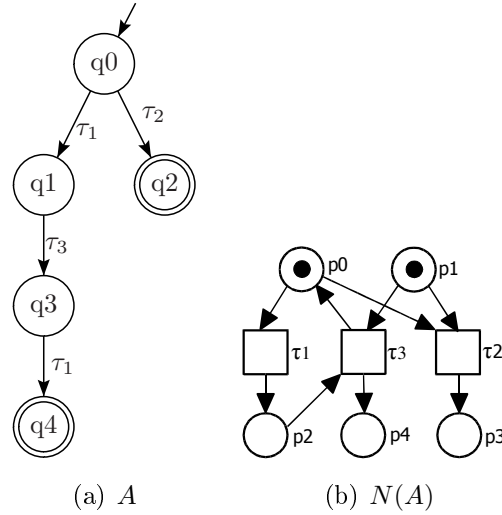


Abbildung 3.7: Das aus dem ESVA  $A$  (a) synthetisierte oWFN  $N(A)$  ist nicht sound.

Induktionsanfang:  $i = 0$

Bei der Anfangsmarkierung  $m_0$  besitzen wegen Definition 2.4.6 genau die Plätze eine Marke, die Regionen entsprechen, in denen der Anfangszustand  $q_0$  enthalten ist. Der Zustand  $q_{m_0}$  entspricht immer dem Anfangszustand  $q_0$ .

Induktionsschritt:  $i \rightarrow i + 1$

Nach der Induktionsbehauptung besitzen bei der Markierung  $m_i$  genau die Plätze eine Marke, die einer Region entsprechen, in der  $q_{m_i}$  enthalten ist. Durch das Schalten der Transition  $t_{i+1}$  wird die Markierung  $m_{i+1}$  erreicht. Diese Markierung entspricht dem Zustand  $q_{m_{i+1}}$  von  $A$ . Sei  $p \in P$  ein Platz und  $r_p$  eine Region von  $A$ , die dem Platz  $p$  entspricht. Fallunterscheidung für  $q_{m_i}$  und  $q_{m_{i+1}}$ :

- $q_{m_i} \notin r_p$  und  $q_{m_{i+1}} \notin r_p$   
In diesem Fall ist  $r_p$  weder eine Vorregion, noch eine Nachregion von  $\lambda(t_{i+1})$ , daher ist  $m_{i+1}(p) = m_i(p)$ . Aufgrund der Induktionsbehauptung und  $q_{m_i} \notin r_p$ , gilt  $m_i(p) = 0$  und somit ist  $m_{i+1}(p) = 0$ .
- $q_{m_i} \in r_p$  und  $q_{m_{i+1}} \notin r_p$   
In diesem Fall ist  $r_p$  eine Vorregion von  $\lambda(t_{i+1})$ , daher ist  $m_{i+1}(p) = m_i(p) - 1$ . Aufgrund der Induktionsbehauptung und  $q_{m_i} \in r_p$  ist  $m_i(p) = 1$  und somit ist  $m_{i+1}(p) = 0$ .
- $q_{m_i} \notin r_p$  und  $q_{m_{i+1}} \in r_p$   
In diesem Fall ist  $r_p$  eine Nachregion von  $\lambda(t_{i+1})$ , daher ist  $m_{i+1}(p) = m_i(p) + 1$ . Aufgrund der Induktionsbehauptung und  $q_{m_i} \notin r_p$  ist  $m_i(p) = 0$  und somit ist  $m_{i+1}(p) = 1$ .

- $q_{m_i} \in r_p$  und  $q_{m_{i+1}} \in r_p$   
 In diesem Fall ist  $r_p$  weder eine Vorregion, noch eine Nachregion von  $\lambda(t_{i+1})$ , daher ist  $m_{i+1}(p) = m_i(p)$ . Aufgrund der Induktionsbehauptung und  $q_{m_i} \in r_p$ , gilt  $m_i(p) = 1$  und somit ist  $m_{i+1}(p) = 1$ .

Es gilt also  $m_{i+1}(p) = 0$ , falls  $q_{m_{i+1}} \notin r_p$  und  $m_{i+1}(p) = 1$ , falls  $q_{m_{i+1}} \in r_p$ . □

Sei  $N(A)$  ein oWFN, das aus einem ESVA  $A$  synthetisiert wurde mit einem Anfangszustand  $q_0$  und genau einem Endzustand  $q_f$ . Falls jede Transition von  $A$  mit  $q_0$  als Startzustand bzw. mit  $q_f$  als Zielzustand eine Beschriftung besitzt mit der keine andere Transition von  $N(A)$  beschriftet ist, dann ist  $N(A)$  sound.  $N(A)$  besitzt dann zudem genau eine Anfangsmarkierung und genau eine Endmarkierung mit jeweils genau einer Marke auf einem Platz.

**Lemma 3.3.4**

Sei  $A = (Q, I, O, \delta, q_0, q_f)$  ein ESVA, so dass für alle  $q_i \in Q$ ,  $i \in \mathbb{N}$  gilt:

- wenn  $\lambda((q_0, x, q_1)) = \lambda((q_2, x, q_3))$ , dann ist  $q_0 = q_2$  und  $q_1 = q_3$ ,
- wenn  $\lambda((q_4, x, q_0)) = \lambda((q_5, x, q_6))$ , dann ist  $q_4 = q_5$  und  $q_0 = q_6$ ,
- wenn  $\lambda((q_7, x, q_f)) = \lambda((q_8, x, q_9))$ , dann ist  $q_7 = q_8$  und  $q_f = q_9$ .

Dann besitzt das aus  $N(A) = (P_{spn}, T_{spn}, F_{spn}, m_{0_{spn}})$  erweiterte synthetisierte oWFN  $N'(A) = (P, P_i, P_o, T, F, m_0, \Omega)$  mit  $m_{io} : P_i \cup P_o \rightarrow 0$  genau eine Anfangsmarkierung und genau eine Endmarkierung mit jeweils einer Marke auf genau einem Platz und keiner Marke auf allen anderen Plätzen.

**Beweis.** Alle Transitionen mit  $q_0$  als Startzustand bzw. Endzustand besitzen aufgrund der Voraussetzung eine genau einmal verwendete Beschriftung. Die Menge  $\{q_0\}$  ist daher eine minimale nicht triviale Region von  $A$ . Es kann somit keine andere Region geben, die  $q_0$  enthält. Die inneren Plätze von  $N'(A)$  entsprechen den minimalen nicht trivialen Regionen von  $A$ . Sei  $p_0$  der Platz von  $N'(A)$ , der der Region  $\{q_0\}$  entspricht. Bei der Anfangsmarkierung erhalten alle Plätze eine Marke, die einer Region entsprechen, in der  $q_0$  enthalten ist. Da nur die Region  $\{q_0\}$  diesen Zustand enthält, besitzt bei der Anfangsmarkierung von  $N'(A)$  nur der Platz  $p_0$  eine Marke.

Der Endzustand  $q_f$  ist wegen Definition 2.3.1 kein Startzustand einer Transition von  $A$  und alle Transitionen mit  $q_f$  als Endzustand besitzen aufgrund der Voraussetzung eine einmalige Beschriftung. Die Menge  $\{q_f\}$  ist daher eine minimale nicht triviale Region von  $A$ . Sei  $p_f$  der Platz von  $N(A)$ , der der Region  $\{q_f\}$  entspricht. Die dem Zustand  $q_f$  entsprechende Markierung besitzt wegen Lemma 3.3.3 eine Marke auf  $p_f$  und auf allen anderen Plätzen keine Marke. Diese Markierung sei  $m_f$ . Da in  $A$  keine Transitionen mit  $q_f$  als Startzustand existieren, besitzt  $p_f$  keine Transitionen im Nachbereich. Bei

$m_f$  kann somit keine Transition aktiviert sein. Die Markierung  $m_f \oplus m_{io}$  ist folglich eine Endmarkierung von  $N'(A)$ . Es bleibt zu zeigen, daß  $m_f \oplus m_{io}$  die einzige Endmarkierung von  $N'(A)$  ist. In  $A$  muss von jedem Zustand der Zustand  $q_f$  erreichbar sein. Aufgrund der Bisimilarität zwischen  $A$  und dem Erreichbarkeitsgraph von  $N(A)$  muss von jeder erreichbaren Markierung  $m$  von  $N(A)$  die Markierung  $m_f$  erreichbar sein. Gilt  $m \neq m_f$ , dann muss bei  $m$  mindestens eine Transition aktiviert sein, damit eine Schaltfolge zu  $m_f$  existiert. Die Markierung  $m \oplus m_{io}$  kann daher keine Endmarkierung von  $N'(A)$  sein.  $\square$

**Lemma 3.3.5**

Sei  $A = (Q, I, O, \delta, q_0, q_f)$  ein vollständig terminierungsfähiger ESVA und  $N(A)$  das aus  $A$  synthetisierte Petrinetz. Für alle  $q_i \in Q$ ,  $i \in \mathbb{N}$  sei:

- wenn  $\lambda((q_0, x, q_1)) = \lambda((q_2, x, q_3))$ , dann ist  $q_0 = q_2$  und  $q_1 = q_3$ ,
- wenn  $\lambda((q_4, x, q_0)) = \lambda((q_5, x, q_6))$ , dann ist  $q_4 = q_5$  und  $q_0 = q_6$ ,
- wenn  $\lambda((q_7, x, q_f)) = \lambda((q_8, x, q_9))$ , dann ist  $q_7 = q_8$  und  $q_f = q_9$ .

Dann ist das aus  $A$  synthetisierte oWFN  $N'(A)$  sound.

**Beweis.** Die ersten beiden Anforderungen und die fünfte Anforderung der Soundness-Eigenschaft folgen unmittelbar aus Lemma 3.3.4. Die dritte Anforderung wurde bereits innerhalb des Beweises von Lemma 3.3.4 bewiesen. Die vierte Anforderung folgt aus der Bisimilarität zwischen  $A$  und dem Erreichbarkeitsgraph von  $N(A)$  und der Terminierungsfähigkeit von  $A$ . Die sechste Anforderung folgt aus der Bisimilarität zwischen  $A$  und dem Erreichbarkeitsgraph von  $N(A)$  und der Definition von synthetisierten Petri-Netzen (Definition 2.4.6).  $\square$

**Satz 3.3.6**

Sei  $N(A)$  ein aus dem vollständig terminierungsfähigen ESVA  $A = (Q, I, O, \delta, q_0, \Omega)$  synthetisiertes oWFN, das die Soundness Eigenschaft nicht erfüllt.  $A$  kann so umgeformt werden, dass  $N(A)$  sound ist.

**Beweis.** Besitzt  $A$  mindestens zwei Transitionen mit einer identischen Beschriftung und eine der beiden Transitionen besitzt  $q_0$  als Startzustand, dann erhält  $A$  einen neuen Anfangszustand  $q'_0$  und eine Transition mit einer genau einmal verwendeten Beschriftung von  $q'_0$  nach  $q_0$ . Besitzt  $A$  mehr als einen Endzustand, oder mindestens zwei Transitionen mit einer identischen Beschriftung und eine der beiden Transitionen besitzt einen Endzustand als Zielzustand, dann erhält  $A$  einen neuen Endzustand  $q_f$  und von jedem ursprünglichen Endzustand eine Transition mit einmaliger Beschriftung und  $q_f$  als Zielzustand. Wegen Lemma 3.3.5 ist das aus  $A$  synthetisierte oWFN  $N(A)$  sound.  $\square$

Abbildung 3.8 (a) zeigt den ESVA  $A'$ , der aus dem ESVA  $A$  in Abbildung 3.7 (a) entstanden ist.  $A$  besitzt zwei Transitionen mit der Beschriftung  $\tau_1$ . Eine dieser beiden Transitionen besitzt den Anfangszustand  $q_0$  als Startzustand.  $A'$  besitzt einen neuen

Startzustand  $q0'$  und eine neue Transition von  $q0'$  zum alten Startzustand  $q0$ . Diese Transition besitzt in  $A'$  mit  $\tau_0$  eine Beschriftung, die keine andere Transition in  $A'$  besitzt. Der ESVA  $A$  besitzt die beiden Endzustände  $q2$  und  $q4$ .  $A'$  besitzt den neuen Endzustand  $q5$ . Von den ursprünglichen Endzuständen führt jeweils eine neue Transition zu dem neuen Endzustand. Das aus  $A'$  synthetisierte oWFN  $N'(A')$  in Abbildung 3.8 (b) ist sound.

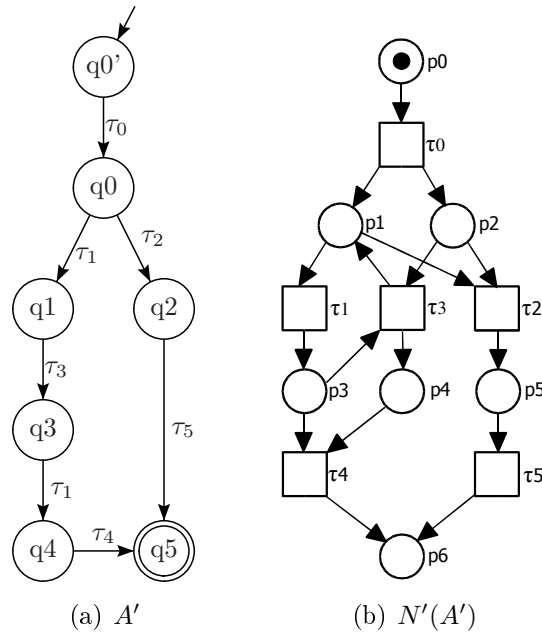


Abbildung 3.8: Das aus dem ESVA  $A'$  (a) synthetisierte oWFN  $N'(A')$  ist sound.

### 3.3.3 Zyklen in offenen Workflownetzen

Ein Zyklus in einem oWFN besteht aus einer Menge von Plätzen und Transitionen. Nur wenn jeder Zyklus bestimmte Eigenschaften erfüllt, dann kann ein oWFN in WS-BPEL übersetzt werden [Kle07]. Bestimmte Strukturen in einem ESVA  $A$  führen bei der Synthese in ein oWFN  $N(A)$  zu Zyklen, die nicht diese notwendigen Eigenschaften besitzen. Durch das Aufsplitten bestimmter Beschriftungen von Transitionen in  $A$  können unerlaubte Zyklen in  $N(A)$  verhindert werden.

#### Definition 3.3.2 (Zyklus in einem offenen Workflownetz)

Sei  $N = (P, P_i, P_o, T, F, m_0, \Omega)$  ein oWFN. Eine Menge von Plätzen und Transitionen  $\{k_0, k_1, \dots, k_n\}, n \in \mathbb{N}^+, k \in P \cup T$ , bildet einen Zyklus in  $N$ , wenn für alle  $k_i$  mit  $i > 0$  gilt  $k_i \in k_{i-1} \bullet$  und für  $k_0$  gilt  $k_0 \in k_n \bullet$ .

Im oWFN  $N(A)$  in Abbildung 3.9(a) bilden die beiden Plätze  $p1$  und  $p3$ , sowie die beiden mit  $!c$  und  $?b$  Transitionen einen Zyklus  $Z$ . Gehört eine Transition zu einem Zyklus, darf diese Transition keine inneren Plätze im Vor- oder Nachbereich besitzen, die sich außerhalb von diesem Zyklus befinden. Die Transition mit der Beschriftung  $!c$  besitzt den Platz  $p2$  im Vorbereich und den Platz  $p4$  im Nachbereich. Beide Plätze sind innere Plätze und gehören nicht zu  $Z$ , daher besitzt  $Z$  nicht die geforderte Eigenschaft.  $N(A)$  ist das synthetisierte oWFN zu dem SVA  $A$  in Abbildung 3.9(b). Der SVA  $A'$  in Abbildung 3.9(c) ist aus  $A$  durch das Aufsplitten der Beschriftung  $?b$  in die Beschriftungen  $?b_1$  und  $?b_2$  entstanden. Das aus  $A'$  synthetisierte oWFN  $N'(A')$  in Abbildung 3.9(d) enthält keine unerlaubten Zyklen.

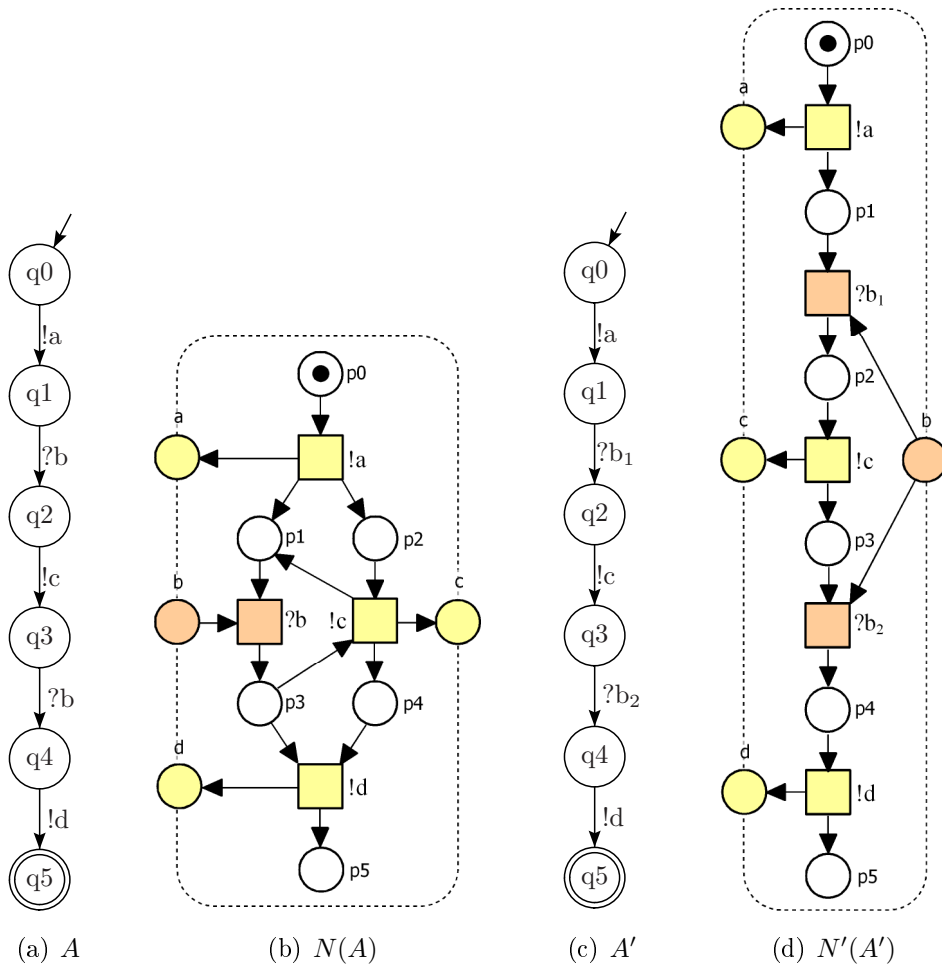


Abbildung 3.9: Vermeidung von unerlaubten Zyklen.

Unerlaubte Zyklen können bei der Synthese aus einem SVA durch das Aufsplitten von Beschriftungen immer verhindert werden. Der folgende Satz 3.3.7 besagt, dass wenn jede Transition eines SVAs  $A$  eine eigene Beschriftung besitzt, im aus  $A$  synthetisierten oWFN  $N(A)$  keine unerlaubten Zyklen vorhanden sind, da alle Plätze im Vor- und Nachbereich

einer Transition, die zu einem Zyklus gehört, ebenfalls zu diesem Zyklus gehören müssen.

**Satz 3.3.7**

Sei  $A = (Q, I, O, \delta, q_0, \Omega)$  ein ESVA mit  $t_1 = t_2$ , falls  $\lambda(t_1) = \lambda(t_2)$ , für alle  $t_1, t_2 \in T$  und  $N(A) = (P, P_i, P_o, T, F, m_0, \Omega)$  ein aus  $A$  synthetisiertes oWFN. Gehört eine Transition von  $N(A)$  zu einem Zyklus, dann gehören alle inneren Plätze im Vor- und Nachbereich dieser Transition auch zu dem Zyklus.

**Beweis.** Da jede Transition in  $A$  eine einmalige Beschriftung besitzt, bildet jeder Zustand eine minimale Region von  $A$  [CKLY98]. Jede Transition von  $N$  besitzt genau einen inneren Platz im Vorbereich und einen inneren Platz im Nachbereich, da jede Beschriftung in  $A$  einmalig ist und aus diesem Grund genau eine Vor- und eine Nachregion besitzt. Gehört eine Transition zu einem Zyklus, müssen somit auch der Platz im Vorbereich und der Platz im Nachbereich zu dem Zyklus gehören.  $\square$

Nur wenn die Zyklen in einem, aus einem SVA  $A$  synthetisierten, oWFN  $N(A)$  eine bestimmte Struktur besitzen, kann  $N(A)$  nach WS-BPEL übersetzt werden. Durch das Aufsplitten von Beschriftungen der Transitionen von  $A$ , ist die Vermeidung von unerlaubten Zyklen in  $N(A)$  immer möglich. Es müssen jedoch nicht immer alle Beschriftungen aufgesplittet werden um unerlaubte Zyklen zu vermeiden. Ein Beispiel hierfür ist das oWFN  $N'(A')$  in Abbildung 3.2 das aus dem ESVA  $A'$  in Abbildung 2.10 entstanden ist.

## 3.4 Verhaltensanalyse transformierter Serviceautomaten

Das Ziel dieser Arbeit ist es, aus einem SVA ein oWFN zu synthetisieren, das nach WS-BPEL übersetzt werden kann. Ein SVA muss elementar sein um in ein oWFN synthetisiert zu werden. Hierfür muss der SVA die in Abschnitt 2.4.2 definierten Axiome erfüllen. Für die Übersetzung nach WS-BPEL darf ein oWFN keine Empfangskonflikte besitzen (vgl. Abschnitt 3.3.1), muss sound sein (vgl. Abschnitt 3.3.2) und darf keine unerlaubten Zyklen besitzen (vgl. Abschnitt 3.3.3). In den vorherigen Abschnitten und Kapiteln wurden Transformationen für SVAs beschrieben, um diese Eigenschaften zu gewährleisten. In diesem Abschnitt sollen die Auswirkungen der Transformationen auf das Verhalten eines SVAs untersucht werden.

Die verwendeten Transformationen lassen sich in zwei verschiedene Arten von Transformationen einteilen. Erstens in das Aufsplitten von Transitionsbeschriftungen und zweitens in das Einfügen von internen Schritten.

Das Aufsplitten von Transitionsbeschriftungen wird verwendet um einen SVA zu transformieren bei dem das vierte oder fünfte Axiom eines elementaren SVAs nicht erfüllt

ist, oder um unerlaubte Zyklen im synthetisierten oWFN zu vermeiden. Ein SVA wird durch das Aufsplitten von Transitionsbeschriftungen in einen splitmorphen SVA (vgl. Abschnitt 2.4.2) transformiert. Da sich eine solche Transformation lediglich auf die Beschriftung einer Transition auswirkt und deren Aktion dagegen unverändert läßt, wird das Verhalten eines SVAs hierdurch nicht verändert.

Bei dem Einfügen eines internen Schrittes erhält ein SVA einen neuen Zustand und eine mit  $\tau$ -beschriftete Transition, die den neuen Zustand als Startzustand oder Zielzustand besitzt. Diese Transformation wird verwendet um das erste Axiom eines elementaren SVAs zu erfüllen und um in dem, aus dem SVA synthetisierten, oWFN die Soundness-Eigenschaft zu gewährleisten und Empfangskonflikte zu verhindern.

Das erste Axiom eines elementaren SVAs verbietet Selbstschleifen. Angenommen ein SVA  $A$  besitzt eine Transition mit dem gleichen Start- und Zielzustand  $q$ . Dann wird in  $A$  ein neuer Zustand  $q'$  eingefügt, der zum neuen Zielzustand dieser Transition wird (vgl. Abschnitt 3.1). Zudem erhält  $A$  einen neuen internen Schritt mit dem neuen Zustand  $q'$  als Startzustand und  $q$  als Zielzustand. Das Verhalten des transformierten SVAs bleibt gleich mit dem einzigen Unterschied, das auf eine Transition, die vorher eine Selbstschleife gebildet hat, ein interner Schritt folgt. Dieser interne Schritt führt dann wieder in den Zustand, zu dem ursprünglich die Selbstschleife führte.

Muss ein SVA  $A$  transformiert werden, damit das aus  $A$  synthetisierte oWFN sound ist, werden ebenfalls neue Zustände und interne Schritte in  $A$  eingefügt. Diese Transformationen betreffen entweder den Anfangszustand oder die Endzustände von  $A$ . In bestimmten Fällen wird ein neuer Anfangszustand eingefügt, von dem ein interner Schritt zu dem ursprünglichen Anfangszustand führt. Ebenso ist es möglich, dass ein neuer Endzustand in  $A$  eingefügt wird, zu dem interne Schritte von den ursprünglichen Endzuständen führen. Das Verhalten von  $A$  zwischen dem ursprünglichen Anfangszustand und den ursprünglichen Endzuständen bleibt bei diesen Transformationen demnach unverändert.

Der letzte Grund für das Einfügen interner Schritte in einen SVA  $A$  ist das Verhindern von Empfangskonflikten in dem aus  $A$  synthetisierten oWFN. Die notwendigen Transformationen von  $A$  bestehen im Einfügen von neuen Zuständen und internen Schritten. Eine formale Analyse der Verhaltensänderung durch diese Transformationen soll im Rahmen dieser Arbeit nicht erfolgen. Es sollen jedoch ohne Beweis zwei Beobachtungen gemacht werden. Erstens werden bei diesen Transformationen lediglich interne Schritte in den SVA eingefügt. Zweitens sind in jedem Zustand, der nicht neu hinzugefügt wurde, genau die Aktionen ausführbar, die auch vor der Transformation ausführbar waren.



# 4 Implementierung der Synthese offener Workflownetze

Ein Prozess in der Web Services Business Process Execution Language (WS-BPEL) kann in ein formales Modell in Form eines offenen Workflownetzes (oWFN)  $N$  überführt werden [LMSW06]. Sofern  $N$  bedienbar ist, kann ein Interaktionsgraph (IG) berechnet werden, der das Verhalten eines Partners von  $N$  beschreibt [Wei06]. Ein IG kann als Serviceautomat (SVA) interpretiert werden, bei dem jeder Zustand einem Knoten des IGs entspricht. Im Rahmen dieser Arbeit wurde gezeigt wie zu einem SVA, der bestimmte Eigenschaften erfüllt, ein synthetisiertes oWFN erstellt werden kann, welches alle notwendigen Eigenschaften besitzt um in WS-BPEL übersetzt werden zu können. Die Berechnung synthetisierter oWFNs aus SVAs wurde in dem Tool SVA2oWFN umgesetzt.

Anhand eines Beispiels wird in diesem Kapitel die Arbeitsweise dieses Tools dargestellt. Als Beispiel dient der Interaktionsgraph aus Abbildung 2.8, der noch einmal in Abbildung 4.1 dargestellt ist. Aus Gründen der Übersichtlichkeit sind die Knoten mit  $v_0$  bis  $v_7$  benannt und nicht wie in Abbildung 2.8 die entsprechenden Markierungsmengen der einzelnen Knoten dargestellt.

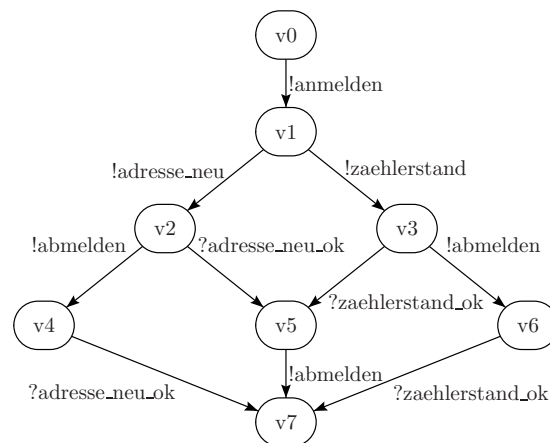


Abbildung 4.1: Interaktionsgraph  $I$  des oWFNs  $N$  aus Abbildung 2.4(a).

## 4.1 Das Tool SVA2oWFN

Die Synthese offener Workflownetze aus Serviceautomaten und die Zusicherung der notwendigen Eigenschaften für die Übersetzung in WS-BPEL wurde im Rahmen dieser Arbeit in dem Tool SVA2oWFN implementiert. Die Eingabe des Tools ist ein SVA  $A$  und berechnet wird ein synthetisiertes oWFN  $N(A)$ . Das Tool überprüft alle geforderten Eigenschaften und führt die notwendigen Transformationen von  $A$  aus. Im Folgenden sind alle geforderten Eigenschaften in der Reihenfolge der Bearbeitung aufgelistet:

1.  $A$  muss vollständig terminierungsfähig sein.
2.  $N(A)$  darf keinen Empfangskonflikt besitzen.
3.  $N(A)$  muss sound sein.
4.  $N(A)$  darf keine unerlaubten Zyklen besitzen.
5.  $A$  muss elementar sein.

In dieser Reihenfolge der einzelnen Schritte bleiben die zugesicherten Eigenschaften eines Schrittes auch nach Änderungen an  $A$  durch die nachfolgenden Schritte bestehen. Ist  $A$  beispielsweise vollständig terminierungsfähig, bleibt diese Eigenschaft bei allen Änderungen, die aus den späteren Schritten resultieren können, erhalten.

Im ersten Schritt wird die vollständige Terminierungsfähigkeit von  $A$  überprüft. Die vollständige Terminierungsfähigkeit wird als gegeben vorausgesetzt. Ist  $A$  nicht vollständig terminierungsfähig, bricht das Tool mit einer entsprechenden Meldung ab.

Der zweite Schritt besteht darin, Empfangskonflikte in  $N(A)$  zu vermeiden. Ein Empfangskonflikt kann nur auftreten, wenn in  $A$  ein Zustand  $q$  mit zwei ausgehenden Transitionen existiert, die Nachrichten vom gleichen Inputkanal  $x$  konsumieren. Besitzt  $q$  keine ausgehende Transition, die Nachrichten von einem anderen Inputkanal als  $x$  konsumiert, wird der Empfangskonflikt durch eine Transformation von  $A$ , wie in Abschnitt 3.3.1 beschrieben, verhindert. Besitzt  $q$  dagegen eine solche Transition, kann der Empfangskonflikt nicht verhindert werden und das Tool bricht mit einer entsprechenden Meldung ab.

Das synthetisierte oWFN muss sound sein. Dies wird im dritten Schritt zugesichert. Entscheidend ist der Anfangszustand und die Endzustände von  $A$ . Jeder dieser Zustände darf nur Start- oder Zielzustand von Transitionen sein, deren Beschriftung in  $A$  genau einmal verwendet wird. Zudem darf  $A$  nur einen Endzustand besitzen.  $N(A)$  ist immer sound, wenn diese Eigenschaften erfüllt sind. Ist eine dieser Eigenschaften dagegen nicht erfüllt, wird  $A$  entsprechend Abschnitt 3.3.2 angepasst.

Im vierten Schritt werden unerlaubte Zyklen in  $N(A)$  verhindert. Jede Transition in einem Zyklus darf keine Plätze im Vor- und Nachbereich besitzen, die nicht zum Zyklus gehören. Zyklen mit dieser Eigenschaft können, durch die in Abschnitt 3.3.3 beschriebenen Transformationen von  $A$ , immer verhindert werden.

Im fünften und letzten Schritt wird  $A$ , sofern  $A$  nicht bereits elementar ist, in einen elementaren SVA transformiert. Ist das Aufsplitten von Transitionsbeschriftungen nötig, wird hierbei die in [CKLY98] beschriebene Heuristik verwendet. Bei dieser Heuristik wird immer die Beschriftung einer Transition gesplittet, die für die meisten momentanen Konflikte verantwortlich ist.

Sind die einzelnen Schritte erfolgreich durchlaufen worden, kann das synthetisierte oWFN  $N(A)$  berechnet und ausgegeben werden.  $N(A)$  besitzt dann alle in [Kle07] geforderten Eigenschaften für die Übersetzung in WS-BPEL.

## 4.2 Ein Beispiel für die Synthese eines offenen Workflownetzes

Die Synthese offener Workflownetze beruht im Rahmen dieser Arbeit auf Serviceautomaten. Um zu einem IG  $I$  ein offenes Workflownetz zu synthetisieren, muss  $I$  somit als ein SVA  $A$  interpretiert werden. Hierbei entspricht jeder Knoten von  $I$  einem Zustand von  $A$ . Der Wurzelknoten entspricht dem Anfangszustand und jeder Knoten ohne ausgehende Kanten ist ein Endzustand von  $A$ .

Der SVA  $A$  in Abbildung 4.2 entspricht dem IG  $I$  in Abbildung 4.1. Für jede Kante von  $I$  existiert in  $A$  eine Transition mit identischer Beschriftung. Für die Kante mit der Beschriftung `!anmelden` zwischen den Knoten `v0` und `v1` besitzt  $A$  eine identisch beschriftete Transition zwischen den Zuständen `q0` und `q1`. Die Knoten `v0` bis `v7` werden in  $A$  zu den Zuständen `q0` bis `q7`. Der Wurzelknoten `v0` von  $I$  entspricht dem Anfangszustand `q0` von  $A$ . In  $I$  gibt es mit `v7` genau einen Knoten ohne ausgehenden Kanten und dieser entspricht dem Endzustand `q7` von  $A$ .

Die einzelnen Schritte des Tools `SVA2oWFN` wurden für den SVA  $A$  aus Abbildung 4.2 ausgeführt. Den daraus resultierenden SVA  $A'$  zeigt Abbildung 4.3.  $A'$  besitzt im Vergleich zu  $A$  den neuen Endzustand `q8` und eine neue Transition mit der Beschriftung `τ`. Transitionsbeschriftungen von  $A$  mussten nicht aufgesplittet werden.

Der SVA  $A$  in Abbildung 4.2 ist vollständig terminierungsfähig, da von jedem Zustand der Endzustand `q7` erreichbar ist. In  $A$  existiert kein Zustand mit zwei ausgehenden Transitionen mit einer identischen Beschriftung. Es können daher in  $N(A')$  keine Empfangskonflikte entstehen. Der SVA  $A$  in Abbildung 4.2 ist nicht sound, da der Endzustand `q7` Zielzustand einer Transition mit der Beschriftung `!abmelden` ist und diese Beschriftung

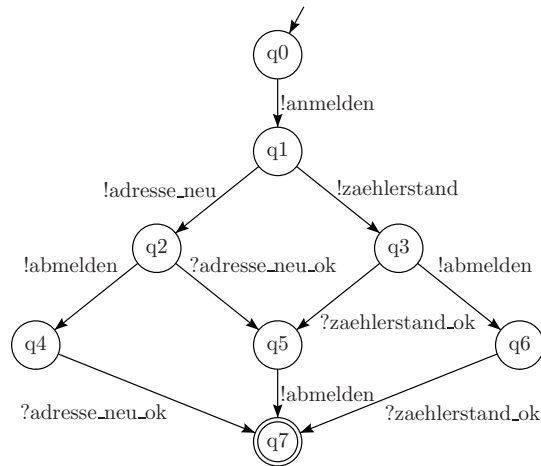


Abbildung 4.2: Der dem IG aus Abbildung 4.1 entsprechende SVA  $A$ .

ung mehr als einmal verwendet wird. Der SVA  $A'$  besitzt daher den neuen Endzustand  $q8$  und eine neue Transition von  $q7$  nach  $q8$  mit der Beschriftung  $\tau$ . Unerlaubte Zyklen können in  $N(A')$  nicht entstehen und  $A'$  ist elementar.

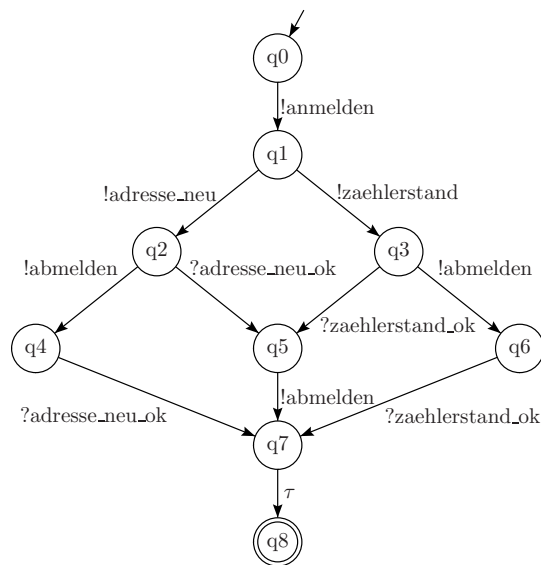


Abbildung 4.3: Der aus dem SVA  $A$  in Abbildung 4.2 entstandene SVA  $A'$ .

Ein SVA besitzt nach dem erfolgreichen Ausführen der einzelnen Schritte des Tools alle notwendigen Eigenschaften für die Synthese in ein oWFN. Nach der Ausführung dieser Schritte wird von dem Tool das entsprechende synthetisierte oWFN berechnet und erstellt.

Das aus  $A'$  synthetisierte oWFN  $N(A')$  ist in Abbildung 4.4 dargestellt. Es besitzt alle in [Kle07] geforderten Eigenschaften für die Übersetzung in WS-BPEL.  $N(A')$  hat die Anfangsmarkierung  $m_0 = [q_0]$  und besitzt mit  $m_f = [p_7]$  genau eine Endmarkierung.

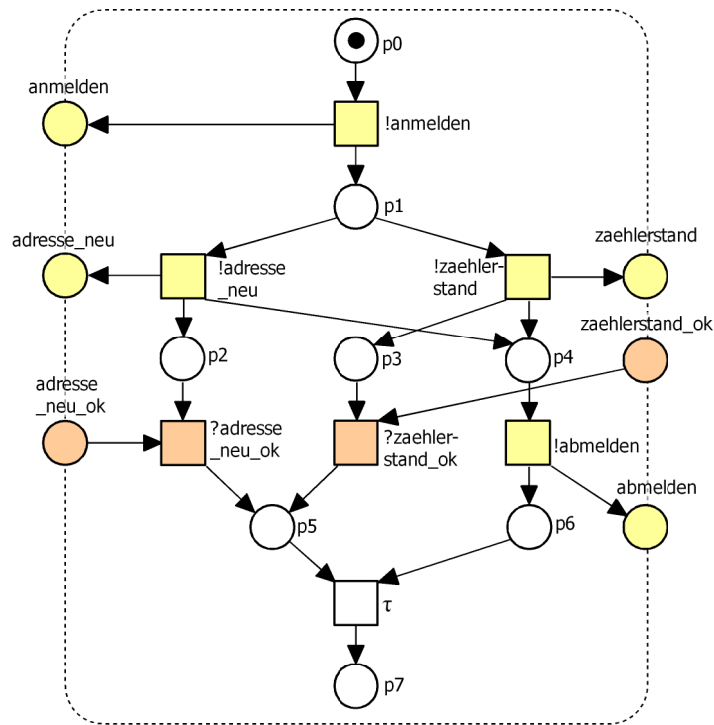


Abbildung 4.4: Das aus dem SVA  $A$  in Abbildung 4.3 synthetisierte oWFN  $N(A')$ .



# 5 Zusammenfassung und Ausblick

Dieses Kapitel liefert eine Zusammenfassung und einen Ausblick auf Grundlage der Ergebnisse dieser Arbeit. Der erste Abschnitt beschreibt die erreichten Ergebnisse, insbesondere die Synthese offener Workflownetze aus Serviceautomaten und die offenen Punkte dieser Arbeit. Der zweite Abschnitt beleuchtet in einem Ausblick den Nutzen, der aus den Ergebnissen dieser Arbeit resultiert.

## 5.1 Zusammenfassung

Eine formale Übersetzung von SVAs in oWFNs existierte bisher nicht. Das erste Ziel dieser Arbeit war es, eine solche Übersetzung formal zu definieren. Hierfür wurde im dritten Kapitel das synthetisierte oWFN definiert und gezeigt, wie ein solches oWFN aus einem SVA berechnet werden kann. Sofern es sich bei dem SVA um ein elementaren SVA handelt, ist die Übersetzung in ein synthetisiertes oWFN immer möglich.

Ein SVA kann nur in ein oWFN übersetzt werden, wenn es sich um einen elementaren SVA handelt. Ein SVA ist elementar, wenn es bestimmte Eigenschaften erfüllt. Jeder SVA, bei dem jeder Zustand vom Anfangszustand aus erreichbar ist, kann in einen elementaren SVA transformiert werden. Dies wurde ebenso im dritten Kapitel gezeigt, wie auch die notwendigen Schritte für die Transformation eines SVAs in ein oWFN.

Das zweite Ziel dieser Arbeit war es, dass ein synthetisiertes oWFN alle Eigenschaften für die Übersetzung in WS-BPEL besitzt. Bei diesen Eigenschaften handelt es sich um vermiedene Empfangskonflikte, die Soundness-Eigenschaft und um das Verhindern von unerlaubten Zyklen. Bestimmte Strukturen im SVA führen bei der Synthese zu unerlaubten Zyklen im resultierenden oWFN. In Abschnitt 3.3.3 wurde gezeigt, dass unerlaubte Zyklen in oWFNs immer verhindert werden können. Eine genauere Beschreibung der Strukturen von SVAs, die im oWFN zu unerlaubten Zyklen führen, erfolgte im Rahmen dieser Arbeit nicht.

Ein weiterer Punkt, der noch genauer untersucht werden sollte, ist die Verhaltensänderung eines SVAs durch die beschriebenen Transformationsschritte. Eine Beschreibung dieser Änderungen erfolgte im Abschnitt 3.4. Was noch fehlt ist eine formale Analyse dieser Änderungen um mit Sicherheit die Frage beantworten zu können, ob und wenn

ja welche der beschriebenen Transformationen zu unerwünschten Verhaltensänderungen führen können.

Die Berechnung eines synthetisierten oWFNs aus einem SVA wurde, im Rahmen dieser Arbeit, in dem Tool SVA2oWFN umgesetzt. Dieses Tool erhält als Eingabe einen SVA und berechnet als Ausgabe ein synthetisiertes oWFN. Handelt es sich bei der Eingabe nicht um einen elementaren SVA, wird dieser SVA zunächst in einen elementaren SVA transformiert. Im zweiten Schritt werden die notwendigen Transformationen durchgeführt, damit das synthetisierte oWFN in WS-BPEL übersetzt werden kann. Im dritten Schritt wird das synthetisierte oWFN berechnet und ausgegeben. Die Beschreibung der einzelnen Schritte dieses Tools wurde, anhand eines Beispiels, im vierten Kapitel beschrieben.

## 5.2 Ausblick

WS-BPEL ist eine weit verbreitete Sprache für die Beschreibung von Geschäftsprozessen. Da WS-BPEL keine formale Semantik besitzt, existiert eine Übersetzung von WS-BPEL in oWFNs. Für ein bedienbares oWFN kann ein Partner in Form eines SVAs generiert werden. Aus einem solchen Partner kann zunächst ein oWFN synthetisiert werden und das oWFN in WS-BPEL übersetzt werden. Bis auf die Synthese von oWFNs aus SVAs wurde jeder dieser Schritte, in vorherigen Arbeiten, bereits formal definiert. Die formale Definition der Synthese von oWFNs aus SVAs ist das zentrale Ergebnis dieser Arbeit.

Bisher war es möglich, einen WS-BPEL Prozess in ein oWFN zu übersetzen und für dieses oWFN zu entscheiden, ob ein Partner für eine sinnvolle Kommunikation existiert. Existiert ein solcher Partner, kann das Verhalten dieses Partners in Form eines SVAs beschrieben werden. Da es bereits möglich war, ein oWFN in WS-BPEL zu übersetzen, ist es mit den Ergebnissen dieser Arbeit möglich, einen SVA, der das Verhalten eines WS-BPEL Prozesses beschreibt, wiederum in WS-BPEL zu übersetzen. Folglich kann für jeden WS-BPEL Prozess, der einen sinnvoll kommunizierenden Partner besitzt, ein Partner in WS-BPEL berechnet werden.

Alle Schritte für die Erstellung eines Partners für einen WS-BPEL Prozess wurden, bis auf die Synthese von oWFNs aus SVAs, bereits in Tools umgesetzt. Da im Rahmen dieser Arbeit die Synthese von oWFNs aus SVAs ebenfalls in einem Tool realisiert wurde, existiert für die Berechnung und Erstellung eines Partners für einen WS-BPEL Prozess eine vollständige Toolchain.

# Literaturverzeichnis

- [AAA<sup>+</sup>07] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guizar, N. Kartha, C. K. Liu, R. Khalaf, D. König, M. Marin, V. Mehta, S. Thatte, D.v.d. Rijn, P. Yendluri, A. Yiu: *Web Services Business Process Execution Language Version 2.0*. Committee Specification, Organization for the Advancement of Structured Information Standards (OASIS), 2007.
- [Aal97] W.M.P. van der Aalst: *Verification of Workflow Nets*. In: P. Azema und G. Balbo: *Application and Theory of Petri Nets 1997*, Lecture Notes in Computer Science 1248:407-426, Berlin: Springer-Verlag, 1997.
- [CKLY98] J. Cortadella, M. Kishinevsky, L. Lavagno, A. Yakolev: *Deriving Petri Nets from Finite Transition Systems*. In: *IEEE Transactions on Computers*, vol. 47:859-882, 1998.
- [DR96] J. Desel, W. Reisig: *The synthesis problem of Petri nets*. In: *Acta Informatica* 33:297-315, 1996.
- [ER90] A. Ehrenfeucht, G. Rozenberg: *Partial 2-structures (Part I & II)*. In: *Acta Informatica* 27:315-368, 1990.
- [Got00] K. Gottschalk: *Web Services architecture overview*. IBM developerWorks, Whitepaper, 2000.
- [Kle07] Jens Kleine: *Transformation von offenen Workflow-Netzen zu abstrakten WS-BPEL-Prozessen*. Diplomarbeit, Humboldt-Universität zu Berlin, 2007.
- [LMSW06] N. Lohmann, P. Massuthe, C. Stahl, D. Weinberg: *Analyzing Interacting BPEL Processes*. In: S. Dustdar, J.L. Fiadeiro, A. Sheth: *Fourth International Conference on Business Process Management (BPM 2006)*, 5-7 September 2006 Vienna, Austria, Lecture Notes in Computer Science 4102:17-32, Springer Verlag, 2006.
- [Mil89] R. Milner: *Communication and Concurrency*. Prentice Hall, 1989.
- [MRS05] P. Massuthe, W. Reisig, K. Schmidt: *An Operating Guideline Approach*

- to the SOA*. In: *Annals of Mathematics, Computing & Teleinformatics* 1 (3):35-43, 2005.
- [MS05] P. Massuthe, K. Schmidt: *Operating Guidelines - An Automata-Theoretic Foundation for the Service-Oriented Architecture*. In: K.Y. Cai, A. Ohnishi, M. Lau: *Proceedings of the Fifth International Conference on Quality Software (QSIC 2005)*:452-457, Melbourne Australia, IEEE Computer Society, 2005.
- [Rei85] W. Reisig: *Petri Nets*. EATCS Monographs on Theoretical Computer Science, Berlin, Heidelberg, New York, Tokyo: Springer-Verlag, 1985.
- [Sch05] K. Schmidt: *Controllability of Open Workflow Nets*. In: Jörg Desel, Ulrich Frank: *Enterprise Modelling and Information Systems Architectures*, vol. P-75 *Lecture Notes in Informatics (LNI)*, Entwicklungsmethoden für Informationssysteme und deren Anwendung:236-249, Bonn: Köllen Druck+Verlag GmbH, 2005.
- [Sta05] C. Stahl: *A Petri Net Semantics for BPEL*, Techn. Report 188, Humboldt-Universität zu Berlin, 2005.
- [Wei06] D. Weinberg: *Reduction Rules for Interaction Graphs*. Technischer Bericht 198, Humboldt-Universität zu Berlin, 2006.

## **Erklärung**

Hiermit erkläre ich, die vorliegende Arbeit „Synthese offener Worflownetze aus Serviceautomaten“ selbstständig und ohne fremde Hilfe verfasst zu haben. Verwendet habe ich nur die angegebene Literatur.

Ich erkläre hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Institutes für Informatik der Humboldt-Universität zu Berlin ausgestellt werden darf.

Berlin, den 15. Januar 2008

