

# Laufzeitersetzbarkeit von Services

Nannette Liske

22. April 2007

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Grundlagen</b>	<b>3</b>
<b>3</b>	<b>Laufzeitersetzung</b>	<b>5</b>
<b>4</b>	<b>Algorithmus für die Beschriftung</b>	<b>7</b>
4.1	Beispiel . . . . .	11
4.2	Korrektheit und Vollständigkeit . . . . .	13
<b>5</b>	<b>Berechnen einer Laufzeitersetzung</b>	<b>17</b>
<b>6</b>	<b>Fazit</b>	<b>19</b>

## 1 Einleitung

Ein bekanntes Modell zur Beschreibung von Geschäftsprozessen sind *Service Automaten* [Mar03]. Diese endlichen Automaten modellieren sowohl das Verhalten eines Kunden als auch das einer Firma während ihrer Interaktion. Betrachten wir einen bestimmten Service Automaten, dann ist die *Operating Guideline* [MS05a, MRS05] eine kompakte Darstellung aller Service Automaten, die mit ihm verklemmungsfrei interagieren. Repräsentiert der Service Automat das Verhalten einer Firma, dann beschreibt die Operating Guideline alle für diese Firma geeigneten Kundenverhalten.

Zwei Service Automaten gelten als (bedienungs-) *äquivalent* [Mar03], wenn ihre Operating Guidelines übereinstimmen. In diesem Fall bleiben, wenn einer der Service Automaten zur Entwurfszeit durch den anderen ersetzt wird, die Interaktionspartner die selben. In der Veranschaulichung mit der Firma bedeutet dies, dass zwei Firmen, deren zugehörige Service Automaten äquivalent sind, mit den selben Kunden umgehen können.

Wann ein Service zur Entwurfszeit gegen einen anderen ausgetauscht werden kann, wenn man sich auf nur eine Strategie beschränkt, ist in [Ric02] untersucht worden. Verallgemeinert wurde dieser Ansatz in [Bre07], bei dem die Strategien durch eine gegebene Operating Guideline beschrieben sind.

Für manche Anwendungen ist allerdings die Frage interessanter, ob denn ein Service Automat *zur Laufzeit*, also während der Interaktion mit einem Partner, durch einen anderen Service Automaten ersetzt werden kann. Eine mögliche Motivation bietet die Situation, wenn eine Versicherungsfirma von einer anderen übernommen wird. Die internen Geschäftsprozesse werden verändert; für alle Kunden, die laufende Verträge mit dem Unternehmen haben, soll sich dabei allerdings nichts ändern. Genauso könnte es für eine Firma von Interesse sein, nur einen bestimmten Kundenstamm an eine andere Firma zu übergeben. Auch hier sollen die Kunden ihr Kommunikationsmuster beibehalten können.

Die vorliegende Arbeit führt daher das Konzept der *Laufzeitersetzungen* für Service Automaten ein und untersucht anschließend, wie eine solche Laufzeitersetzung berechnet werden kann.

Kapitel 2 beschäftigt sich zunächst mit den Grundlagen und kann von Lesern, welche mit Service Automaten und Operating Guidelines vertraut sind, übersprungen werden. In Kapitel 3 werden dann die neuen Begriffe zur Beschreibung und Bestimmung der Laufzeitersetzung definiert. Anschließend folgt in Kapitel 4 der Hauptteil dieser Arbeit, nämlich ein Algorithmus zur Beschriftung der Service Automaten mit Informationen über bestimmte Interaktionszustände. Die durch die Beschriftung markierten Interaktionszustände geben letztendlich Auskunft darüber, an welchen Stellen eine Überführung vom einen zum anderen Service erfolgen kann. Die Korrektheit und Vollständigkeit des Algorithmus' wird anschließend bewiesen. Der eigentliche Algorithmus zum Berechnen einer Laufzeitersetzung und einige Korollare werden in Kapitel 5 präsentiert.

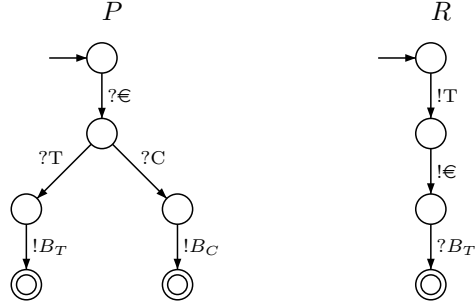


Abbildung 1: Zwei Service Automaten  $P$  und  $R$ .

## 2 Grundlagen

Als Modell für die Services, auf dem die Laufzeitersetzung definiert werden soll, verwenden wir spezielle endliche Automaten. Diese so genannten *Service Automaten* sind in der Literatur bereits etabliert [MS05a, MW06]. Im Hinblick auf die Untersuchung der Ersetzbarkeit erweisen sie sich zudem als besonders geeignet.

Zwei Services interagieren in unserem Modell durch *asynchrone Kommunikation*. Das bedeutet, es gibt einen *Nachrichtenkanal*, der alle gesendeten Nachrichten solange enthält bis diese empfangen werden. Die Reihenfolge der Nachrichten bleibt unberücksichtigt. Mit  $MC$  bezeichnen wir die (endliche) Menge aller möglichen Nachrichten. Die folgenden Definitionen sind angelehnt an [MW06].

### Definition 1 (Service Automat)

Ein Service Automat ist ein nichtdeterministischer Automat  $A = [I, Q, T, q_0, \Omega]$ , bestehend aus einem Interface  $I = [I_{in}, I_{out}]$ , einer Menge von Zuständen  $Q$ , einer Menge  $T \subseteq Q \times L \times Q$  von Transitionen, einem Startzustand  $q_0 \in Q$  und einer Menge  $\Omega \subseteq Q$  von Endzuständen. Dabei gilt  $I_{in} \cup I_{out} \subseteq MC$ ,  $I_{in} \cap I_{out} = \emptyset$  und  $L = \{?a \mid a \in I_{in}\} \cup \{!a \mid a \in I_{out}\} \cup \{\tau\}$ .

Abbildung 1 zeigt die grafische Darstellung zweier Service Automaten.

In der Literatur wurden bislang hauptsächlich zyklenfreie Services untersucht, weil die Analyse von Automaten mit Zyklen ungleich komplizierter ist. Auch in dieser Arbeit sollen nur azyklische Services untersucht werden. Da jeder azyklische Automat als gerichteter Baum dargestellt werden kann und Bäume besonders einfach zu untersuchende Strukturen darstellen, beschränken wir uns im Folgendem auf Baumautomaten als Services.

### Definition 2 (Interaktion von Service Automaten)

Seien  $P$  und  $R$  Service Automaten. O.B.d.A. sei  $Q_P \cap Q_R = \emptyset$ . Das die Interaktion beschreibende Transitionssystem  $(P \oplus R) = (Q, T, q_0)$  besteht aus einer Menge von

Zuständen  $Q \subseteq Q_P \times Q_R \times \text{bags}(MC)$  und einer Menge von gelabelten Transitionen  $T \subseteq Q \times (T_P \cup T_R) \times Q$ . Die Zustände und Transitionen sind wie folgt induktiv definiert:

- $q_0 = (q_{0_R}, q_{0_R}, \{\})$  ist ein Zustand von  $Q$ .
- Wenn  $q = (q_P, q_R, M)$  ein Zustand ist und eine Transition  $t$  existiert mit
  - $t = (q_R, !a, q'_R) \in T_R$ , dann ist  $q' = (q_P, q'_R, M + a) \in Q$  und  $(q, t, q') \in T$ ,
  - $t = (q_P, !a, q'_P) \in T_P$ , dann ist  $q' = (q'_P, q_R, M + a) \in Q$  und  $(q, t, q') \in T$ ,
  - $t = (q_R, ?a, q'_R) \in T_R$  und  $a \in M$ , dann ist  $q' = (q_P, q'_R, M - a) \in Q$  und  $(q, t, q') \in T$ ,
  - $t = (q_P, ?a, q'_P) \in T_P$  und  $a \in M$ , dann ist  $q' = (q'_P, q_R, M - a) \in Q$  und  $(q, t, q') \in T$ ,
  - $t = (q_R, \tau, q'_R) \in T_R$ , dann ist  $q' = (q_P, q'_R, M) \in Q$  und  $(q, t, q') \in T$  und
  - $t = (q_P, \tau, q'_P) \in T_P$ , dann ist  $q' = (q'_P, q_R, M) \in Q$  und  $(q, t, q') \in T$ .

Ein Zustand  $q = (q_P, q_R, M)$  ohne Nachfolger ist ein *Deadlock*. Ein Deadlock mit  $q_P \in \Omega_P$ ,  $q_R \in \Omega_R$  und  $M = \emptyset$ , heißt *Endzustand*.

**Definition 3 (schwach terminierend)**

Seien  $P$  und  $R$  zwei Service Automaten.  $P \oplus R$  heißt schwach terminierend, wenn alle Deadlocks von  $P \oplus R$  auch Endzustände sind.

Das Erreichen eines Deadlocks, der kein Endzustand ist, ist ein unerwünschtes Verhalten. Für einen gegebenen Service  $P$  bezeichnen wir daher einen Automaten  $R$  als *Strategie* für  $P$ , wenn  $P \oplus R$  schwach terminiert. Zum Beispiel ist der Service Automat  $R$  in Abbildung 2 eine Strategie für den Automaten  $P$ .

Um alle Strategien für  $P$  zu repräsentieren, nutzen wir das Konzept der Operating Guidelines [MS05a, MRS05].

**Definition 4 (Subautomat)**

Ein Automat  $A' = [I_{A'}, Q_{A'}, T_{A'}, q_{0_{A'}}, \Omega_{A'}]$  ist Subautomat von  $A = [I_A, Q_A, T_A, q_{0_A}, \Omega_A]$  genau dann, wenn  $Q_{A'} \subseteq Q_A$ ,  $q_{0_{A'}} = q_{0_A}$ ,  $T_{A'} \subseteq T_A$  und  $\Omega_{A'} \supseteq \Omega_A \cap Q_{A'}$ .

**Definition 5 (Annotation)**

Sei  $\phi$  eine Funktion, die jedem Zustand  $q$  eines Automaten  $A$  eine boolesche Formel  $\phi(q)$  zuweist, wobei als Variablen die Elemente der Menge  $\{!a, ?a \mid a \in MC\}$  verwendet werden. Dann heißt  $\phi$  Annotation von  $A$ .

Ferner nennen wir  $A^\phi$  einen *annotierten Automaten*.

**Definition 6 (erfüllender Subautomat)**

Sei  $A^\phi$  ein annotierter Automat,  $A'$  ein Subautomat von  $A$ .

$A'$  heißt genau dann erfüllender Subautomat von  $A^\phi$ , wenn für jeden Zustand  $q \in Q_{A'}$  gilt:

Belegt man jede Variable  $x$  der Formel  $\phi(q)$  genau dann mit wahr, wenn es eine entsprechende Kante  $(q, x, q') \in T_{A'}$  gibt, so erfüllt diese Belegung  $\phi(q)$ .

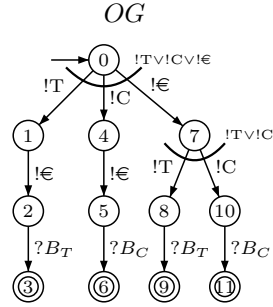


Abbildung 2: Eine Operating Guideline  $OG$  für den Automaten  $P$  aus Abbildung 1. Die Annotationen der nicht beschrifteten Zustände entsprechen dem Label der ausgehenden Transition.

**Definition 7 (Operating Guideline)**

Sei  $P$  ein Service und  $OG = A^\phi$  ein deterministischer annotierter Automat.  $OG$  heißt genau dann Operating Guideline zu  $P$ , wenn

- jede Strategie  $S$  von  $P$  ein erfüllender Subautomat von  $OG$  und
- jeder erfüllende Subautomat von  $OG$  eine Strategie von  $P$  ist.

Zur einfacheren Lesbarkeit verwenden wir im folgenden  $Q_{OG}$  statt  $Q_A$  für die Zustände und  $T_{OG}$  statt  $T_A$  für die Transitionen der Operating Guideline.

In Abbildung 2 ist zur Anschauung eine Operating Guideline für den Automaten  $P$  aus Abbildung 1 dargestellt. Man sieht leicht, dass eine Operating Guideline die Menge derjenigen Services repräsentiert, die als Interaktionspartner für  $P$  in Frage kommen. Die Berechnung einer solchen Operating Guideline  $OG$  zu einem Service Automaten  $P$  ist in [MS05c] beschrieben. Die Autoren geben weiterhin ein Verfahren an, bei dem anhand der Operating Guideline entschieden wird, ob ein Service Automat  $R$  eine Strategie für  $P$  darstellt oder nicht.

**Definition 8 (Comply<sup>w</sup>(OG))**

Sei  $OG$  eine Operating Guideline und  $w \in Q_{OG}$ . Die Menge  $\text{Comply}^w(OG)$  bezeichne alle Service Automaten, die erfüllende Subautomaten von  $OG$  sind und den Zustand  $w$  beinhalten.

### 3 Laufzeitersetzung

Zunächst wollen wir formal festlegen und verdeutlichen, was eine korrekte Laufzeitersetzung ist. Anschließend werden wir ein Hilfsmittel angeben, das sich relativ einfach berechnen lässt. Nach einem Beweis der Korrektheit und Vollständigkeit des angegebenen Algorithmus' sind wir auch in der Lage, das eigentliche Problem – nämlich das Berechnen einer korrekten Laufzeitersetzung – algorithmisch einfach zu lösen.

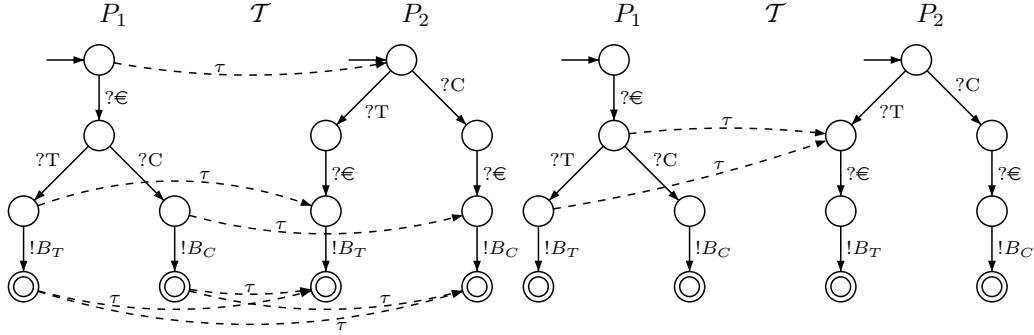


Abbildung 3: Eine gültige (links) und eine ungültige Laufzeitersetzung (rechts) bezüglich der  $OG$  aus Abbildung 2.

### Definition 9 (gültige Laufzeitersetzung)

Seien  $P_1$  und  $P_2$  zwei Service Automaten und  $OG$  eine beliebige Operating Guideline. Eine nichtleere Menge  $\mathcal{T} \subseteq (Q_{P_1} \times \tau \times Q_{P_2})$  von  $\tau$ -Transitionen zwischen den Automaten heißt Laufzeitersetzung. Wir nennen eine Laufzeitersetzung gültig bezüglich  $OG$ , falls für jeden erfüllenden Subautomaten  $R$  von  $OG$  gilt: Wenn  $P_1 \oplus R$  schwach terminiert, so terminiert auch  $(P_1 \cup P_2 \cup \mathcal{T}) \oplus R$  schwach.

Bildlich dargestellt heißt das, dass ein Service Automaten  $R$  anhand der Interaktion nicht „merkt“, ob sein Interaktionspartner  $P_1$  oder  $P_1 \cup P_2 \cup \mathcal{T}$  ist. Man beachte, dass die  $\tau$ -Transitionen nur zum Service  $P_2$  gerichtet sind. Dadurch wird bei der Interaktion, sobald zum ersten Mal ein Zustand von  $P_2$  erreicht wurde, nie wieder ein Zustand von  $P_1$  verwendet. Zur Veranschaulichung sind in Abbildung 3 zwei verschiedene Laufzeitersetzungen dargestellt, von denen nur die linke gültig ist (bezüglich der  $OG$  aus Abbildung 2). Die rechte Laufzeitersetzung ist offensichtlich nicht gültig, da der Service  $P_2$  auf  $?e$  wartet während  $P_1$  bereits  $?e$  erhalten hat.

### Proposition 1

Seien  $\mathcal{T}_1$  und  $\mathcal{T}_2$  gültige Laufzeitersetzungen, dann ist auch  $\mathcal{T}_1 \cup \mathcal{T}_2$  eine gültige Laufzeitersetzung.

### Definition 10 (Maximalität)

Eine gültige Laufzeitersetzung  $\mathcal{T}$  heißt maximal, wenn es keine  $\tau$ -Transitionen gibt, so dass  $\tau \notin \mathcal{T}$  und  $\mathcal{T} \cup \{\tau\}$  eine gültige Laufzeitersetzung ist.

Die Idee des in dieser Arbeit vorgestellten Algorithmus' zum Finden einer maximalen gültigen Laufzeitersetzung ist: Für jeden Zustand der gegebenen Service Automaten  $P_1$  und  $P_2$  speichern wir alle „passenden“ Zustände eines potentiellen Interaktionspartners  $R$  (ein Service gemäß der gegebenen  $OG$ ) zusammen mit dem „erforderlichen“ Nachrichtenkanal. Anschließend werden anhand dieser Informationen diejenigen Zustands-paare aus  $Q_{P_1} \times Q_{P_2}$  gefunden, die für die Laufzeitersetzung durch eine  $\tau$ -Transition verbunden werden können.

**Definition 11 (Beschriftung)**

Sei  $P$  ein Service Automat und  $OG$  eine Operating Guideline. Als Beschriftung von  $P$  bezeichnen wir eine Abbildung  $B : Q_P \rightarrow \mathfrak{P}(Q_{OG} \times \text{bags}(MC))$ .

Für jeden Zustand des Service Automaten  $P$  notieren wir also Paare aus  $OG$ -Zustand und Nachrichtenkanal. Für die Beschriftung wollen wir solche Paare finden, die eine schwach terminierende Interaktion zwischen dem Service Automaten  $P$  und einem Service  $R$  gemäß  $OG$  beschreiben.

**Definition 12 (Deadlockfreiheit von  $(v, w, m)$ )**

Sei  $P$  ein Service Automat,  $v \in Q_P$ ,  $OG$  eine Operating Guideline,  $w \in Q_{OG}$  und  $m$  ein Nachrichtenkanal. Dann heißt  $(v, w, m)$  deadlockfrei, falls für alle  $R \in \text{Comply}^w(OG)$  das Transitionssystem aus  $P$  mit Startzustand  $v$  und  $R$  mit Startzustand  $w$  bei initialem Nachrichtenkanal  $m$  schwach terminiert.

## 4 Algorithmus für die Beschriftung

Wir haben nun alle nötigen Begriffe zusammen, die uns helfen, einen Algorithmus zum Berechnen gültiger Laufzeitersetzungen anzugeben und dessen Korrektheit zu beweisen.

Sei von nun an  $OG$  eine beliebige, aber fest vorgegebene Operating Guideline. Wie im vorherigen Abschnitt angedeutet, wollen wir nun zu beliebigen Service Automaten die Information über „passende“ Zustände speichern. Sei daher ein Service Automat  $P$  ebenfalls beliebig, aber im folgenden fest vorgegeben.

Die Speicherung der Information erfolgt, wie oben definiert, durch eine *Beschriftung*  $B$  der Zustände von  $P$  derart, dass jede dieser Beschriftungen einen zulässigen – d.h. deadlockfreien – Zustand der Interaktion von  $P$  mit einem Service gemäß  $OG$  beschreibt. Für jeden Knoten  $v \in Q_P$  und jedes Paar  $(w, m)$  aus der Beschriftungsmenge  $B(v)$  soll also die Deadlockfreiheit von  $(v, w, m)$  gelten. Das bedeutet: Befände sich der Service  $P$  im Zustand  $v$  während der Interaktion mit einem anderen Service gemäß  $OG$ , welcher sich im Zustand  $w$  befindet, und ist  $m$  dabei der aktuelle Nachrichtenkanal, so können die Services nicht in einen Deadlock geraten, der kein Endzustand ist.

Zur Veranschaulichung zeigt die Abbildung 4 einen bereits komplett beschrifteten Automaten  $P$ . Dabei ist beispielsweise der Zustand  $c$  mit  $(2, \emptyset)$  beschriftet. Das bedeutet laut unserer Vorgabe, dass  $(c, 2, \emptyset)$  deadlockfrei sein soll. Tatsächlich terminiert ein entsprechendes Transitionssystem ab diesem Zustand schwach, da in der Interaktion nun  $P$  nur noch die Möglichkeit hat, die Nachricht  $B_T$  zu senden. Anschließend wird ein Service  $R$  aus  $\text{Comply}^2(OG)$  diese Nachricht empfangen, womit die Interaktion in einen erlaubten Endzustand gelangt.

Diese korrekte Beschriftung ist das eigentliche Kernstück dieser Arbeit. Für die algorithmische Umsetzung verwenden wir den folgenden, relativ simplen Ansatz:

Wir durchlaufen die Guideline  $OG$  und den Automaten  $P$  per Tiefensuche (DFS). Für jeden Zustand  $w \in Q_{OG}$  und jeden Zustand  $v \in Q_P$  entscheiden wir dabei

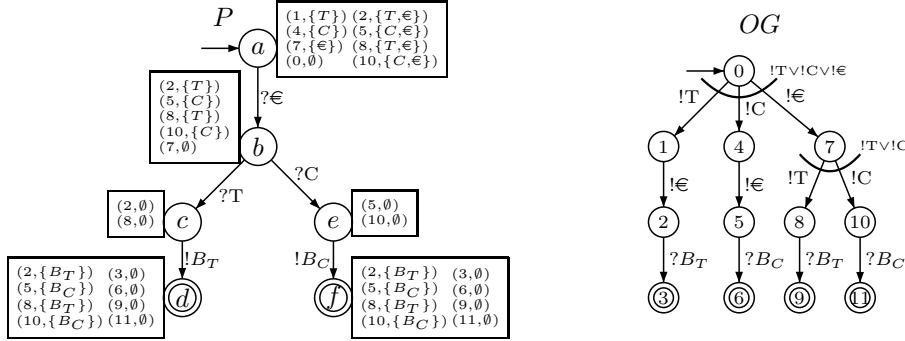


Abbildung 4: Ein bezüglich  $OG$  beschrifteter Service Automat  $P$ .

zunächst, welche Nachrichtenkanäle  $m \in bags(MC)$  überhaupt als Kandidaten für eine Beschriftung  $(w, m)$  des Zustands  $v$  in Frage kommen. Anschließend überprüfen wir mit der rekursiv gewonnenen Information, welche Zustände  $(v, w, m)$  tatsächlich deadlockfrei sind und somit zur Beschriftung hinzugefügt werden.

Es folgt die Implementation im Pseudo-Code:

BESCHRIFTE(Service  $P$ , Operating Guideline  $OG$ )

- (1) **foreach**  $v \in Q_P$
- (2)  $B(v) := \emptyset$
- (3)  $DFSOG(P, q_{0OG})$
- (4) **return**  $B$

$DFSOG$ (Service  $P$ ,  $OG$ -Zustand  $w$ )

- (1) **foreach**  $w' : [w, e, w'] \in T_{OG}$
- (2)  $DFSOG(w', P)$
- (3)  $DFSOG(w, q_{0P})$

$DFSOG$ ( $P$ -Zustand  $v$ ,  $OG$ -Zustand  $w$ )

- (1) **foreach**  $v' : [v, e, v'] \in T_P$
- (2)  $DFSOG(v, w')$
- (3) **if**  $w \in \Omega_{OG}$  **and**  $v \in \Omega_P$
- (4)  $B(v) := B(v) \cup \{(w, \emptyset)\}$
- (5) **else**
- (6)  $BESCHRIFTEANHANDP(v, w)$
- (7)  $BESCHRIFTEANHANDOG(v, w)$

Den Rekursionsanfang bilden die Endzustände der Automaten. Da die Interaktion zweier Automaten garantiert schwach terminiert, sobald sie sich in einem Endzustand befindet, werden die Endzustände von  $P$  mit den Endzuständen von  $OG$  und leerem Nachrichtenkanal beschriftet (Z. 4,  $DFSOG$ ). Weiterhin gilt, dass jeder Nicht-Endzustand  $(v, w, m)$  einer Interaktion dadurch fortgesetzt wird, dass  $P$  einen Schritt zu einem Nachfolgezustand von  $v$  geht oder aber  $R$  zu einem Nachfolger von  $w$  wechselt. Daher teilt sich die Suche nach korrekten Beschriftungen in der Funktion  $DFSOG$  in zwei Teile: das Prüfen der Nachfolgezustände anhand der  $P$ -Transitionen und anhand der

OG-Transitionen (Z. 6 und 7, DFSP).

Der Algorithmus verwendet dabei die rekursiv gewonnenen Informationen über die Nachfolgestände, um die Beschriftungen für den aktuellen Zustand  $w$  zu ermitteln.

BESCHRIFTEANHANDP( $v, w$ )

- (1)  $M := \text{BESTIMMEKANAELEP}(v, w)$
- (2) **foreach**  $m \in M$
- (3)     **if**  $\text{KEINDEADLOCKP}(v, w, m)$
- (4)          $B(v) := B(v) \cup \{(w, m)\}$

BESCHRIFTEANHANDOG( $v, w$ )

- (1)  $M := \text{BESTIMMEKANAELEOG}(v, w)$
- (2) **foreach**  $m \in M$
- (3)     **if**  $\text{KEINDEADLOCKOG}(v, w, m)$  **and**
- (4)          $\text{KEINDEADLOCKANNOTATION}(w, m)$
- (5)          $B(v) := B(v) \cup \{(w, m)\}$

In den Funktionen BESCHRIFTEANHANDP und BESCHRIFTEANHANDOG wird jeweils zuerst eine Menge von Nachrichtenkanälen zur potentiellen Beschriftung des aktuellen Zustandes  $v$  bestimmt. Im zweiten Schritt wird anschließend für jeden dieser ermittelten Kandidaten der dazugehörige Interaktionszustand auf Deadlockfreiheit überprüft.

Den ersten Schritt – das Bestimmen der möglichen Nachrichtenkanäle – führen folgende Funktionen durch:

BESTIMMEKANAELEP( $v, w$ )

- (1)  $M := \emptyset$
- (2) **foreach**  $[v, e, v'] \in T_P$
- (3)     **foreach**  $m : (w, m) \in B(v')$
- (4)         **if**  $e = !a$  **and**  $a \in m$
- (5)              $M := M \cup \{m - \{a\}\}$
- (6)         **if**  $e = ?a$
- (7)              $M := M \cup \{m + \{a\}\}$
- (8)         **if**  $e = \tau$
- (9)              $M := M \cup \{m\}$
- (10) **return**  $M$

BESTIMMEKANAELEOG( $v, w$ )

- (1)  $M := \emptyset$
- (2) **foreach**  $[w, e, w'] \in T_{OG}$
- (3)     **foreach**  $m : (w', m) \in B(v)$
- (4)         **if**  $e = !a$  **and**  $a \in m$
- (5)              $M := M \cup \{m - \{a\}\}$
- (6)         **if**  $e = ?a$
- (7)              $M := M \cup \{m + \{a\}\}$
- (8) **return**  $M$

Die Menge  $M$  der möglichen neuen Beschriftungen für  $v$  ergibt sich aus den bereits vorhandenen Beschriftungen von  $v$  mit  $w'$  für einen Nachfolger  $w'$  von  $w$  bzw. den Beschriftungen von  $v'$  mit  $w$  für einen Nachfolger  $v'$  von  $v$ . Der jeweilige Nachrichtenkanal aus der vorhandenen Beschriftung wird dabei entsprechend dem Transitionslabel, das zu diesem Nachfolger führt, modifiziert und als weiterer Kandidat zu  $M$  hinzugefügt. Dass solch ein neuer Nachrichtenkanal  $m$  bei der Interaktion der Automaten in den Zuständen  $w$  bzw.  $v$  durch das Benutzen einer Transition gemäß dem Transitionslabel derart überführt wird, dass der neue Zustand wieder deadlockfrei ist, ist zwar ein notwendiges aber kein hinreichendes Kriterium dafür, dass  $(v, w, m)$  selbst deadlockfrei ist. Daher gilt  $m$  zunächst nur als Kandidat für die Beschriftung. Es fehlt noch der Test, ob die entsprechende Transition im Zustand  $(v, w, m)$  überhaupt verwendet werden darf. Dies übernehmen die folgenden Funktionen:

KEINDEADLOCKP( $v, w, m$ )

```
(1)  foreach [ $v, ?a, v'$ ]  $\in T_P$  and  $a \in m$ 
(2)    if  $(w, m - \{a\}) \notin B(v')$ 
(3)      return false
(4)  foreach [ $v, !a, v'$ ]  $\in T_P$ 
(5)    if  $(w, m + \{a\}) \notin B(v')$ 
(6)      return false
(7)  foreach [ $v, \tau, v'$ ]  $\in T_P$ 
(8)    if  $(w, m) \notin B(v')$ 
(9)      return false
(10) return true
```

KEINDEADLOCKOG( $v, w, m$ )

```
(1)  foreach [ $w, ?a, w'$ ]  $\in T_{OG}$  mit  $a \in m$ 
(2)    if  $(w', m - \{a\}) \notin B(v)$ 
(3)      return false
(4)  foreach [ $w, !a, w'$ ]  $\in T_{OG}$ 
(5)    if  $(w', m + \{a\}) \notin B(v)$ 
(6)      return false
(7)  return true
```

In den Funktionen KEINDEADLOCKP und KEINDEADLOCKOG prüfen wir, ob jeder erreichbare Nachfolgezustand eine solche Beschriftung trägt, dass eine Fortsetzung der Interaktion gewährleistet ist.

Eine wichtige Besonderheit bei KEINDEADLOCKOG ist, dass ein Automat  $R \in \text{Comply}^w(OG)$  lediglich eine Teilmenge der Nachfolgezustände von  $w$  besitzt. Wir müssen also noch sicherstellen, dass für jeden  $R \in \text{Comply}^w(OG)$  mindestens ein erreichbarer Nachfolger existiert.

KEINDEADLOCKANNOTATION( $w, m$ )

```
(1)   $L := \emptyset$ 
(2)  foreach [ $w, e, w'$ ]  $\in T_{OG}$ 
(3)    if  $e = !a$  or  $(e = ?a$  and  $a \in m)$ 
(4)       $L := L \cup \{e\}$ 
(5)  Sei  $\phi(w) = \alpha_1 \wedge \dots \wedge \alpha_k$ 
(6)  foreach  $\alpha_i = \beta_{i,1} \vee \dots \vee \beta_{i,l}$ 
(7)    if  $\{\beta_{i,j} \mid j = 1 \dots l\} \subseteq L$ 
(8)      return true
(9)  return false
```

Der Algorithmus für die Berechnung einer Operating Guideline in [MS05c] erzeugt die Annotation der Zustände in Form einer konjunktiven Normalform. Daher gehen wir in KEINDEADLOCKANNOTATION ebenfalls von dieser Form aus. Wir wollen wissen, ob es für jede Menge von Nachfolgern, die die Annotation erfüllen, einen erreichbaren Nachfolger mit unserem Nachrichtenkanal gibt. Im Beweis in Abschnitt 4.2 zeigen wir, dass Deadlockfreiheit genau dann gewährleistet ist, wenn aus mindestens einer Klausel alle Nachfolger erreichbar sind.

## 4.1 Beispiel

Schauen wir uns zur Verdeutlichung der Funktionsweise den Algorithmus zur Beschriftung des Automaten aus Abbildung 4 genauer an. Wir betrachten dazu einige bestimmte Zeitpunkte während der Abarbeitung, die veranschaulichen, wie die Beschriftung des Zustandes  $b$  zustande kommt.

Die äußere Rekursion  $\text{DFSOG}$  traversiert die Operating Guideline. Beim ersten Rekursionsende ist dabei  $w = 3$  der aktuelle Knoten. Dann wird die Funktion  $\text{DFS}(a, 3)$  aufgerufen, die ihrerseits den Automaten  $P$  rekursiv komplett traversiert. Da  $w = 3$  ein Endzustand ist, sieht die Beschriftung anschließend wie in Abbildung 5 aus.

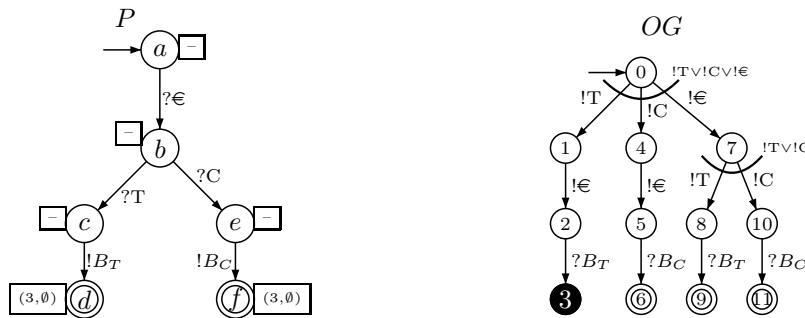


Abbildung 5: Der Zwischenstand der Beschriftung *nach* dem Aufruf  $\text{DFS}(a, 3)$ .

In der Operating Guideline geht  $\text{DFSOG}$  nun rekursiv eine Stufe zurück zum Knoten  $w = 2$  und startet erneut die innere Rekursion  $\text{DFS}$  auf dem Service Automaten  $P$ . Betrachten wir nun einmal genauer die innere Rekursion in dem Aufruf  $\text{DFS}(b, 2)$ , und halten vor Zeile 3 dieser Funktion an. Dann ergibt sich die in Abbildung 6 dargestellte Beschriftung.

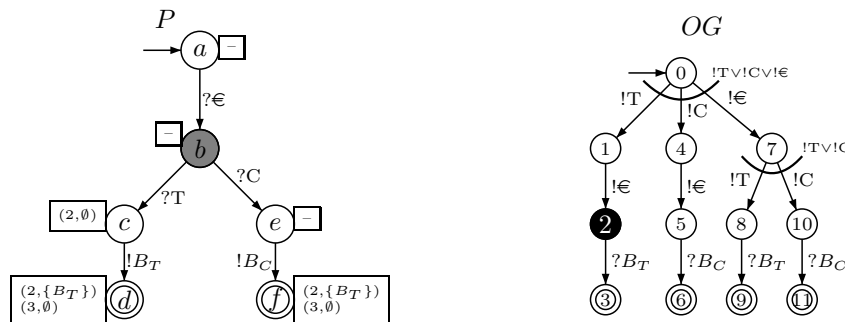


Abbildung 6: Der Zwischenstand der Beschriftung *während* des Aufrufes  $\text{DFS}(b, 2)$ .

Die Zeilen 3 und 4 dieser Funktion werden übersprungen, sodass nun in Zeilen 6 und

7 die Beschriftungen des Zustandes  $w = 2$  mit dem  $P$ -Zustand  $v = b$  vorgenommen werden.

Der erste Funktionsaufruf ist also  $\text{BESCHRIFTEANHANDP}(b, 2)$ . Dabei werden erst die möglichen Nachrichtenkanäle bestimmt zu  $M = \{\{T\}\}$ . Dieser eine Nachrichtenkanal  $\{T\}$  kommt zustande, weil der Nachfolgezustand  $c$  bereits mit  $(2, \emptyset)$  beschriftet ist und die dazugehörige Transition das Label  $?T$  trägt. Andere Nachfolger sind hingegen nicht mit 2 beschriftet. Das ist auch der Grund, warum innerhalb der Beschriftungsfunktion der Aufruf  $\text{KEINDEADLOCKP}(b, 2, \{T\})$  als Ergebnis **true** liefert. Die erste Beschriftung am Zustand  $b$  ist somit  $(2, \{T\})$ .

Der zweite Funktionsaufruf,  $\text{BESCHRIFTEANHANDOG}(b, 2)$ , verändert nun nichts mehr an der Beschriftung, da keine möglichen Kandidaten existieren ( $b$  ist nämlich nicht mit 3, dem einzigen Nachfolger von 2, beschriftet).

Ähnlich sieht es aus, wenn die äußere Rekursion den Zustand  $w = 5$  erreicht. Wir betrachten nun wieder den Fall, dass der Nachfolger, 6, von  $w$  bereits abgearbeitet wurde und wir uns wieder in der inneren Rekursion beim Zustand  $v = b$  befinden. Während des Aufrufes  $\text{DFSP}(b, 5)$ , genauer in Zeile 3, liegt eine Beschriftung gemäß Abbildung 7 vor.

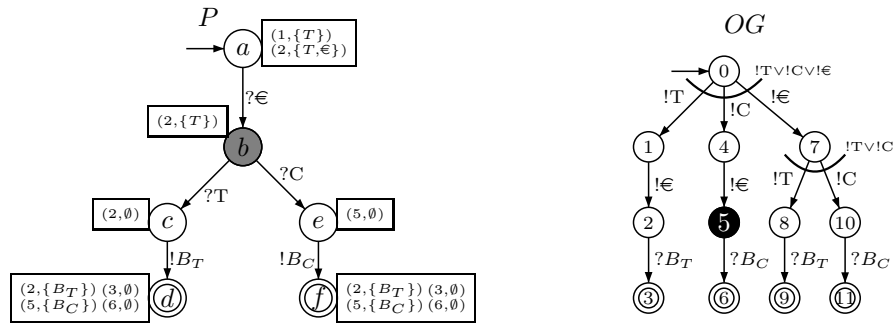


Abbildung 7: Der Zwischenstand der Beschriftung während des Aufrufes  $\text{DFSP}(b, 5)$ .

Hier verändert analog zur vorherigen Betrachtung nur die erste Beschriftungsfunktion,  $\text{BESCHRIFTEANHANDP}(b, 5)$ , die Beschriftung des Zustandes  $b$ . Beim zweiten Funktionsaufruf,  $\text{BESCHRIFTEANHANDOG}(b, 5)$ , bleibt die Menge der möglichen Nachrichtenkanäle wieder leer.

Die rekursiven Aufrufe  $\text{DFSP}(b, 8)$  und  $\text{DFSP}(b, 10)$  verlaufen ganz analog.

Anders verhält es sich zum Beispiel, wenn anschließend in der äußeren Rekursion für den Zustand  $w = 7$  die Rekursion auf  $P$  gestartet wird. Unser Augenmerk liegt erneut auf der Bearbeitung des  $P$ -Zustandes  $v = b$ . Vor dem Aufruf  $\text{BESCHRIFTEANHANDP}(b, 7)$  sieht die Beschriftung wie in Abbildung 8 dargestellt aus.

Die rekursiven Aufrufe für die Nachfolger von  $b$  verändern allesamt nichts an dieser

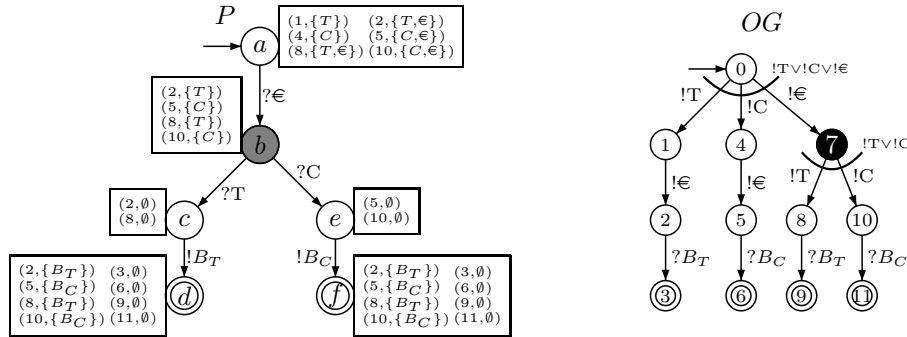


Abbildung 8: Der Zwischenstand der Beschriftung vor dem Aufruf  $DFSP(b, 7)$ .

Beschriftung. Danach, in Zeile 6, wird der Aufruf  $BESCHRIFTEANHANDP(b, 7)$  nichts tun, weil kein Nachfolger von  $b$  mit 7 beschriftet ist.

Der Zustand  $b$  ist allerdings bereits sowohl mit  $(8, \{T\})$  als auch mit  $(10, \{C\})$  beschriftet. Beide Zustände sind auch Nachfolger von  $w = 7$  in  $OG$  und der vermerkte Nachrichtenkanal enthält jeweils die Nachricht, die als Transitionslabel mit einem  $!$  versehen ist. Daher findet der Aufruf  $BESCHRIFTEANHANDOG(b, 7)$  den Kandidaten  $\emptyset$  gleich zweimal als möglichen Nachrichtenkanal. Die Funktion  $KEINDEADLOCKOG(b, 7, \emptyset)$  stellt nun sicher, dass der Zustand  $v = b$  tatsächlich mit allen Nachfolgern von  $w = 7$  und dem entsprechend veränderten Nachrichtenkanal beschriftet ist.

Zuletzt überprüft noch die Funktion  $KEINDEADLOCKANNOTATION(7, \emptyset)$ , ob auch jeder Subautomat aus  $Comply^7(OG)$  eine Transition ausführen kann, wenn der Nachrichtenkanal  $m = \emptyset$  ist. Die Menge der Label der möglichen Transitionen wird dabei zu  $L = \{!T, !C\}$  bestimmt. Die Annotation lautet  $\Phi(7) = !T \vee !C$ , dass heißt  $KEINDEADLOCKANNOTATION$  prüft in Zeile 7, ob  $\{!T, !C\} \subseteq L$ . In unserem Beispiel ist dies der Fall, also wird die Beschriftung  $(7, \emptyset)$  zu  $B(v)$  hinzugefügt.

Die komplette Beschriftung ist in Abbildung 4 dargestellt.

## 4.2 Korrektheit und Vollständigkeit

Die Korrektheit und Vollständigkeit des präsentierten Algorithmus' lässt sich folgendermaßen formalisieren:

**Satz 1** Sei  $P$  ein Service und  $OG$  eine Operating Guideline. Für jeden Zustand  $v \in Q_P$ ,  $w \in Q_{OG}$  und jeden Nachrichtenkanal  $m$  gilt:  $(v, w, m)$  ist genau dann deadlockfrei, wenn der Algorithmus  $BESCHRIFTE(P, OG)$  den Zustand  $v$  mit  $(w, m)$  beschriftet.

Für den Beweis benötigen wir die zwei nachstehenden Lemmata, die jeweils die korrekte Funktionsweise der Teilprozeduren  $BESCHRIFTEANHANDP$  bzw.  $BESCHRIFTEANHANDOG$  sicherstellen.

Für die einfachere Lesbarkeit verwenden wir im folgenden den Begriff „*erreichbarer Nachfolger*“ für Zustände aus  $P$  bzw.  $OG$ , wenn durch den jeweiligen Kontext ein Interaktionszustand  $(v, w, m)$  beschrieben ist. Die *erreichbaren Nachfolger* von  $v \in Q_P$  sind dann Zustände  $v' \in Q_P$ , sodass in der Interaktion ein Nachfolgezustand  $(v', w, m')$  existiert. Für  $w \in Q_{OG}$  seien die erreichbaren Nachfolger jene Zustände  $w' \in Q_{OG}$ , für die es einen Nachfolgezustand  $(v, w', m')$  der Interaktion gibt.

**Lemma 1** *Die Funktion BESCHRIFTEANHANDP beschriftet einen Zustand  $v \in Q_P \setminus \Omega_P$  genau dann mit  $(w, m)$ , wenn  $v$  erreichbare Nachfolger hat und alle erreichbaren Nachfolger von  $v$  mit  $w$  und entsprechendem Nachrichtenkanal beschriftet sind. Formal lautet die Bedingung: Die Menge*

$$M_P := \{(v', m - \{a\}) \mid [v, !a, v'] \in T_P \text{ und } a \in m\} \cup \\ \{(v', m + \{a\}) \mid [v, ?a, v'] \in T_P\} \cup \\ \{(v', m) \mid [v, \tau, v'] \in T_P\}$$

ist nichtleer und für jedes  $(v', m') \in M_P$  ist  $v'$  mit  $(w, m')$  beschriftet.

**Beweis:**

„ $\Rightarrow$ “ Vorausgesetzt, die Funktion BESCHRIFTEANHANDP beschriftet  $v$  mit  $(w, m)$ . Zu zeigen ist, dass  $v$  erreichbare Nachfolger hat und dass alle Nachfolger entsprechend beschriftet sind.

Damit  $v$  mit  $(w, m)$  beschriftet werden konnte, musste die Funktion BESTIMMEKANAELEP unter anderem den Nachrichtenkanal  $m$  zurückgeliefert haben. Damit ist aber auch klar, dass wenigstens ein erreichbarer Nachfolger existiert.

Nun zur Beschriftung aller erreichbaren Nachfolger:

1. Kann  $P$  eine Nachricht  $!a$  senden, also  $[v, !a, v'] \in T_P$ , dann ist  $v'$  beschriftet mit  $(w, m + \{a\})$ , da die Methode KEINDEADLOCKP sonst in Zeile 5 **false** zurückgeliefert hätte.
2. Kann  $P$  eine Nachricht  $?a$  empfangen, die im Nachrichtenkanal enthalten ist, also  $[v, ?a, v'] \in T_P$  und  $e \in m$ , dann ist  $v'$  beschriftet mit  $(w, m - \{a\})$ , da die Methode KEINDEADLOCKP sonst in Zeile 2 **false** zurückgeliefert hätte.
3. Kann  $P$  eine  $\tau$ -Transition ausführen, also  $[v, \tau, w'] \in T_P$ , dann ist  $v'$  beschriftet mit  $(w, m)$ , da die Methode KEINDEADLOCKP sonst in Zeile 8 **false** zurückgeliefert hätte.

„ $\Leftarrow$ “ Vorausgesetzt, es gibt erreichbare Nachfolger von  $v$  und alle erreichbaren Nachfolger sind entsprechend beschriftet. Zu zeigen ist, dass die Funktion BESCHRIFTEANHANDP  $v$  mit  $(w, m)$  beschriftet.

Da es erreichbare Nachfolger gibt, wird BESTIMMEKANAELP auf jeden Fall auch  $m$  zurückliefern. Ebenso wird die Funktion KEINDEADLOCKP den Wert **true** liefern, da nach Voraussetzung die erreichbaren Nachfolger von  $v$  entsprechend beschriftet sind.

Also fügt BESCHRIFTEANHANDP die Beschriftung  $(w, m)$  zu  $v$  hinzu.  $\square$

Aus diesem Lemma folgt, dass ein Zustand  $v \in Q_P$  korrekt beschriftet wird bezüglich der erreichbaren Nachfolger in  $P$ , falls alle Nachfolger korrekt beschriftet wurden.

Das zweite Lemma ist ähnlich, mit der kleinen Änderung, dass ein Service gemäß  $OG$  nur eine Teilmenge der Nachfolger (und damit auch nur eine Teilmenge der erreichbaren Nachfolger) hat, was wir berücksichtigen müssen.

**Lemma 2** *Die Funktion BESCHRIFTEANHANDOG beschriftet einen Zustand  $v$  genau dann mit  $(w, m)$ ,  $w \in Q_{OG} \setminus \Omega_{OG}$ , wenn in jedem Service  $R \in \text{Comply}^w(OG)$  der Zustand  $w$  erreichbare Nachfolger hat und  $v$  mit allen erreichbaren Nachfolgern von  $w$  und jeweils entsprechendem Nachrichtenkanal beschriftet ist. Formal lautet die Bedingung: Für jede Labelmenge  $A$ , deren Belegung mit **wahr** die Formel  $\phi(w)$  erfüllt, ist die Menge*

$$M_{OG}^A := \{(w', m - \{a\}) \mid !a \in A, [w, !a, w'] \in T_{OG} \text{ und } a \in m\} \cup \\ \{(w', m + \{a\}) \mid ?a \in A, [w, ?a, w'] \in T_{OG}\}$$

nichtleer und für jedes  $(w', m') \in M_{OG}^A$  ist  $v$  mit  $(w', m')$  beschriftet.

**Beweis:**

„ $\Rightarrow$ “ Vorausgesetzt, die Funktion BESCHRIFTEANHANDOG beschriftet  $v$  mit  $(w, m)$ . Zu zeigen ist, dass  $w$  in jedem Service  $R \in \text{Comply}^w(OG)$  erreichbare Nachfolger hat und dass  $v$  mit allen Nachfolgern entsprechend beschriftet ist.

Damit  $v$  mit  $(w, m)$  beschriftet werden konnte, musste die Funktion BESTIMMEKANAELP unter anderem den Nachrichtenkanal  $m$  zurückgeliefert haben und die Funktion KEINDEADLOCKANNOTATION muss **true** zurückgeliefert haben. Das geschieht nur, wenn für mindestens eine Klausel  $\alpha_{i^*} = \bigvee_j \beta_{i^*,j}$  der Formel  $\phi(w) = \bigwedge_i \alpha_i$  gilt: Für alle Label  $\beta_{i^*,j}$  ist der Zustand  $v$  bereits mit dem Nachfolger  $w'$  von  $w$ , der durch die Transition mit dem Label  $\beta_{i^*,j}$  erreicht wird, und entsprechendem Nachrichtenkanal beschriftet.

Nach Definition erfüllender Subautomaten muss nun jeder Service  $R \in \text{Comply}^w(OG)$  mindestens eine Transition von  $w$  mit einem dieser Label  $\beta_j$  besitzen. Somit gibt es stets mindestens einen erreichbaren Nachfolger von  $w$ .

Nun zur Beschriftung aller erreichbaren Nachfolger:

1.  $OG$  kann eine Nachricht  $!a$  senden, also  $[w, !a, w'] \in T_{OG}$ . Dann ist  $v$  beschriftet mit  $(w', m + \{a\})$ , da die Methode KEINDEADLOCKOG sonst **false** zurückgeliefert hätte.

2.  $P$  kann eine Nachricht  $?a$  empfangen, die im Nachrichtenkanal enthalten ist, also  $[w, ?a, w'] \in T_{OG}$  und  $e \in m$ . Dann ist  $v$  beschriftet mit  $(w', m - \{a\})$ , da die Methode KEINDEADLOCKOG sonst **false** zurückgeliefert hätte.

„ $\Leftarrow$ “ Vorausgesetzt, in jedem Service  $R \in \text{Comply}^w(OG)$  gibt es erreichbare Nachfolger  $w'$  von  $w$  und  $v$  ist mit diesen Nachfolgern entsprechend beschriftet. Zu zeigen ist, dass die Funktion BESCHRIFTEANHANDOG  $v$  mit  $(w, m)$  beschriftet.

Da es erreichbare Nachfolger gibt, wird BESTIMMEKANAEOG auf jeden Fall auch  $m$  zurückliefern. Ebenso wird die Funktion KEINDEADLOCKOG den Wert **true** liefern, da nach Voraussetzung ja  $v$  mit den erreichbaren Nachfolger von  $w$  entsprechend beschriftet ist.

Angenommen, die Funktion KEINDEADLOCKANNOTATION würde **false** zurückliefern. Dann gäbe es für jede Klausel  $\alpha_i = \bigvee_j \beta_{i,j}$  der Formel  $\phi(w) = \bigwedge_i \alpha_i$  jeweils mindestens ein Label  $\beta_{i,j_i^*}$ , sodass  $v$  *nicht* mit dem Nachfolger  $w'$ , der durch die Transition mit diesem Label erreicht wird, beschriftet ist. Da nach Voraussetzung aber jeder erreichbare Nachfolger  $w'$  als Beschriftung an  $v$  steht, hat derjenige Service  $R^* \in \text{Comply}^w(OG)$ , der ausschließlich die Transitionen mit den Labeln  $\beta_{i,j_i^*}$  verwendet, *keinen* erreichbaren Nachfolger. Das ist ein Widerspruch zur Voraussetzung, dass jeder Service aus  $\text{Comply}^w(OG)$  erreichbare Nachfolger von  $w$  besitzt.

Also liefert auch die Funktion KEINDEADLOCKANNOTATION den Wert **true** und BESCHRIFTEANHANDP fügt die Beschriftung  $(w, m)$  zu  $v$  hinzu.  $\square$

Der eigentliche Beweis des Satzes gestaltet sich dank der beiden vorhergehenden Lemmata recht einfach. Wir wollen zeigen, dass  $(v, w, m)$  genau dann deadlockfrei ist, wenn der Algorithmus BESCHRIFTE( $OG, P$ ) den Zustand  $v$  mit  $(w, m)$  beschriftet.

**Beweis:** Durch die verschachtelten Aufrufe der Funktionen DFSOG( $P, v$ ) (äußere Rekursion) und DFSP( $v, w$ ) (innere Rekursion) wird gewährleistet, dass die Unterprozeduren BESCHRIFTEANHANDP( $v, w$ ) und BESCHRIFTEANHANDOG( $v, w$ ) erst aufgerufen werden, *nachdem* alle Paare  $(v', w')$  für Nachfolger  $v' \in Q_P$  von  $v$  und Nachfolger  $w' \in Q_{OG}$  von  $w$  abgearbeitet wurden.

Für die Beweisführung benötigen wir daher eine Induktion über die Zustandspaare  $(v, w) \in Q_P \times Q_{OG}$  von den Blättern hin zur Wurzel.

**Induktionsanfang:** Seien  $v \in \Omega_P$  und  $w \in \Omega_{OG}$  Endzustände. Dann ist  $(v, w, m)$  genau für  $m = \emptyset$  ein deadlockfreier Interaktionszustand, und der Algorithmus beschriftet  $v$  mit  $(w, \emptyset)$ .

**Induktionsvor.:** Für alle Nachfolger  $v'$  von  $v$  gilt:  $(v', w, m)$  ist genau dann deadlockfrei, wenn  $v'$  mit  $(w, m)$  beschriftet ist. Und für alle Nachfolger  $w'$  von  $w$  gilt:  $(v, w', m)$  ist genau dann deadlockfrei, wenn  $v$  mit  $(w', m)$  beschriftet ist.

**Induktionsbeweis:** z.z.:  $(v, w, m)$  für  $v \in Q_P$  und  $w \in Q_{OG}$  ist genau dann deadlockfrei, wenn  $v$  mit  $(w, m)$  beschriftet wird.

„ $\Rightarrow$ “ Eine Möglichkeit, weshalb  $(v, w, m)$  ein deadlockfreier Interaktionszustand sein kann, ist, dass es für jeden Interaktionspartner  $R \in \text{Comply}^w(OG)$  von  $P$  einen Nachfolgezustand  $w' \in Q_R$  von  $w$  gibt, sodass ein deadlockfreier Interaktionszustand  $(v, w', m')$  existiert. Nach Induktionsannahme ist  $v$  dann bereits mit  $(w', m)$  beschriftet. Deshalb fügt, laut Lemma 2, die Funktion  $\text{BESCHRIFTEANHANDOG}(v, w)$  die Beschriftung  $(w, m)$  zu  $v$  hinzu.

Die andere Möglichkeit für die Deadlockfreiheit von  $(v, w, m)$  ist, dass  $v$  einen Nachfolger  $v' \in Q_P$  besitzt, sodass ein deadlockfreier Interaktionszustand  $(v', w, m')$  existiert. Analog muss es bereits die Beschriftung  $(w, m')$  für  $v'$  geben und laut Lemma 1 fügt die Funktion  $\text{BESCHRIFTEANHANDP}(v, w)$  die Beschriftung  $(w, m)$  zu  $v$  hinzu.

„ $\Leftarrow$ “ Wurde die Beschriftung  $(w, m)$  zu  $v$  hinzugefügt, so ist dies durch den Aufruf  $\text{BESCHRIFTEANHANDOG}(v, w)$  oder  $\text{BESCHRIFTEANHANDP}(v, w)$  geschehen.

Im ersten Fall bedeutet dies laut Lemma 2, dass für alle Services  $R \in \text{Comply}^w(OG)$  ein erreichbarer Nachfolger  $w' \in Q_R$  von  $w$  existiert und  $v$  bereits mit all diesen Zuständen und jeweils entsprechendem Nachrichtenkanal  $m'$  beschriftet ist. Nach Induktionsvoraussetzung sind die jeweiligen Interaktionszustände  $(v, w', m')$  auch deadlockfrei, womit die Deadlockfreiheit von  $(v, w, m)$  folgt.

Der zweite Fall gewährleistet nach Lemma 1, dass  $v$  erreichbare Nachfolger hat und all diese mit  $w$  und entsprechendem Nachrichtenkanal  $m'$  beschriftet sind. Analog sind die Interaktionszustände  $(v', w, m')$  nach Induktionsvoraussetzung deadlockfrei und folglich ebenso  $(v, w, m)$ .

□

## 5 Berechnen einer Laufzeitersetzung

Den Algorithmus zum Beschriften von Service Automaten aus dem vorherigen Abschnitt können wir nun verwenden, um gültige Laufzeitersetzungen von einem Service  $P_1$  zu einem Service  $P_2$  bezüglich einer gegebenen Operating Guideline  $OG$  zu bestimmen.

$\text{LAUFZEITERSETZUNG}(\text{Service } P_1, \text{Service } P_2, \text{Operating Guideline } OG)$

(1)  $B_1 := \text{BESCHRIFTE}(P_1, OG)$

(2)  $B_2 := \text{BESCHRIFTE}(P_2, OG)$

(3) **return**  $\mathcal{T} = \{[v_1, \tau, v_2] \mid v_1 \in Q_{P_1}, v_2 \in Q_{P_2}, B_1(v_1) \subseteq B_2(v_2)\}$

Aufgrund der Korrektheit und Vollständigkeit der berechneten Beschriftungen ist die folgende Aussage nicht schwer zu beweisen.

**Satz 2** *Die vom Algorithmus  $\text{LAUFZEITERSETZUNG}$  berechnete Laufzeitersetzung  $\mathcal{T}$  ist gültig.*

**Beweis:** Sei  $R$  ein beliebiger Service gemäß  $OG$ . Zu zeigen ist laut Definition von *Laufzeitersetzung*, dass  $(P_1 \cup \mathcal{T} \cup P_2) \oplus R$  schwach terminiert, sobald  $P_1 \oplus R$  schwach terminiert.

Nehmen wir also an, dass  $P_1 \oplus R$  schwach terminiert, aber  $(P_1 \cup \mathcal{T} \cup P_2) \oplus R$  einen Deadlock besitzt. Laut Annahme muss ein Zustand aus  $Q_{P_2}$  an diesem Deadlock beteiligt sein. Sei daher  $[v_1, \tau, v_2]$ ,  $v_1 \in Q_{P_1}$ ,  $v_2 \in Q_{P_2}$ , jene  $\tau$ -Transition, die bei der Interaktion von  $R$  mit  $(P_1 \cup \mathcal{T} \cup P_2)$  verwendet wird.

Der Zustand der Interaktion zum Zeitpunkt nach Verwendung dieser  $\tau$ -Transition sei  $(v_2, w, m)$ . Da der Zustand nach Annahme nicht deadlockfrei ist, findet der Algorithmus  $\text{BESCHRIFTE}(P_2, OG)$  die Beschriftung  $(w, m)$  für  $v_2$  *nicht*. Hingegen findet der Algorithmus  $\text{BESCHRIFTE}(P_1, OG)$  die Beschriftung  $(w, m)$  für  $v_1$ , da  $P_1 \oplus R$  deadlockfrei ist. Dann gilt aber  $B(v_1) \not\subseteq B(v_2)$ , was im Widerspruch zur Existenz der Transition  $[v_1, \tau, v_2]$  steht.

Somit ist unsere Annahme falsch und  $(P_1 \cup \mathcal{T} \cup P_2) \oplus R$  terminiert schwach.  $\square$

Aus der Vollständigkeit des Beschriftungsalgorithmus' und der Tatsache, dass die  $\tau$ -Transitionen nur in eine Richtung zeigen, ergibt sich folgendes Korollar.

**Korollar 1** *Die vom Algorithmus LAUFZEITERSETZUNG berechnete Laufzeitersetzung ist eindeutig und maximal.*

Mit dem Algorithmus zur Bestimmung der Laufzeitersetzung zweier Service Automaten sind wir nun auch in der Lage, die Frage nach der (Bedienbarkeits-)Äquivalenz zu beantworten. Damit erweitern wir für kreisfreie Service Automaten die Resultate von [Ric02] und [Bre07].

**Proposition 2** *Zwei kreisfreie Service Automaten  $P_1$  und  $P_2$  sind genau dann (bedienbarkeits-)äquivalent für alle Strategien gemäß  $OG$ , wenn die von  $\text{LAUFZEITERSETZUNG}(P_1, P_2, OG)$  berechnete Menge die  $\tau$ -Transition  $[q_{0_{P_1}}, q_{0_{P_2}}]$  und die von  $\text{LAUFZEITERSETZUNG}(P_2, P_1, OG)$  berechnete Menge die  $\tau$ -Transition  $[q_{0_{P_2}}, q_{0_{P_1}}]$  beinhalten. Dabei seien  $q_{0_{P_1}}$  und  $q_{0_{P_2}}$  die Startzustände der beiden Services.*

## 6 Fazit

In dieser Arbeit haben wir das Konzept gültiger Laufzeitersetzungen für Service Automaten zu einer gegebenen Operating Guideline eingeführt. Damit erweitern wir den in [Mar03] eingeführten Begriff der (Bedienbarkeits-)Äquivalenz zweier Service Automaten. Im Gegensatz zu den in [Ric02, Bre07] untersuchten Ersetzungsmöglichkeiten, die sich auf die Entwurfszeit beschränken, betrachten wir die Ersetzung zur Laufzeit, also während einer Interaktion.

Ein besonderes Augenmerk lag auf kreisfreien Service Automaten, da wir für diese Modelle einen Algorithmus angeben konnten, der in der Lage ist, folgendes Problem zu lösen: Gegeben zwei Service Automaten  $P_1, P_2$  und eine Operating Guideline  $OG$ , soll eine maximale Laufzeitersetzung der beiden Services berechnet werden, die während der Interaktion mit einem Service  $R$  gemäß  $OG$  den Service  $P_1$  gegen  $P_2$  austauscht. Tatsächlich ist für kreisfreie Service Automaten das vorliegende Resultat – wie Proposition 2 zeigt – eine echte Verallgemeinerung der Ersetzungsbegriffe in [Ric02, Bre07].

Als Erweiterung dieser Arbeit könnten auch nicht-kreisfreie Service Automaten untersucht werden. Ebenso ist es denkbar, das Konzept der Laufzeiterweiterung auf andere Modelle zur Beschreibung von Prozessen, wie z.B. offene Workflow-Netze (s. [MS05b]) zu übertragen.

## Literatur

- [Bre07] Jan Bretschneider. Produktbedienungsanleitungen zur Charakterisierung austauschbarer Services. Diplomarbeit, Humboldt-Universität zu Berlin, 2007.
- [Mar03] Axel Martens. Verteilte Geschäftsprozesse - Modellierung und Verifikation mit Hilfe von Web Services. Dissertation, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2003. Erschienen in Wiki: Stuttgart, Berlin & Paris.
- [MRS05] Peter Massuthe, Wolfgang Reisig, and Karsten Schmidt. An Operating Guideline Approach to the SOA. *Annals of Mathematics, Computing & Teleinformatics*, 1(3):35-43, 2005.
- [MS05a] Peter Massuthe und Karsten Schmidt. Operating Guidelines - an Automata-Theoretic Foundation for the Service-Oriented Architecture. In Kai-Yuan Cai, Atsushi Ohnishi, and M.F. Lau, editors, *Proceedings of the Fifth International Conference on Quality Software (QSIC 2005)*, Melbourne, Australia, pages 452-457, September 2005. IEEE Computer Society.
- [MS05b] Peter Massuthe and Karsten Schmidt. Operating Guidelines for Services. In Karsten Schmidt and Christian Stahl, editors, *12. Workshop Algorithmen und Werkzeuge für Petrinetze (AWPN 2005)*, *Proceedings*, pages 78-83, September 2005. Humboldt-Universität zu Berlin.
- [MS05c] Peter Massuthe und Karsten Schmidt. Matching Nondeterministic Services with Operating Guidelines, *Informatik-Berichte 193*, Humboldt-Universität zu Berlin, 2005.
- [MW06] Peter Massuthe und Karsten Wolf. An Algorithm for Matching Nondeterministic Services with Operating Guidelines. In Frank Leymann, Wolfgang Reisig, Satish R. Thatte, and Wil M. P. van der Aalst, editors, *The Role of Business Processes in Service Oriented Architectures*, number 06291 of *Dagstuhl Seminar Proceedings*, 2006. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.
- [Ric02] Wolf Richter. Spezifikation und Implementation organisationsübergreifender Geschäftsprozesse mit Petrinetzen. Diplomarbeit, Humboldt-Universität zu Berlin, 2002