

Studienarbeit

Zielgerichtete Strategien

Alexander Schulz

Hulmboldt-Universität zu Berlin

Unter den Linden 6

D-10099 Berlin

aschulz@informatik.hu-berlin.de

1. Juli 2007

Zusammenfassung

Es werden Services betrachtet, die mittels offener Workflownetze (oWFNs) modelliert werden. Für diese Services werden *zielgerichtete Strategien* definiert.

Zielgerichtete Strategien sind Automaten, die sowohl ein oWFN bedienen (d.h. ohne Deadlocks mit ihnen interagieren) als auch ein bestimmtes Ziel verfolgen. Mögliche Ziele sind das Erzwingen (*enforce*) oder Verhindern (*exclude*) von Verhalten seitens des oWFN.

Es werden weiterhin Algorithmen angegeben, um *allgemeinste* zielführende Strategien zu konstruieren, die alle Strategien enthalten, die dieses Ziel verfolgen.

1 Einleitung

Geschäftsprozesse in der Wirtschaft werden immer komplexer, nicht zuletzt durch die Notwendigkeit, auf die Geschäftsprozesse von Vertragspartnern abgestimmt zu sein. Um solche Prozesse auch *automatisiert* überprüfen und optimieren und somit deren Effizienz steigern zu können, wurden verschiedene formale Methoden der Modellierung vorgeschlagen. Ein zentraler Begriff in diesen Modellierungen ist der des *Service*. Ein Service ist eine abstrakte Sicht auf einen Prozess, die lediglich sein Verhalten und die Schnittstelle zur Außenwelt berücksichtigt.

Die in [1] vorgestellte *Service-orientierte Architektur* (SOA) definiert, wie Services miteinander interagieren. Dabei nehmen die Services drei verschiedene Rollen an: *Service-Provider* stellen eine bestimmte Leistung in Form eines Service zur Verfügung, *Service-Requester* sind auf der Suche nach einer Leistung in Form eines Service mit dem sie

interagieren können. Die dritte Rolle ist der *Service-Broker*, dessen Aufgabe es ist, dem Service-Requester diejenigen Service-Provider zuzuordnen, die die gesuchte Leistung anbieten und mit ihm verhaltenskompatibel sind.

Ein Service \mathcal{R} , der sinnvoll mit einem anderen Service \mathcal{P} zusammenarbeitet, heißt *Strategie* für \mathcal{P} , man nennt \mathcal{P} dann bedienbar. Um festzustellen, ob für einen gegebenen Service überhaupt eine Strategie existiert, wurde in [2] ein Verfahren zur Konstruktion einer *allgemeinsten* Strategie angegeben, die alle anderen Strategien enthält. Sie ist darüber hinaus Vorstufe zu der in [3] vorgestellten Bedienungsanleitung, die es dem Service-Broker erlaubt, einem Service-Provider schnell die Service-Requester zuzuordnen, die ihn bedienen.

Bei der Analyse stellt sich jedoch ein Problem: Der Begriff der Bedienbarkeit, wie er in [2] definiert wurde, ist nicht sehr scharf. Ein Service \mathcal{F} wäre danach bedienbar, selbst wenn die einzige fehlerfreie Interaktion mit \mathcal{F} im Abbrechen der Kommunikation bestünde. Die Bedienbarkeit lässt sich also nach dieser Definition nur unabhängig von der Semantik entscheiden; die Frage, ob die Interaktion zweier Services nicht nur fehlerfrei ist, sondern überdies noch das gewünschte Ergebnis liefert, kann nicht beantwortet werden. Gerade dieser Frage kommt jedoch eine hohe Bedeutung zu, wenn man automatisiert Geschäftsprozesse auf ihre Interaktionsfähigkeit prüfen möchte.

Das Problem lässt sich lösen, indem man einen neuen, schärferen Bedienbarkeitsbegriff einführt, der zusätzliche, semantische Forderungen an die Interaktion stellt. In dieser Arbeit sollen zu diesem Zweck *zielgerichtete* Strategien betrachtet werden, die semantische Zielsetzungen verfolgen. Mögliche Ziele sind das Erzwingen (*enforce*) sowie das Verhindern (*exclude*) von Verhalten. Existiert eine zielgerichtete Strategie, ist der Service in Hinblick auf dieses Ziel bedienbar. Darüber hinaus werden analog zu [2] Methoden vorgestellt, die die allgemeinste, ein bestimmtes Ziel verfolgende, Strategie ermitteln.

Der Rest der Arbeit ist wie folgt gegliedert: In Abschnitt 2 werden die formalen Grundlagen zur Modellierung von Service-Providern und Service-Requestern sowie deren Verknüpfung vorgestellt. Darüber hinaus werden wichtige Eigenschaften der Modelle definiert. Daraufhin werden in Abschnitt 3 die möglichen Ziele definiert und jeweils ein Algorithmus zur Konstruktion allgemeinsten Strategien zu den jeweiligen Zielen angegeben. Schließlich werden in Abschnitt 4 eine Zusammenfassung und ein Ausblick auf weitere Entwicklungsmöglichkeiten gegeben.

2 Grundlagen

2.1 Service-Provider

Zur Modellierung eines Service-Providers werden die in [4] vorgestellten offenen Workflownetze (oWFN) benutzt. Es handelt sich bei oWFNs um eine Klasse von Petrinetzen mit ausgezeichneten Kommunikationsplätzen. Jeder Kommunikationsplatz modelliert einen Nachrichtenkanal, über den Nachrichten gesendet oder empfangen werden können. Eine Nachricht wird als Token modelliert, ihr Typ wird durch den Kommunikationsplatz, auf dem sie liegt, bestimmt. Vom Inhalt der Nachricht wird dabei abstrahiert und lediglich ihr Vorhandensein berücksichtigt.

Definition 1 (Petrietz) Ein Petrietz ist ein Tupel $[P, T, F, m_0]$, bestehend aus:

- einer endlichen Menge P von Plätzen,
- einer endlichen Menge T von Transitionen mit $P \cap T = \emptyset$,
- einer Menge F von Bögen mit $F \subseteq (P \times T) \cup (T \times P)$,
- einer Anfangsmarkierung m_0 .

Eine Markierung m ist eine Abbildung $m : P \rightarrow \mathbb{N}$ und ordnet jedem Platz die Anzahl der Token zu, die auf ihm liegen. Für ein $P' \subseteq P$ ist $[P']$ die Markierung, die jedem $p \in P'$ ein Token und allen anderen Plätzen keine Token zuweist. Die Menge aller möglichen Markierungen ist die Menge aller $|P|$ -Tupel natürlicher Zahlen und wird mit \mathbb{N}^P bezeichnet.

Definition 2 (oWFN) Ein offenes Workflownetz (oWFN) ist ein erweitertes Petrietz $N = [P, T, F, in, out, m_0, \Omega]$ mit:

- $[P, T, F, m_0]$ ist ein Petrietz,
- einer Menge von Inputplätzen $in \subseteq P$, so dass für alle $t \in T$ gilt: Wenn $p \in in$, dann $(t, p) \notin F$,
- einer Menge von Outputplätzen $out \subseteq P$, $in \cap out = \emptyset$, so dass für alle $t \in T$ gilt: Wenn $p \in out$, dann $(p, t) \notin F$,
- einer Menge von Endmarkierungen Ω ,
- ist $m \in \Omega$, so existiert kein $t \in T$, das in m schalten kann (siehe Definition 4).

Die Menge $in \cup out$ heißt *Interface* von N und stellt die Schnittstelle des oWFN zu anderen Services dar. Das Petrietz, das entsteht, wenn man die Interfaceplätze samt anhängender Bögen aus N entfernt, heißt das *Innere* von N , notiert als $inner(N)$. Der letzte Punkt der Definition 2 spiegelt die Intuition wieder, dass ein oWFN nicht aus einem Endzustand heraus von selbst in Aktion geraten sollte.

Grafisch werden Plätze durch Kreise, Transitionen durch Quadrate repräsentiert. Die Bögen aus F werden als Pfeile dargestellt und Markierungen durch $m(p)$ Token (schwarzen Punkten) auf p . Das oWFN wird von einem gestrichelten Rechteck umgeben, auf dessen Rand die Interfaceplätze liegen.

Abbildung 1 zeigt eine oWFN-Modellierung für die Bestellabwicklung eines Online-Versandhandels. Nach dem Aufgeben der Bestellung (Interfaceplatz B) hat der Kunde die Wahl, sich als neuer Kunde zu registrieren (Interfaceplatz N) oder als bereits bekannter Kunde zu authentifizieren (K). Registrierte Kunden haben die Möglichkeit, zwischen den Zahlungsmethoden Visa (V) oder Nachnahme (C) zu wählen. Neue Kunden haben nur die Möglichkeit C . Danach wird dem Kunden die Zahlungsbestätigung L übermittelt. Die Menge der Endmarkierungen ist in diesem Beispiel $\Omega = \{[p_7], [p_8], [p_9]\}$.

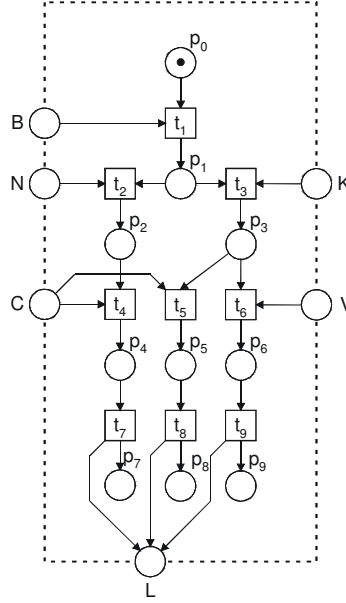


Abbildung 1: Die Bestellabwicklung eines Online-Versandhandels als oWFN O .

Definition 3 (Vorbereich, Nachbereich, Überdecken) Gegeben sei ein oWFN $N = [P, T, F, in, out, m_0, \Omega]$. Seien m, m' Markierungen von N und $t \in T$. Dann gilt:

- die Menge $\bullet t = \{p \mid (p, t) \in F\}$ heißt Vorbereich von t ,
- die Menge $t^\bullet = \{p \mid (t, p) \in F\}$ heißt Nachbereich von t ,
- m überdeckt m' ($m \geq m'$), wenn für alle $p \in P$ gilt: $m(p) \geq m'(p)$.

Somit lässt sich das Verhalten von oWFNs analog zu dem von Petrinetzen wie folgt definieren:

Definition 4 (Schalten) Eine Transition $t \in T$ kann in m schalten, wenn $m \geq [\bullet t]$ gilt. Schaltet t in m ($m \xrightarrow{t} m'$), so entsteht eine neue Markierung m' , die wie folgt definiert ist:

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

$m \xrightarrow{t} m'$ wird als Schritt bezeichnet; m' ist von m aus erreichbar. Die Menge aller von m_0 aus in beliebig vielen Schritten erreichbaren Markierungen wird $R_N(m_0)$ genannt.

Ausgehend von dieser Definition lässt sich für ein Petrinetz eine Struktur angeben, die dessen Verhalten widerspiegelt, der *Erreichbarkeitsgraph*. Die Knoten dieses Graphen sind die Markierungen aus $R_N(m_0)$, die Schritte von N sind die Kanten des Graphen.

Um eine endliche Interaktion mit dem oWFN garantieren zu können, werden im Folgenden zusätzliche Forderungen an ein oWFN gestellt:

- (1) der Erreichbarkeitsgraph von $inner(N)$ ist zyklensfrei,
- (2) keine Transition von $inner(N)$ hat einen leeren Vorbereich.

Definition 5 (Deadlocks) *Gegeben sei ein oWFN $N = [P, T, F, in, out, m_0, \Omega]$. Eine Markierung m , in der keine Transition schalten kann, heißt Deadlock. Ein Deadlock m heißt interner Deadlock, wenn folgende Bedingungen erfüllt sind:*

- $m \notin \Omega$,
- für alle $p \in out$ gilt $m(p) = 0$,
- für alle $t \in T$ existiert ein $p \in (\bullet t \setminus in)$ mit $m(p) = 0$.

Ein Deadlock, der weder ein interner Deadlock noch eine Endmarkierung ist, heißt externer Deadlock.

Bei einem internen Deadlock existiert also für jede Transition $t \in T$ mindestens ein Platz des Inneren von N , der im Vorbereich von t liegt und nicht markiert ist. Damit kann keine Transition schalten, auch wenn das oWFN eine Nachricht empfängt und somit ein Token auf einem Inputplatz generiert wird. Da außerdem gefordert wird, dass auf den Outputplätzen keine Token liegen, ist ein interner Deadlock permanent. Externe Deadlocks lassen sich dagegen durch Nachrichten auflösen. So ist in dem oWFN von Abbildung 1 die Anfangsmarkierung $[p_0]$ ein externer Deadlock. Das Senden der Nachricht B an das oWFN hebt diesen Deadlock auf. Die Markierung $[p_7, V]$ hingegen ist ein interner Deadlock: $[p_7, V]$ ist keine Endmarkierung und das Token auf V kann nicht abgeräumt werden, da kein Token auf p_3 liegt.

2.2 Service-Requester

Die Kommunikationspartner der so beschriebenen Service-Provider, die Service-Requester, werden in dieser Arbeit nicht durch oWFNs modelliert, sondern durch die in [5] eingeführten *Service-Automaten*. Die in den folgenden Kapiteln vorgestellten Konstruktionen orientieren sich an einzelnen *Zuständen* des Requesters. Es ist möglich, oWFNs aus einem Zustandsraum zu konstruieren, doch die Verwendung eines zustandsorientierten Modells wie des Service-Automaten macht diesen Schritt unnötig.

Definition 6 (Service-Automat) *Ein Service-Automat ist ein nichtdeterministischer Automat $A = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ mit:*

- einer Menge I_{in} eingehender Nachrichtenkanäle,
- einer Menge I_{out} ausgehender Nachrichtenkanäle mit $I_{in} \cap I_{out} = \emptyset$,

- einer Menge Q von Zuständen,
- einer Menge $E \subseteq Q \times L \times Q$ von Zustandsübergängen, mit $L = \{?x \mid x \in I_{in}\} \cup \{!x \mid x \in I_{out}\} \cup \{\tau\}$,
- einem Anfangszustand $q_0 \in Q$,
- einer Menge von Endzuständen $Q_\Omega \subseteq Q$, so dass für jedes $q \in Q_\Omega$ gilt: aus $[q, l, q'] \in E$ folgt $l \in \{?x \mid x \in I_{in}\}$.

L wird Menge der *Label* von A genannt. Ein Label $?x$ kennzeichnet ein Empfangereignis, während $!x$ für ein Sendeereignis steht. Um die Richtung der Kommunikation eindeutig zu halten, wird diese *immer* aus Sicht der *Umgebung* des oWFN, d.h. des Service-Requesters angegeben.

Genau wie oWFNs sollen auch Service-Automaten nicht ohne Einfluss der Umgebung aus einem Endzustand in einen anderen Zustand übergehen. Deswegen wird in Definition 6 gefordert, dass Endzustände lediglich durch Empfangsereignisse verlassen werden können.

Im Folgenden werden nur Service-Automaten $A = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ betrachtet, die zyklensfrei sind und somit die transitive Hülle von E irreflexiv ist.

Zustände werden grafisch durch Punkte dargestellt, die mit der Bezeichnung des Zustandes beschriftet sind. Pfeile zwischen Zuständen repräsentieren Zustandsübergänge und sind mit dem zugehörigen Label beschriftet.

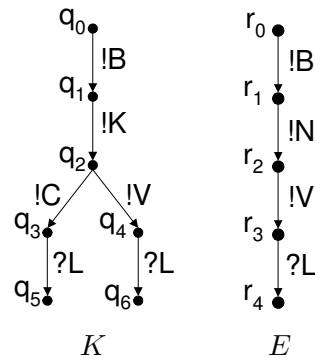


Abbildung 2: Service-Automaten K und E .

Der in Abbildung 2 dargestellte Service-Automat K modelliert einen Bestandskunden für den Online-Versandhandel aus Abbildung 1. Die Menge der Endzustände sei $Q_\Omega = \{[q_5], [q_6]\}$.

2.3 Kommunikation

Um eine Kommunikation zwischen Provider und Requester abbilden zu können, muss nun die *Interaktion* eines oWFN \mathcal{P} mit einem Service-Automaten \mathcal{R} definiert werden.

Zu diesem Zweck werden \mathcal{R} und \mathcal{P} zu einem *Interaktionssystem* verschmolzen, dessen Verhalten wie folgt definiert ist:

Definition 7 (Interaktionssystem) Gegeben sei ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$ mit $I_{in} = out$ und $in = I_{out}$.

$\mathcal{R} \oplus \mathcal{P} = [S, X, s_{ini}, S_\Omega]$ ist ein Interaktionssystem mit Situationen $S \subseteq Q \times \mathbb{N}^P$ und einer Menge von Aktionen $X \subseteq S \times (T \cup E \cup \{\tau\}) \times S$, die folgendermaßen induktiv definiert sind:

- Die Anfangssituation ist $s_{ini} = [q_0, m_0] \in S$,
- ist $s = [q, m] \in S$ und es existiert eine Transition $t \in T$, so dass $m \xrightarrow{t} m'$, dann ist $s' = [q, m'] \in S$ und $[s, t, s'] \in X$,
- ist $s = [q, m] \in S$ und es existiert ein Zustandsübergang $e = [q, \tau, q'] \in E$, dann ist $s' = [q', m] \in S$ und $[q, \tau, q'] \in X$,
- ist $s = [q, m] \in S$, $a \in in$, $m(a) > 0$ und es existiert ein Zustandsübergang $e = [q, ?a, q'] \in E$, dann ist $s' = [q', m - a] \in S$ und $[q, e, q'] \in X$,

$$\text{dabei ist } (m - a)(p) = \begin{cases} m(p) & p \neq a, \\ m(p) - 1 & p = a \text{ und } m(p) > 0, \\ 0 & \text{sonst.} \end{cases}$$

- ist $s = [q, m] \in S$, $a \in out$ und es existiert ein Zustandsübergang $e = [q, !a, q'] \in E$, dann ist $s' = [q', m + a] \in S$ und $[q, e, q'] \in X$,

$$\text{dabei ist } (m + a)(p) = \begin{cases} m(p) & p \neq a, \\ m(p) + 1 & p = a. \end{cases}$$

Sei $a = [s, l, s'] \in X$. Dann überführt die mit l beschriftete Aktion a die Situation s in die Situation s' .

Die Menge S_Ω der Endsituationen enthält alle $s = [q, m] \in S$, für die $q \in Q_\Omega$ und $m \in \Omega$ gilt. Eine Situation $[q, m] \notin S_\Omega$ ohne Nachfolger heißt Deadlock-Situation.

Im Folgenden wird immer davon ausgegangen, dass bei der Komposition eines Service-Automaten R mit einem oWFN P die Bedingung $I_{in} = out$ und $I_{out} = in$ erfüllt wird.

Die Situationen eines Interaktionssystems werden grafisch durch ein Tupel aus Zustand des Service-Automaten und der Markierung des oWFN repräsentiert, wobei der Index der Situation zur besseren Übersicht als Index an das Tupel geschrieben wird. Eine Aktion $a = [s, l, s']$ wird als mit l beschrifteter Pfeil von s nach s' dargestellt. Abbildung 3 zeigt das aus dem Requester K (Abbildung 2) und dem oWFN O (Abbildung 1) gebildete Interaktionssystem.

Der in Abbildung 2 gezeigte Service-Automat E interagiert mit dem oWFN O aus Abbildung 1 offensichtlich fehlerhaft, da er trotz der Angabe, Neukunde zu sein (!N), auf Zahlung per Visa-Karte (!V) besteht. Als Folge kann keine der beiden Komponenten O und E zu einer Endmarkierung bzw. in einen Endzustand gelangen – das Interaktionssystem $E \oplus O$ verharrt in der Deadlock-Situation $[r_3, [r_3, V]]$.

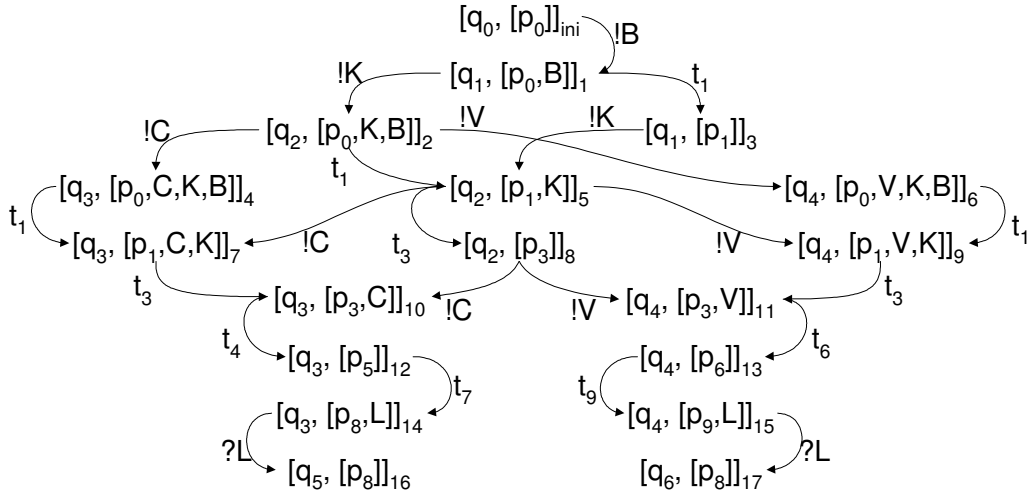


Abbildung 3: Das Interaktionssystem $K \oplus O$ mit den Endsituationen $s_{16} = [q_5, [p_8]]$ und $s_{17} = [q_6, [p_8]]$.

Der Zweck eines Interaktionssystems ist die Modellierung der Kommunikation zwischen Requester und Provider. Da eine Kommunikation meistens aus einer Abfolge von Nachrichten besteht, ist es sinnvoll, eine solche Abfolge im Interaktionssystem identifizieren zu können:

Definition 8 (Ablauf, Teilablauf) Gegeben sei ein Interaktionssystem $G = [S, X, s_{ini}, S_\Omega]$. Ein Ablauf von G ist eine Reihe von Aktionen a_1, a_2, \dots, a_n mit $a_i = [s_{i-1}, l_i, s_i] \in X$, so dass $s_n \in S_\Omega$ und $s_0 = s_{ini}$. Ein Teilablauf von G ist ein Anfangsstück a_1, a_2, \dots, a_l der Länge l eines Ablaufes.

Ein Ablauf ist somit eine Kette von Aktionen von G , die die Startsituation in eine Endsituation überführt. Betrachtet man das Beispiel des Service-Automaten E aus Abbildung 2, dann hat das Interaktionssystem $E \oplus O$ somit keinen Ablauf (da $[r_3, [p_3, V]]$ eine Deadlock-Situation ist), aber sowohl z.B. $!B$ als auch $!B, t_1$ sind Teilabläufe von $E \oplus O$.

Für eine gegebene Situation s eines Interaktionssystems G kann nicht eindeutig angegeben werden, welcher Teilablauf von G s_{ini_G} in s überführt. In Abschnitt 3.2 ist es aber nötig, entscheiden zu können, ob in jedem Ablauf von G , der in einer Endsituation s endet, ein bestimmtes Ereignis eingetreten ist. Zu diesem Zweck wird nun eine erweiterte Struktur, der *Ablaufbaum*, definiert:

Definition 9 (Ablaufbaum) Gegeben sei ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein *oWFN* $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$. Sei $G = \mathcal{P} \oplus \mathcal{R} = [S, X, s_{ini}, S_\Omega]$. Der Ablaufbaum von G ist ein Graph $AB = [V_{AB}, E_{AB}, v_{ini}, mark]$ mit:

- der Menge V der Knoten,

- der Menge $E_{AB} \subseteq V \times (T \cup E \cup \{\tau\}) \times V$ der Kanten,
- dem Startknoten v_{ini} ,
- einer Abbildung $mark : V \rightarrow \mathbb{N}^P$.

Die Abbildung $mark$ ordnet jedem Knoten eine Markierung des oWFN zu, die Relation \sim ordnet jedem Knoten die entsprechende Situation des Interaktionssystems zu. V, E und $mark$ sind wie folgt induktiv definiert:

- $v_{ini} \in V$ mit $mark(v_{ini}) = m_0$; v_{ini} entspricht s_{ini} ($v_{ini} \sim s_{ini}$)
- Sei $v \in V$, $v \sim s$. Dann gilt für jede Aktion $a = [s, l, s'] \in X$, $s' = [q', m']$:
 - (a) $v' \in V$ mit: $v' \sim s'$, $mark(v') = m'$,
 - (b) $e = [v, l, v'] \in E_{AB}$,
 - (c) $\neg \exists v'' \neq v, l : [v'', l, v'] \in E_{AB}$.

$[v, l, v']$ stellt einen Übergang von v nach v' dar, v' heißt Nachfolger von v ; v Vorgänger von v' .

Die Wahl einer Baumstruktur bewirkt, dass tatsächlich für jeden Knoten v der Teilablauf aus G rekonstruierbar ist, der G in die Situation s mit $v \sim s$ überführt. Dieser Teilablauf heißt *Geschichte von v* .

Definition 10 (Geschichte von v) Gegeben seien ein Service-Automat \mathcal{R} und ein oWFN \mathcal{P} . Sei weiterhin $G = \mathcal{R} \oplus \mathcal{P}$ das entstehende Interaktionssystem, AB_G sein Ablaufbaum und v ein Knoten von AB_G .

Ein Ablauf $h(v) = a_1, \dots, a_n$, $a_i = [s_{i-1}, l_i, s_i]$ heißt Geschichte von v , wenn folgende Bedingungen gelten:

- $s_0 = s_{ini}$,
- für alle s_i existiert ein $v_i \in V$ mit $v_i \sim s_i$,
- $v_n = v$,
- für jedes a_i existiert ein $e_i = [v_{i-1}, l_i, v_i] \in E_{AB}$.

Jeder Ablauf von G findet sich als Geschichte eines Knotens in AB_G wieder und jeder Knoten von AB_G hat als Geschichte einen Teilablauf von G . Der Ablaufbaum $AB_{K \oplus O}$ des Interaktionssystems aus Abbildung 3 beinhaltet 10 Abläufe mit insgesamt 56 Knoten. Abbildung 4 zeigt ein weniger umfangreiches Beispiel.

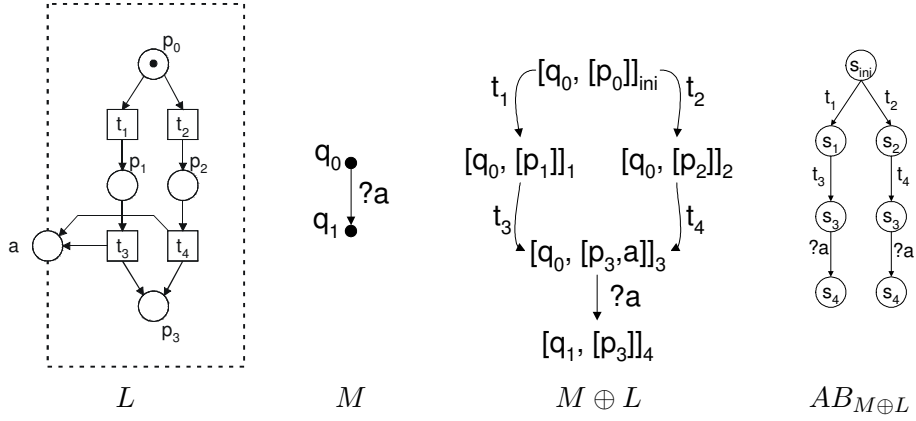


Abbildung 4: Beispiel eines Ablaufbaumes.

2.4 Eigenschaften

Um über die vorgestellten Konstrukte argumentieren zu können, werden nun verschiedene Eigenschaften von oWFN, Service-Automaten und Interaktionssystemen definiert:

Definition 11 (schwach terminierend, Strategie, bedienbar) Gegeben sei ein oWFN \mathcal{P} und ein Service-Automat \mathcal{R} . Dann gilt:

- $\mathcal{R} \oplus \mathcal{P}$ heißt schwach terminierend, wenn es keine Deadlock-Situationen enthält,
- Jeder Service-Automat \mathcal{R} , für den $\mathcal{R} \oplus \mathcal{P}$ schwach terminiert, heißt Strategie für \mathcal{P} ,
- \mathcal{P} ist bedienbar gdw. es eine Strategie \mathcal{R} für \mathcal{P} gibt.

Es ist nur sinnvoll, Service-Automaten zu betrachten, die zusammenhängend sind, d.h. dass jeder Zustand mit Ausnahme von q_0 mindestens einen Vorgänger hat. Gilt darüber hinaus noch, dass jeder Zustand höchstens einen Vorgänger hat, so ist der Service-Automat *baumförmig*. Ein Service-Automat A heißt *deterministisch*, wenn E keine mit dem Label τ versehene Zustandsübergänge enthält und kein $q \in Q$ existiert, so dass von q zwei mit dem gleichen Label beschriftete Zustandsübergänge ausgehen.

Deterministische Service-Automaten, die außerdem noch baumförmig sind, haben unter anderem die Eigenschaft, dass es für ein *beobachtbares* Verhalten einen kanonischen Vertreter gibt, was für die Konstruktionen im Folgenden wichtig ist. Deswegen verlangen wir dass Service-Automaten deterministisch sind und in Baumform vorliegen; nicht-baumförmige Automaten lassen sich leicht in diese überführen.

Unter Berücksichtigung dieser Forderungen lässt sich nun eine weitere wichtige Eigenschaft von Service-Automaten definieren:

Definition 12 (allgemeinste Strategie) Gegeben sei ein oWFN \mathcal{P} . Ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ ist die allgemeinste Strategie für \mathcal{P} , wenn \mathcal{R} Strategie für \mathcal{P} ist und für alle Strategien $\mathcal{R}' = [I_{in_{\mathcal{R}'}, I_{out_{\mathcal{R}'}, Q_{\mathcal{R}'}, E_{\mathcal{R}'}, q_{0_{\mathcal{R}'}, Q_{\Omega_{\mathcal{R}'}}}]$ für \mathcal{P} gilt: $Q_{\mathcal{R}'} \subseteq Q$.

Das bedeutet, dass alle Strategien Teilbäume der allgemeinsten Strategie sind.

Satz 1 ([2]) Unter allen Service-Automaten in Baumform, die Strategie für ein gegebenes oWFN sind, existiert ein kanonischer Vertreter, die allgemeinste Strategie.

Zur Konstruktion einer allgemeinsten Strategie wurde in [6] eine Funktion benutzt, die einen Zustand q des Requesters auf alle die Zustände abbildet, die oWFN und Requester im Interaktionssystem gemeinsam einnehmen können. Diese *Knowledge* genannte Funktion liefert auch die Mittel, mit der im Folgenden die zielführenden Strategien konstruiert werden.

Definition 13 (Knowledge) Gegeben seien ein Service-Automat \mathcal{R} und ein oWFN \mathcal{P} . Die Knowledge-Funktion ist eine Abbildung $k_{\mathcal{R}, \mathcal{P}} : Q \rightarrow \mathbb{N}^P$, so dass $k_{\mathcal{R}, \mathcal{P}}(q) = \{m \mid [q, m] \in S_{\mathcal{R} \oplus \mathcal{P}}\}$.

Mit Hilfe der Knowledge-Funktion kann die Eigenschaft der Stabilität von Markierungen definiert werden.

Definition 14 (stabile, transiente Markierungen) Gegeben sei ein Service-Automat \mathcal{R} und ein oWFN \mathcal{P} . Sei $q \in Q$. Eine Markierung $m \in k_{\mathcal{R}, \mathcal{P}}(q)$ heißt stabile Markierung, wenn es keine Markierung $m' \in k_{\mathcal{R}, \mathcal{P}}(q)$ gibt, so dass m' von m aus erreichbar ist. Jede Markierung $m'' \in k_{\mathcal{R}, \mathcal{P}}(q)$, die nicht stabil ist, heißt transient.

Stabile Markierungen bleiben demnach so lange im oWFN bestehen, wie der Requester keine neuen Nachrichten sendet oder empfängt (und damit seinen Zustand ändert). Befindet sich das oWFN dagegen in einer transienten Markierung, können Transitionen schalten, ohne dass der Requester darauf Einfluß hat.

Die Knowledge-Funktion erlaubt es auch, Strategien wie folgt zu charakterisieren:

Satz 2 ([2]) Gegeben sei ein oWFN \mathcal{P} . Ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ ist eine Strategie für \mathcal{P} gdw. $Q \neq \emptyset$ und für alle $q \in Q$ die folgenden Bedingungen gelten:

- $k_{\mathcal{R}, \mathcal{P}}(q)$ enthält keine internen Deadlocks
- für jeden externen Deadlock $m \in k_{\mathcal{R}, \mathcal{P}}(q)$ hat das Interaktionssystem $\mathcal{R} \oplus \mathcal{P}$ eine Aktion, die $[q, m]$ in eine andere Situation überführen kann.

Diese Charakterisierung hat zur Folge, dass alle Zustände q von \mathcal{R} , für die $k_{\mathcal{R}, \mathcal{P}}(q) = \emptyset$ gilt, Bestandteil einer Strategie sein können und damit Bestandteil der allgemeinsten Strategie sind. $k_{\mathcal{R}, \mathcal{P}}(q)$ ist für alle Zustände von \mathcal{R} leer, die auf ein Empfangsereignis folgen, das im Interaktionssystem $\mathcal{R} \oplus \mathcal{P}$ nicht auftritt.

Um die grafische Darstellung einer allgemeinsten Strategie übersichtlich zu halten, werden diese Zustände nicht dargestellt, sie sind aber nach wie vor Bestandteil der Strategie.

3 Zielgerichtete Strategien

In diesem Abschnitt werden die verschiedenen allgemeinen Ziele vorgestellt, die eine Strategie erreichen kann. Weiterhin wird gezeigt, wie man bei gegebenem Service-Provider konstruktiv entscheidet, ob dieses Ziel erreichbar ist.

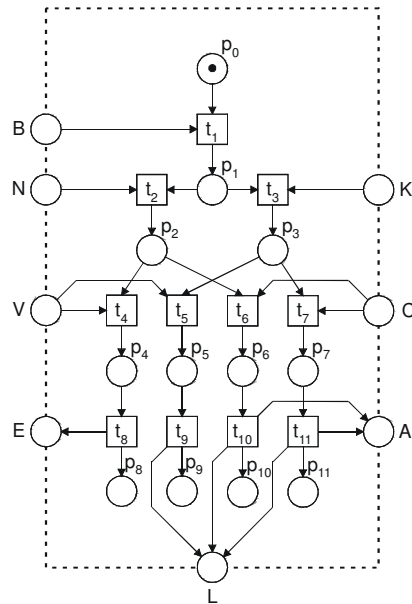


Abbildung 5: Der erweiterte Online-Versandhandel O' .

Abbildung 5 zeigt eine Erweiterung des Beispiels aus Abbildung 1, an dem die einzelnen Ziele erläutert werden: Sollte ein neu registrierter Kunde die Zahlungsoption Visa (Interfaceplatz V) auswählen, so schickt ihm das oWFN nun eine Fehlermeldung E, statt wie bisher in einem Deadlock zu verharren. Außerdem teilt das oWFN allen Kunden, die sich für Nachnahme entschieden haben, mit, dass sie einen Aufpreis zahlen müssen (Interfaceplatz A). Die Menge der Endmarkierungen ist $\Omega_{O'} = \{[p_8], [p_9], [p_{10}], [p_{11}]\}$.

3.1 exclude

Das Ziel *exclude* drückt aus, dass ein bestimmtes Interaktionsverhalten (die Markierung eines Platzes oder das Schalten einer Transition) auf Seiten des Service-Providers vermieden werden soll.

Abbildung 6 zeigt die allgemeinste Strategie für das oWFN aus Abbildung 5. Die mit (a) gekennzeichneten Knoten sind Teile der Strategie, die einen Fehler hervorrufen und die entstehende Fehlermitteilung empfangen. Diese 16 der insgesamt 113 Knoten (also etwa 14%) sind in einer Strategie völlig sinnlos. Sie entstehen dadurch, dass die Fehlerbehandlung *im oWFN* nötig ist und somit in die Berechnung der allgemeinsten Strategie mit einbezogen wird.

Um zu erreichen, dass solche „sinnlosen“ oder einfach unerwünschten Teile nicht in einer Strategie enthalten sind, bedient man sich des Ziels *exclude*. Für den oben beschriebenen Fall wäre die *allgemeinste Strategie unter exclude(E)* gerade die allgemeinste Strategie ohne die mit (a) beschrifteten Knoten.

Das Ziel *exclude* kann wie folgt definiert werden:

Definition 15 (exclude) Gegeben sei ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$. Das Interaktionssystem $G = \mathcal{R} \oplus \mathcal{P}$ terminiere schwach. Dann heißt G

- schwach terminierend unter $exclude(m)$, $m \in \mathbb{N}^P$, falls in keinem Ablauf von G eine Situation $s = [q, m']$ durchlaufen wird, für die $m' \geq m$ gilt,
- schwach terminierend unter $exclude(M)$, $M \subseteq \mathbb{N}^P$, falls für alle $m \in M$ gilt: G terminiert schwach unter $exclude(m)$,
- schwach terminierend unter $exclude(p)$, $p \in P$, falls in keinem Ablauf von G eine Situation $s = [q, m]$ durchlaufen wird, für die $m > [p]$ gilt,
- schwach terminierend unter $exclude(P')$, $P' \subseteq P$, falls für alle $p \in P'$ gilt: G terminiert schwach unter $exclude(p)$,
- schwach terminierend unter $exclude(t)$, $t \in T$, falls kein Ablauf von G eine Aktion enthält, in der t schaltet,
- schwach terminierend unter $exclude(T')$, $T' \subseteq T$, falls für alle $t \in T'$ gilt: G terminiert schwach unter $exclude(t)$.

Die Definition der übrigen Eigenschaften, die für Service-Automaten in Teil 2 definiert wurden, erfolgt analog.

Definition 16 (Strategie, bedienbar, allgemeinste Strategie) Gegeben sei ein oWFN \mathcal{P} und ein Service-Automat \mathcal{R} , \mathcal{R} bediene \mathcal{P} . Dann gilt:

- \mathcal{R} heißt Strategie für \mathcal{P} unter $exclude(X)$, wenn $\mathcal{R} \oplus \mathcal{P}$ unter $exclude(X)$ schwach terminiert,
- \mathcal{P} heißt bedienbar unter $exclude(X)$ gdw. es eine Strategie \mathcal{R} unter $exclude(X)$ für \mathcal{P} gibt,
- \mathcal{R} heißt allgemeinste Strategie unter $exclude(X)$, wenn \mathcal{R} Strategie unter $exclude(X)$ ist und für alle Strategien $\mathcal{R}' = [Q_{\mathcal{R}'}, T_{\mathcal{R}'}, q_{0, \mathcal{R}'}]$ für \mathcal{P} unter $exclude(X)$ gilt: $Q_{\mathcal{R}'} \subseteq Q_{\mathcal{R}}$.

Im Folgenden soll bei der Argumentation über $exclude(X)$ nur noch auf Markierungen m der Situationen $s = [q, m]$ eines Interaktionssystems zurückgegriffen werden. Dazu ist es nötig, die Punkte von Definition 15 auf Überdeckung von Markierungen zurückzuführen.

Lemma 1 Gegeben sei ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$. Dann gilt:

1. das Interaktionssystem $G = \mathcal{R} \oplus \mathcal{P}$ terminiert schwach unter $exclude(p)$, $p \in P$,
gdw. G schwach unter $exclude(\{\{p\}\})$ terminiert,
2. das Interaktionssystem $G = \mathcal{R} \oplus \mathcal{P}$ terminiert schwach unter $exclude(t)$, $t \in T$,
gdw. es schwach unter $exclude(\{\bullet t\})$ terminiert.

Zu (1.) ist leicht zu sehen, dass in jeder Markierung, die $\{\{p\}\}$ überdeckt, p markiert sein muss und umgekehrt jede Markierung, in der p markiert ist, $\{\{p\}\}$ überdecken muss.

Man betrachte zu (2.) eine Aktion $a = [s, t, s']$ von G , $s = [q, m]$, in der t schaltet. Für jedes solche a gilt, dass t in m schalten können muss, mithin $m \geq [\bullet t]$ ist. Gilt nun, dass in keinem Ablauf von G eine Situation $s = [q, m]$ mit $m \geq [\bullet t]$ durchlaufen wird, so kann demzufolge in diesem Ablauf auch keine Aktion enthalten sein, in der t schaltet.

Gibt es umgekehrt keinen Ablauf von G , in dem eine Aktion $a = [s, t, s']$ enthalten ist, so existiert nach Definition 7 keine Situation $s = [q, m] \in S$, in der $m \geq [\bullet t]$ gilt. Somit lässt sich jedes Problem bezüglich $exclude(X)$ auf eines bezüglich $exclude(M)$, $M \subseteq \mathbb{N}^P$, zurückführen, daher werden im Folgenden nur noch diese betrachtet.

Um analog zu Satz 2 Bedingungen an eine Strategie unter $exclude$ formulieren zu können, müssen die Bedingungen aus Definition 15 umformuliert werden. Anstelle von Abläufen müssen die Aussagen über Zustände von \mathcal{R} getroffen werden.

Aus der Definition von Abläufen geht hervor, dass die Menge der Situationen *aller* Abläufe die Menge der von der Anfangssituation aus erreichbaren Situationen ist, und diese ist aufgrund der induktiven Definition gleich $S_{\mathcal{R} \oplus \mathcal{P}}$. Die Menge der mittels der Knowledge-Funktion aus \mathcal{R} abgeleiteten Situationen $\{[q, m] \mid q \in Q, m \in k_{\mathcal{R}, \mathcal{P}}(q)\}$ ist aber, setzt man die Definition von $k_{\mathcal{R}, \mathcal{P}}$ ein, wiederum gleich $S_{\mathcal{R} \oplus \mathcal{P}}$. Somit reicht es aus, die Bedingungen für jedes $q \in Q$ und die dazugehörigen Markierungen $k_{\mathcal{R}, \mathcal{P}}(q)$ zu überprüfen.

Lemma 2 Gegeben sei ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$ und ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$. Dann gilt: \mathcal{R} ist Strategie für \mathcal{P} unter $exclude(M)$, $M \subseteq \mathbb{N}^P$,
gdw. folgende Bedingungen erfüllt sind:

- (a) \mathcal{R} ist Strategie für \mathcal{P} ,
- (b) für alle $q \in Q$ und alle $m \in k_{\mathcal{R}, \mathcal{P}}(q)$ gilt, dass für kein $m' \in M$ $m \geq m'$ erfüllt ist.

Mittels Satz 2 und Lemma 2 ist es nun möglich, analog zu [6] einen Algorithmus anzugeben, der bei gegebenem oWFN \mathcal{P} eine allgemeinste Strategie unter $exclude(M)$ konstruiert.

Ausgangspunkt hierfür ist, dass der Erreichbarkeitsgraph von $inner(\mathcal{P})$ azyklisch ist und daher für jeden Service-Automaten \mathcal{R} der Ablaufbaum von $\mathcal{R} \oplus \mathcal{P}$ endlich ist.

Definition 17 ($depth(\mathcal{P})$) Sei $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$ ein oWFN, und EG' der Erreichbarkeitsgraph des Inneren von \mathcal{P} . Die Menge $T^* = \{t \mid t \in T \wedge (t \bullet \cup \bullet t) \cap (in \cup$

$out) \neq \emptyset\}$ ist die Menge aller Transitionen von \mathcal{P} , die mit einem Interfaceplatz verbunden sind.

Man kann jedem Pfad in EG' eine Zahl zuordnen, die angibt, wie oft eine Transition aus T^* geschaltet hat. Die größte dieser Zahlen sei $depth(\mathcal{P})$

Man kann einen Service-Automaten konstruieren, der auf jedem Pfad die Tiefe $depth(\mathcal{P})$ hat und von jedem Knoten für jedes Label $l \in \{!x \mid x \in in\} \cup \{?x \mid x \in out\}$ ein Sende- bzw. Empfangsereignis ausgeht. Dieser Automat wird mit \mathcal{A} bezeichnet.

In einem deterministischen Service-Automaten \mathcal{R} ist jeder Zustandsübergang mit dem Senden oder Empfangen einer Nachricht verbunden. Da jede Nachricht, die \mathcal{R} sendet, in \mathcal{P} konsumiert werden muss und umgekehrt, kann eine Strategie für \mathcal{P} auch nicht mehr als $depth(\mathcal{P})$ Zustandsübergänge haben. Somit ist jede mögliche Strategie für \mathcal{P} (insbesondere Strategien unter $exclude(M)$) ein Teilautomat von \mathcal{A} .

Algorithmus 1 (exclude) Gegeben sei ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$. Ausgehend von \mathcal{A} , wendet man nun folgenden Algorithmus an, um eine Strategie \mathcal{S} unter $exclude(M)$, $M \subseteq \mathbb{N}^P$ zu konstruieren:

- (I) Streichen aller Zustände $q_{\mathcal{A}} \in Q_{\mathcal{A}}$, bei denen $k_{\mathcal{A}, \mathcal{P}}(q_{\mathcal{A}})$ einen internen Deadlock enthält,
- (II) Streichen aller Zustände $q_{\mathcal{A}} \in Q_{\mathcal{A}}$, bei denen $k_{\mathcal{A}, \mathcal{P}}(q_{\mathcal{A}})$ eine Markierung m' enthält, für die $m' \geq m$ für ein $m \in M$ gilt,
- (III) sukzessives Streichen aller Zustände $q_{\mathcal{A}} \in Q_{\mathcal{A}}$, bei denen $k_{\mathcal{A}, \mathcal{P}}(q_{\mathcal{A}})$ einen externen Deadlock m enthält und die keinen ausgehenden Zustandsübergang haben, der $[q_{\mathcal{A}}, m]$ in eine neue Situation überführt.

Das Streichen eines Zustandes zieht immer auch das Streichen des eingehenden Zustandsübergangs sowie aller ausgehenden Zustandsübergänge und Folgezustände mit sich.

Das sukzessive Streichen im Punkt (III) ist nötig, da durch das Streichen von Zuständen nachfolgende Transitionen wegfallen können und somit neue Zustände ohne Nachfolger entstehen können.

Satz 3 Über Algorithmus 1 lassen sich folgende Aussagen treffen:

1. Sollte die Zustandsmenge des resultierenden Service-Automaten \mathcal{S} leer sein, so ist P nicht bedienbar unter $exclude(M)$. Sonst gilt:
2. \mathcal{S} ist eine Strategie für P unter $exclude(M)$,
3. \mathcal{S} ist sogar allgemeinste Strategie unter $exclude(M)$.

Beweis: Wie schon gezeigt, werden durch den Algorithmus nur solche Zustände entfernt, die nicht Teil einer Strategie sein können. Ist die Zustandsmenge von \mathcal{S} leer, so wurde

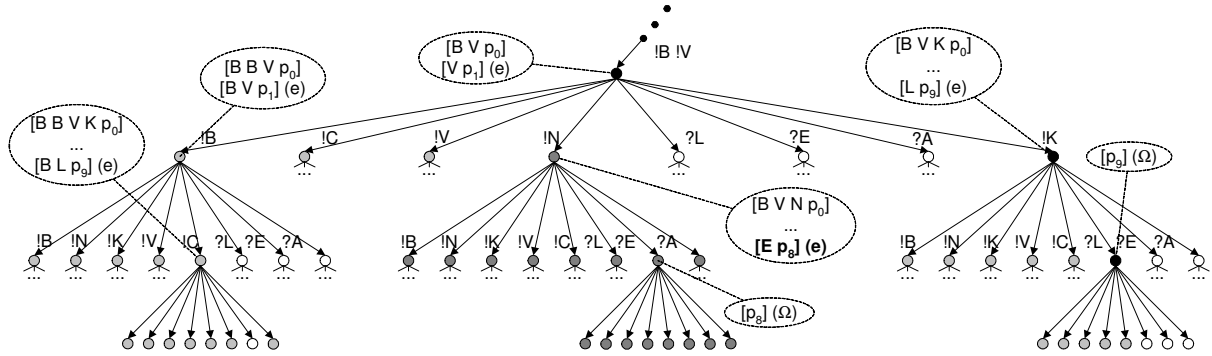


Abbildung 7: Anwendung des Algorithmus an einem Ausschnitt aus \mathcal{A} .

offensichtlich auch der Startzustand mit dieser Begründung gelöscht. Das bedeutet, dass es keine Strategie für \mathcal{P} unter $\text{exclude}(M)$ geben kann und somit \mathcal{P} nicht bedienbar ist.

Zu (2.) In Schritt (II) werden all jene Zustände gelöscht, die Bedingung (b) von Lemma 2 verletzen - und damit zwar Teil einer Strategie für \mathcal{P} sein können, nicht aber einer Strategie für \mathcal{P} unter $\text{exclude}(M)$. Um auch Bedingung (a) zu erfüllen, werden in den Schritten (I) und (III) all jene Zustände gelöscht, die Satz 2 verletzen und damit nicht Teil einer Strategie für \mathcal{P} sind. Damit genügt der resultierende Service-Automat \mathcal{S} per Konstruktion Lemma 2 und ist somit Strategie für \mathcal{P} unter $\text{exclude}(M)$.

Zu (3.) Anhand der Definition von $\text{depth}(\mathcal{P})$ und der Konstruktion von \mathcal{A} ist ersichtlich, dass alle Strategien für \mathcal{P} unter $\text{exclude}(M)$ Teilautomaten von \mathcal{A} sein müssen. Von den im Algorithmus gelöschten Zuständen wurde gezeigt, dass diese *keinesfalls* Teil einer Strategie unter $\text{exclude}(M)$ sein können. Somit ist der resultierende Service-Automat \mathcal{S} kein echter Teilautomat einer anderen Strategie unter $\text{exclude}(M)$, was \mathcal{S} zur allgemeinsten Strategie unter $\text{exclude}(M)$ macht.

Abbildung 7 zeigt einen kleinen Ausschnitt des anhand des oWFN aus Abbildung 5 konstruierten Service-Automaten \mathcal{A} . Anhand dieses Ausschnittes werden die Schritte zur Konstruktion einer allgemeinsten Strategie unter $\text{exclude}(\{[E]\})$ erläutert:

Zustände mit einer leeren Knowledge-Funktion sind weiß markiert, sie erscheinen zwar nicht in der grafischen Darstellung der Strategie, gehören aber sowohl zu \mathcal{A} als auch zur Strategie. Im Beispiel-oWFN könne keine internen Deadlocks auftreten, weswegen Schritt (I) keine Auswirkungen hat. Die Knowledge-Funktion des obersten der dunkelgrau markierten Knoten enthält die Markierung $[p_g, E]$, welche die Markierung $[E]$ überdeckt. Daher wird er (und mit ihm alle seine Kindknoten) in Schritt (II) des Algorithmus gelöscht. Schließlich werden in Schritt (III) die hellgrau markierten Knoten sukzessive gelöscht.

Die in diesem Ausschnitt aufgrund der Forderung $\text{exclude}(E)$ entfernten Knoten finden sich im zweiten mit (a) markierten Bereich in Abbildung 6 wieder.

3.2 enforce

Das Ziel *enforce* soll sicherstellen, dass ein bestimmtes Verhalten auf der Providerseite garantiert eintritt. So könnte zum Beispiel ein Kunde des erweiterten Online-Versandhandels O' aus Abbildung 5 darauf bestehen, mit Visa-Karte zu zahlen. Jedes Verhalten eines Requesters R , das dieser Bedingung genügt und O' bedient, ist eine *Strategie unter enforce(V)* (Definition 19).

Die Bedienbarkeit unter $\text{enforce}(X)$, wie sie in Definition 19 definiert wird, setzt voraus, dass der Requester Einfluss auf das Eintreten des Ereignis X hat. Nur wenn dies der Fall ist, kann es einen Requester geben, der das Ereignis tatsächlich in jedem möglichen Ablauf des Interaktionssystems forciert.

Definition 18 (enforce) *Gegeben seien ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein oWFN $\mathcal{P} = [P, T, F, in, out, \Omega]$. Das kombinierte System $G = \mathcal{R} \oplus \mathcal{P}$ terminiere schwach. G heißt dann*

- schwach terminierend unter $\text{enforce}(m)$, $m \in \mathbb{N}^P$, wenn in jedem Ablauf von G eine Situation $s = [q, m']$ durchlaufen wird, in der $m' \geq m$ gilt,
- schwach terminierend unter $\text{enforce}(M)$, $M \subseteq \mathbb{N}^P$, wenn für jeden Ablauf von G ein $m \in M$ existiert, so dass eine Situation $s = [q, m']$ durchlaufen wird, in der $m' \geq m_\sigma$ gilt,
- schwach terminierend unter $\text{enforce}(p)$, $p \in P$, wenn in jedem Ablauf von G eine Situation $s = [q, m]$ durchlaufen wird, in der $m > [p]$ ist,
- schwach terminierend unter $\text{enforce}(P')$, $P' \subseteq P$, wenn für jeden Ablauf von G ein $p \in P'$ existiert, so dass eine Situation $s = [q, m]$ durchlaufen wird, in der $m > [p_B]$ ist,
- schwach terminierend unter $\text{enforce}(t)$, $t \in T$, wenn in jedem Ablauf von G eine Aktion $[q, t, q']$ vorkommt,
- schwach terminierend unter $\text{enforce}(T')$, $T' \subseteq T$, wenn in jedem Ablauf von G ein $t_B \in T'$ existiert, so dass in dem Ablauf eine Aktion $[q, t_B, q']$ vorkommt.

Die Eigenschaften bedienbar, Strategie und allgemeinste Strategie werden für *enforce* analog zu Definition 16 definiert:

Definition 19 (Strategie, bedienbar, allgemeinste Strategie) *Gegeben sei ein oWFN \mathcal{P} und ein Service-Automat \mathcal{R} . Dann gilt:*

- \mathcal{R} heißt Strategie für \mathcal{P} unter $\text{enforce}(X)$, wenn $\mathcal{R} \oplus \mathcal{P}$ unter $\text{enforce}(X)$ schwach terminiert,
- \mathcal{P} heißt bedienbar unter $\text{enforce}(X)$ gdw. es eine Strategie \mathcal{R} für \mathcal{P} unter $\text{enforce}(X)$ gibt,

- \mathcal{R} heißt allgemeinste Strategie unter $\text{enforce}(X)$, wenn \mathcal{R} Strategie unter $\text{enforce}(X)$ ist und für alle Strategien $\mathcal{R}' = [Q_{\mathcal{R}'}, T_{\mathcal{R}'}, q_{0,\mathcal{R}'}]$ für \mathcal{P} unter $\text{enforce}(X)$ gilt: $Q_{\mathcal{R}'} \subseteq Q$.

Leider ist es nicht möglich, analog zu Lemma 1 alle Bedingungen auf Markierungen zurückzuführen, da das Erreichen einer Markierung $m > [\bullet t]$ das Schalten von t zwar ermöglicht, aber nicht garantiert. Aus diesem Grund ist es erforderlich, den bisherigen Ablauf in die Entscheidung, ob ein Zustand Teil einer Strategie unter $\text{enforce}(X)$ ist, mit einzubeziehen.

Dazu wird zunächst die Eigenschaft der Stabilität von Markierungen (Definition 21) kanonisch auf andere Strukturen übertragen.

Definition 20 (stabiler Teilablauf, stabiler Knoten) Gegeben sei ein Interaktionssystem $G = [S, X, s_{ini}, S_{\Omega}]$ und sein Ablaufbaum $AB = [V_{AB}, E_{AB}, v_{ini}, \text{mark}]$. Sei weiterhin a_1, a_2, \dots, a_n ein Teilablauf von G , mit $a_i = [s_{i-1}, l_i, s_i] \in X$, $s_i = [q_i, m_i]$ und $s_0 = s_{ini}$. Dann gilt:

- a_1, a_2, \dots, a_n heißt stabiler Teilablauf von G , wenn m_n eine stabile Markierung ist,
- $v \in V$ heißt stabiler Knoten, wenn $\text{mark}(v)$ eine stabile Markierung ist.

Definition 21 (transienter Übergang) Gegeben sei ein Interaktionssystem $G = [S, X, s_{ini}, S_{\Omega}]$ und sein Ablaufbaum $AB = [V_{AB}, E_{AB}, v_{ini}, \text{mark}]$. Sei darüber hinaus $e = [v, l, v'] \in E_{AB}$ und v kein stabiler Knoten. Dann heißt e transienter Übergang. Die Menge aller Kanten transienter Übergänge aus E_{AB} wird $E_{AB}|_q$ genannt.

Anhand dieser Definitionen lassen sich nun für einen gegebenen Knoten v eines Ablaufbaumes all jene Nachfolgerknoten von v bestimmen, deren Erreichen den Zustand des Service-Automaten nicht verändert.

Definition 22 ($\text{trans}(v)$) Gegeben sei ein Service-Automat \mathcal{R} und ein oWFN \mathcal{P} . Sei $G = \mathcal{R} \oplus \mathcal{P}$ das resultierende Interaktionssystem und $AB = [V_{AB}, E_{AB}, v_{ini}, \text{mark}]$ der Ablaufbaum von G .

Zuerst wird eine Abbildung $\text{step} : V \rightarrow \wp(V)$ definiert:

$$\text{step}(v) = \{v' \mid \exists e = [v, l, v'] \in E_{AB} : e \text{ ist transienter Übergang}\}$$

Ausgehend von $\text{step}(v)$ wird nun die Abbildung $\text{trans}(v) : V \rightarrow \wp(V)$ wie folgt induktiv definiert:

Induktionsanfang: $v \in \text{trans}(v)$

Induktionsschritt: $\bigcup_{v' \in \text{trans}(v)} \text{step}(v') \subseteq \text{trans}(v)$

Die Abbildung $trans$ ordnet jedem Knoten v alle Knoten zu, die von v aus durch beliebig viele transiente Übergänge zu erreichen sind. Somit ist $trans(v)$ genau die oben beschriebene Menge von Nachfolgerknoten von v .

Auf Basis von $trans$ kann nun eine erweiterte Knowledge-Funktion angegeben werden, die jedem Zustand q eine Menge von Knoten des Ablaufbaumes zuordnet und damit jeden Teilablauf identifiziert, an dessen Ende sich der Requester im Zustand q befindet.

Definition 23 ($k'_{\mathcal{R},\mathcal{P}}$) Gegeben seien ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$. Sei ferner $G = \mathcal{R} \oplus \mathcal{P} = [S, X, s_{ini}, S_\Omega]$ das resultierende Interaktionssystem und $AB = [V_{AB}, E_{AB}, v_{ini}, mark]$ der Ablaufbaum von G . Die Abbildung $k'_{\mathcal{R},\mathcal{P}} : Q \rightarrow V_{AB}$ ist dann wie folgt induktiv definiert:

Induktionsanfang:

$$k'_{\mathcal{R},\mathcal{P}}(q_0) = trans(v_{ini})$$

Induktionsschritt:

Sei $q \in Q$ mit $k'_{\mathcal{R},\mathcal{P}}(q) = V_q$ und $q' \in Q$, $q' \neq q$ und es existiere ein l , so dass $[q, l, q'] \in E$.

Sei weiterhin $V' = \{v' \mid [v, l, v'] \in E \text{ mit } v \in V_q \text{ und } v' \sim s', s' = [q', m_p]\}$. Dann gilt:

$$k'_{\mathcal{R},\mathcal{P}}(q') = \bigcup_{v' \in V'} trans(v')$$

Es ist anhand Definition 22 leicht zu sehen, dass $k'_{\mathcal{R},\mathcal{P}}(q_0)$ tatsächlich *alle* Knoten des Ablaufbaumes enthält, in denen sich \mathcal{R} in q_0 befindet. Betrachtet man nun einen Zustandsübergang $z = [q, l, q'] \in E$, so ist $k'_{\mathcal{R},\mathcal{P}}(q) = V_q$ die Menge der Knoten, in denen dieser Zustandsübergang z eintreten kann. V' ist nach der Definition die Menge aller Knoten v' des Ablaufbaumes, die über einen Übergang $e = [v, l, v']$ von einem Knoten $v \in V_q$ aus erreichbar sind. Da \mathcal{R} deterministisch ist, befindet sich \mathcal{R} in jedem dieser Knoten im Zustand q' .

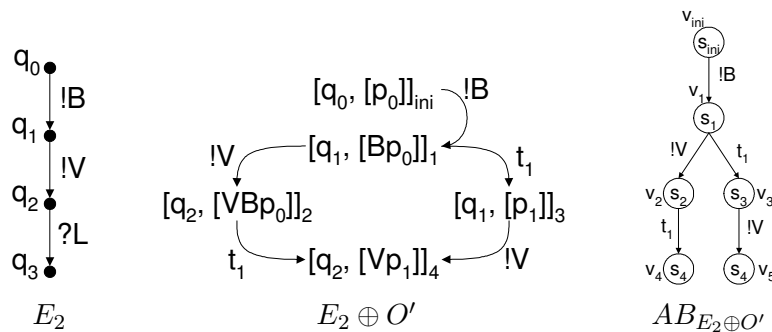


Abbildung 8: Service-Automat E_2 in Interaktion mit O' .

Somit ordnet diese Erweiterung der Knowledge-Funktion jedem Zustand des Service-Automaten *alle* diejenigen Knoten des Ablaufbaumes zu, deren Geschichte als Teilablauf

R in den Zustand q' überführen, sowie deren durch transiente Übergänge erreichbare Nachfolger. Betrachtet man das Beispiel aus Abbildung 8, so ordnet $k'_{E_2, O'}$ den einzelnen Zuständen von E_2 folgende Werte zu:

$$\begin{aligned} k'_{E_2, O'}(q_0) &= \{v_{ini}\} \\ k'_{E_2, O'}(q_1) &= \{v_1, v_3\} \\ k'_{E_2, O'}(q_2) &= \{v_2, v_4, v_5\} \\ k'_{E_2, O'}(q_3) &= \emptyset \end{aligned}$$

Die leere Menge tritt – genau wie bei der Knowledge-Funktion – auf, wenn ein Zustand keine Entsprechung im Interaktionssystem hat.

Auf der Basis von $k'_{\mathcal{R}, \mathcal{P}}$ lassen sich nun weitere Funktionen definieren, die nur solche Knoten enthalten, in deren Geschichte ein bestimmtes Ereignis noch nicht eingetreten ist. Dazu muss vorher die Abbildung *trans* erweitert werden:

Definition 24 ($trans^X(v)$) *Gegeben sei ein Service-Automat \mathcal{R} und ein oWFN \mathcal{P} . Sei $G = \mathcal{R} \oplus \mathcal{P}$ das resultierende Interaktionssystem und $AB = [V_{AB}, E_{AB}, v_{ini}, mark]$ der Ablaufbaum von G . Sei weiterhin $m \in \mathbb{N}^P$, $M \subseteq \mathbb{N}^P$, $p \in P$, $P' \subseteq P$, $t \in T$, $T' \subseteq T$ gegeben.*

Analog zu Definition 22 werden zuerst Abbildungen $step^X : V \rightarrow \wp(V)$ definiert:

- $step^m(v) = \{v' \mid \exists e = [v, l, v'] \in E_{AB}|_q \wedge mark(v') \not\geq m\}$,
- $step^M(v) = \{v' \mid \exists e = [v, l, v'] \in E_{AB}|_q \wedge \nexists m \in M : mark(v) \geq m\}$,
- $step^p(v) = \{v' \mid \exists e = [v, l, v'] \in E_{AB}|_q \wedge mark(v) \not\geq [\{p\}]\}$,
- $step^{P'}(v) = \{v' \mid \exists e = [v, l, v'] \in E_{AB}|_q \wedge \nexists p \in P' : mark(v) \geq [\{p\}]\}$,
- $step^t(v) = \{v' \mid \exists e = [v, l, v'] \in E_{AB}|_q \wedge l \neq t\}$,
- $step^{T'}(v) = \{v' \mid \exists e = [v, l, v'] \in E_{AB}|_q \wedge \nexists t \in T' : l = t\}$.

Ausgehend von $step^X(v)$ wird nun die Abbildung $trans^X(v) : V \rightarrow \wp(V)$ wie folgt induktiv definiert:

Induktionsanfang:

- $v \in trans^m(v)$, falls $mark(v) \not\geq m$,
- $v \in trans^M(v)$, falls $\nexists m \in M : mark(v) \geq m$,
- $v \in trans^p(v)$, falls $mark(v)(p) = 0$,
- $v \in trans^{P'}(v)$, falls $\nexists p \in P' : mark(v) \geq [\{p\}]$,
- $v \in trans^t(v)$,
- $v \in trans^{T'}(v)$.

Induktionsschritt:

$$\bigcup_{v' \in trans^X(v)} step^X(v') \subseteq trans^X(v)$$

Die Menge $trans^X(v)$ enthält demnach nur solche Knoten, deren Erreichen von v aus nicht dazu geeignet ist, Definition 18 zu genügen und die den Zustand von \mathcal{R} unverändert lassen. So ist in Abbildung 8 $trans^{t_1}(v_2) = \emptyset$, da v_4 zwar sehr wohl in $step(v_2)$ enthalten ist, nicht aber in $step^{t_1}(v_2)$. Auf dieser Basis kann nun auch $k'_{\mathcal{R},\mathcal{P}}$ derart erweitert werden, dass das Wissen um ein bestimmtes Ereignis einfließt:

Definition 25 ($k_{\mathcal{R},\mathcal{P}}^X$) Gegeben seien ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$. Sei ferner $G = \mathcal{R} \oplus \mathcal{P} = [S, X, s_{ini}, S_\Omega]$ das resultierende Interaktionssystem und $AB = [V_{AB}, E_{AB}, v_{ini}, mark]$ der Ablaufbaum von G .

Für $m \in \mathbb{N}^P$, $M \subseteq \mathbb{N}^P$, $p \in P$, $P' \subseteq P$, $t \in T$, $T' \subseteq T$ sind die Knowledge-Funktionen $k_{\mathcal{R},\mathcal{P}}^m$, $k_{\mathcal{R},\mathcal{P}}^M$, $k_{\mathcal{R},\mathcal{P}}^p$, $k_{\mathcal{R},\mathcal{P}}^{P'}$, $k_{\mathcal{R},\mathcal{P}}^t$, $k_{\mathcal{R},\mathcal{P}}^{T'}$ wie folgt induktiv definiert ($k_{\mathcal{R},\mathcal{P}}^X$ steht dabei stellvertretend für die jeweilig passende Knowledge-Funktion):

Induktionsanfang:

$$k_{\mathcal{R},\mathcal{P}}^X(q_0) = trans^X(v_{ini})$$

Induktionsschritt:

Sei $q \in Q$ mit $k_{\mathcal{R},\mathcal{P}}^X(q) = V_q$ und $q' \in Q$, $q' \neq q$ und es existiere ein l , so dass $[q, l, q'] \in E$.

Sei weiterhin $V' = \{v' \mid [v, l, v'] \in E_{AB} \text{ mit } v \in V_q \text{ und } v' \sim s', s' = [q', m]\}$. Dann gilt:

$$k_{\mathcal{R},\mathcal{P}}^X(q') = \bigcup_{v' \in V'} trans^X(v')$$

Lemma 3 Die Menge $\{h(v) \mid v \in k_{\mathcal{R},\mathcal{P}}^X(q)\}$ der Geschichten der Knoten aus $k_{\mathcal{R},\mathcal{P}}^X$ enthält alle Teilabläufe von G , die \mathcal{R} in q überführen und noch nicht Definition 18 bezüglich $enforce(X)$ genügen.

Beweisskizze: Die Geschichten der Knoten aus $k'_{\mathcal{R},\mathcal{P}}(q)$ geben alle Teilabläufe wieder, die \mathcal{R} in q überführen. Die Definition von $k_{\mathcal{R},\mathcal{P}}^X$ nimmt nur Knoten aus $trans^X(v_{ini})$ auf, in denen G per Definition von $trans^X$ nicht schon die Bedingungen für $exclude(X)$ erfüllt. In jedem Schritt werden gegenüber $k'_{\mathcal{R},\mathcal{P}}$ durch die Verwendung von $trans^X$ nur diejenigen Knoten weggelassen, mit deren Erreichen ebendiese Bedingungen erfüllt werden.

Im eingangs angegebenen Beispiel war das Ziel $enforce(V)$. Für Abbildung 8 ist die zu betrachtende Funktion demnach $k_{E_2, O'}^V$:

$$\begin{aligned} k_{E_2, O'}^V(q_0) &= \{v_{ini}\} \\ k_{E_2, O'}^V(q_1) &= \{v_1, v_3\} \\ k_{E_2, O'}^V(q_2) &= \emptyset \\ k_{E_2, O'}^V(q_3) &= \emptyset \end{aligned}$$

Im Gegensatz zu $k_{\mathcal{R},\mathcal{P}}$ und $k'_{\mathcal{R},\mathcal{P}}$ lässt sich bei $k_{\mathcal{R},\mathcal{P}}^X$ aus der leeren Menge nun nicht

mehr schließen, dass der betreffende Zustand keine Entsprechung im Interaktionssystem hat (wie das bei q_3 der Fall ist). Es ist auch möglich, dass alle stabilen Teilabläufe, die den Service-Automaten in den fraglichen Zustand überführen, schon der Forderung $\text{enforce}(X)$ genügen (wie bei q_2 zu sehen).

Auf Basis von Lemma 3 kann man nun analog zu Lemma 2 Strategien unter $\text{enforce}(X)$ charakterisieren:

Lemma 4 *Gegeben sei ein Service-Automat $\mathcal{R} = [I_{in}, I_{out}, Q, E, q_0, Q_\Omega]$ und ein oWFN $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$. Sei $G = \mathcal{R} \oplus \mathcal{P}$ das Interaktionssystem und $AB = [V_{AB}, E_{AB}, v_{ini}, mark]$ sein Ablaufbaum. Dann gilt: \mathcal{R} ist Strategie für \mathcal{P} unter $\text{enforce}(X)$, gdw. folgende Bedingungen erfüllt sind:*

- (a) \mathcal{R} ist Strategie für \mathcal{P} ,
- (b) für alle $q \in Q$ gilt: wenn $k_{\mathcal{R}, \mathcal{P}}^X(q)$ einen stabilen Knoten v enthält, so existiert ein Zustandsübergang $[q, l, q'] \in E$, so dass im Ablaufbaum eine Kante $[v, l, v'] \in E_{AB}$ existiert.

Die Beschränkung auf stabile Knoten ist deswegen notwendig, weil bei transienten Knoten im resultierende Interaktionssystem $G = \mathcal{R} \oplus \mathcal{P}$ noch weitere Situationen durchlaufen werden können und somit die Erfüllung der Bedingungen aus Definition 18 nicht abschätzbar ist.

Betrachtet man einen Zustand $q \in Q$, für den in $k_{\mathcal{R}, \mathcal{P}}^X(q)$ mindestens einen stabiler Knoten enthalten ist, so existiert mindestens ein Teilablauf in G , der \mathcal{R} in q überführt und nicht Definition 18 genügt. Das zu erzwingende Ereignis ist also noch nicht in jedem Falle eingetreten. Wird ein solcher stabiler Knoten nicht aufgelöst, d.h. es existiert keine Kante, die v in ein v' überführt, so wird in dem Teilablauf, dessen Ende v markiert, das Ereignis X niemals eintreten.

Enthält $k_{\mathcal{R}, \mathcal{P}}^X(q)$ hingegen keine stabilen Knoten und die Bedingung (a) ist erfüllt, so genügen nach Lemma 3 alle stabilen Teilabläufe von G , die \mathcal{R} in den Zustand q überführen, der Definition 18.

Mittels dieses Lemmas kann man nun einen Algorithmus angeben, der analog zu Algorithmus 1 bei gegebenem oWFN \mathcal{P} aus \mathcal{A} die allgemeinste Strategie unter $\text{enforce}(X)$ konstruiert.

Algorithmus 2 (enforce) *Bei gegebenem $\mathcal{P} = [P, T, F, in, out, m_0, \Omega]$ kann nun ausgehend von \mathcal{A} folgender Algorithmus zur Konstruktion der allgemeinsten Strategie unter $\text{enforce}(X)$ angegeben werden. X steht dabei wieder stellvertretend für $m \in \mathbb{N}^P$, $M \subseteq \mathbb{N}^P$, $p \in P$, $P' \subseteq P$, $t \in T$, $T' \subseteq T$. Sei $G = \mathcal{A} \oplus \mathcal{P}$ das anfängliche Interaktionssystem, AB_G sein Ablaufbaum.*

1. Streichen aller Zustände $q_A \in Q_A$, bei denen $k_{\mathcal{A}, \mathcal{P}}(q_A) = \emptyset$
2. Streichen aller Zustände $q_A \in Q_A$, bei denen $k_{\mathcal{A}, \mathcal{P}}(q_A)$ einen internen Deadlock enthält

3. (a) *Sukzessives Streichen aller Zustände $q_A \in Q_A$, bei denen $k_{A,\mathcal{P}}(q_A)$ einen externen Deadlock m enthält und die keinen ausgehenden Zustandsübergang haben, der $[q_A, m]$ in eine neue Situation überführt.*
- (b) *Sukzessives Streichen aller Zustände $q_A \in Q_A$, bei denen $k_{A,\mathcal{P}}^X(q_A)$ einen stabilen Knoten v enthält und die keinen ausgehenden Zustandsübergang $[q_A, l, q']$ haben, so dass $[v, l, v'] \in E_{AB}$*

Das Streichen eines Zustandes zieht immer auch das Streichen des eingehenden Zustandsübergangs sowie aller ausgehenden Zustandsübergänge und Folgezustände mit sich. Jede Streichung erfolgt parallel in \mathcal{A} sowie in G und AB_G .

Anhand des Beispiels aus Abbildung 8 sieht man aber, dass $k_{A,\mathcal{P}}^X$ nicht dazu geeignet ist, Forderung a aus Lemma 4 zu entscheiden, weswegen in den Schritten 2 und 3a zusätzlich auf $k_{A,\mathcal{P}}$ zurückgegriffen wird.

Die Streichungen in Schritt 3a und 3b müssen so lange parallel ausgeführt werden, bis in keinem der beiden Schritte mehr Streichungen vorgenommen werden können. Das ist nötig, weil durch das Streichen von Zuständen die Bedingungen für deren Vorgänger eventuell nicht mehr erfüllt sind.

Satz 4 *Über Algorithmus 2 lassen sich folgende Aussagen treffen:*

1. *Der resultierende Service-Automat \mathcal{S} ist eine Strategie für \mathcal{P} unter $\text{enforce}(X)$.*
2. *\mathcal{S} ist sogar allgemeinste Strategie unter $\text{enforce}(X)$.*
3. *Sollte die Zustandsmenge von \mathcal{S} leer sein, so ist \mathcal{P} nicht bedienbar unter $\text{enforce}(X)$.*

Der Beweis läuft völlig analog zu Satz 3 ab, wobei zur Charakterisierung einer Strategie unter $\text{enforce}(X)$ Lemma 4 herangezogen wird.

4 Zusammenfassung

Es wurde das Konzept von zielgerichteten Strategien vorgestellt. Insbesondere wurden die Ziele *exclude* und *enforce* eingeführt sowie für diese Ziele ein Algorithmus zur Konstruktion der allgemeinsten dieses Ziel verfolgenden Strategie angegeben. Mittels des Zieles *enforce* ist es möglich, zu überprüfen, ob eine bestimmte Verhaltensweise X eines Service-Providers P überhaupt durch einen Service-Requester R hervorgerufen werden kann. Ist dies nicht der Fall, so ist P nicht bedienbar unter $\text{enforce}(X)$. Analog ist es mittels der Frage nach der Bedienbarkeit unter $\text{exclude}(X)$ möglich, die Vermeidbarkeit von unerwünschtem Verhalten zu überprüfen.

Die allgemeinsten Strategien können zur Erstellung zielgerichteter Bedienungsanleitungen herangezogen werden. Diese können zum Beispiel genutzt werden, um mittels *exclude* bedienungsbedingte Fehler auszuschließen und somit die Effizienz der durch einen Service-Broker durchgeführten Zuordnung Provider - Requester zu erhöhen oder

mittels enforce aufgabenspezifische Bedienungsanleitungen für Services mit großem Leistungsangebot zur Verfügung zu stellen.

Aufbauend auf dieser Arbeit kann eine Möglichkeit geschaffen werden, die vorgestellten Ziele zu verknüpfen, um weitere in der Analyse eines Service interessante Fragen beantworten zu können. Darüber hinaus erlaubt dies die Vorzüge von Bedienungsanleitungen unter enforce und exclude zu kombinieren.

Literatur

- [1] Gottschalk, K.: Web services architecture overview. IBM whitepaper, IBM developerWorks (2000)
- [2] Schmidt, K.: Controllability of Open Workflow Nets. In Desel, J., Frank, U., eds.: Enterprise Modelling and Information Systems Architectures. Volume P-75 of Lecture Notes in Informatics (LNI)., Bonn, Entwicklungsmethoden für Informationssysteme und deren Anwendung (EMISA, RWTH Aachen), Köllen Druck+Verlag GmbH (2005) 236–249
- [3] Massuthe, P., Schmidt, K.: Operating Guidelines - an Automata-Theoretic Foundation for the Service-Oriented Architecture. In Cai, K.Y., Ohnishi, A., Lau, M., eds.: Proceedings of the Fifth International Conference on Quality Software (QSIC 2005), Melbourne, Australia, IEEE Computer Society (2005) 452–457
- [4] Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. *Annals of Mathematics, Computing & Teleinformatics* **1**(3) (2005) 35–43
- [5] Massuthe, P., Schmidt, K.: Matching Nondeterministic Services with Operating Guidelines. *Informatik-Berichte* 193, Humboldt-Universität zu Berlin (2005)
- [6] Massuthe, P., Schmidt, K.: Operating Guidelines - an Alternative to Public View. *Informatik-Berichte* 189, Humboldt-Universität zu Berlin (2005)