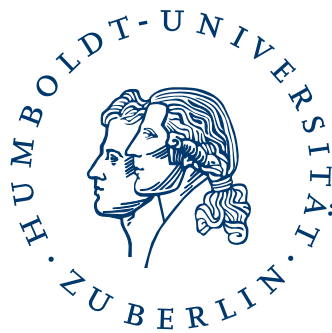


Diplomarbeit

Strukturelle Reduktion von Bedienungsanleitungen

Christian Gierds

25. Januar 2008



Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik
Gutachter:
Prof. Dr. Wolfgang Reisig
Prof. Dr. Karsten Wolf

Zusammenfassung

In dieser Arbeit werden wir uns mit *Diamantenstrukturen* in *Bedienungsanleitungen* beschäftigen. Im Rahmen der *Service-Oriented Architecture* beschreiben Bedienungsanleitungen Kommunikationspartner (*Strategien*) von Diensten. Ein großer Vorteil der Bedienungsanleitungen ist die endliche Repräsentation der gewöhnlich unendlich großen Menge aller Strategien eines Dienstes. Bedienungsanleitungen können nichtsdestotrotz sehr groß werden. Diamantenstrukturen sind eine der Hauptursachen dafür.

Wir wollen eine Kurzschreibweise einführen, die das Eintreten von Ereignissen in jeder beliebigen Reihenfolge in einem Zustandsübergang beschreibt. Anschließend werden wir verschiedene Muster für Diamantenstrukturen vorstellen, also Muster mit Ereignissen, die in jeder beliebigen Reihenfolge stattfinden können. Für diese Muster werden wir Ersetzungen mit der eingeführten Kurzschreibweise angeben.

Wir werden ferner Algorithmen angeben, welche die von uns definierten Diamantenmuster in Bedienungsanleitungen erkennen.

Inhaltsverzeichnis

1	Einleitung	6
2	Begriffsbildung	9
2.1	Ein Modell zur Beschreibung von Services	9
2.2	Serviceautomaten	10
2.3	Kommunikation von Serviceautomaten	13
3	Erweiterung der Begriffswelt	24
3.1	Serviceautomaten mit nebenläufigen Ereignissen	24
3.2	Diamantenstrukturen	29
4	Reduktion der Bedienungsanleitung	33
4.1	Elementare Pattern	33
4.2	Erweiterte Pattern	45
4.3	Geeignete Wahl des ersten Knotens eines Diamanten	61
4.4	Beispiele	62
5	Fazit	69
5.1	Zusammenfassung	69
5.2	Ausblick	70
6	Danksagung	73
	Literaturverzeichnis	75
	Erklärung	77

1 Einleitung

In einer zunehmend stärker vernetzten Welt spielen *Webservices* eine immer größere Rolle. Ein Webservice ist dabei eine Komponente, die eine Schnittstelle, ein Interface, zu ihrer Umgebung besitzt, über die sie kommuniziert. Der Webservice wird über eine URL¹ aufgerufen. Wie der Name sagt, stellt ein Webservice einen Dienst zur Verfügung. Das können simple Dienste wie die Erstellung eines PDF-Dokumentes sein, aber auch komplexere wie Online-Shopping oder -Banking.

Als Konzept, um die Interaktion von Webservices zu steuern und zu beschreiben, hat sich das Paradigma der dienstorientierten Architektur (*Service-Oriented Architecture, SOA*) etabliert. Eine häufige Interpretation des SOA-Prinzips ist in Abbildung 1.1 zu sehen.

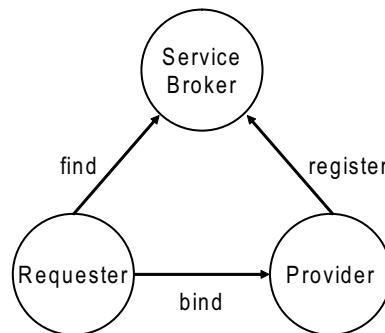


Abbildung 1.1: Eine mögliches Modell des SOA-Prinzips: Ein Anbieter (*provider*) registriert seine Dienste bei einem Vermittler (*service broker*), damit andere sie nutzen können. Ein Dienstanwender (*requester*) kann dann über den Vermittler einen gewünschten Dienst finden und mit ihm interagieren.

Dieses Konzept ist bewusst wagen gehalten und eine praktische Umsetzung hat sich bisher nicht durchgesetzt.

Eines der Hauptprobleme ist zu entscheiden, wann zwei Dienste grundsätzlich und dann auch sinnvoll miteinander kommunizieren können. Können wir diese Frage klären, möchten wir dies auch möglichst schnell entscheiden können.

¹URL = Uniform Resource Locator, in etwa „eindeutige Ressourcenauffindung“

Ein Ansatz, der in den letzten Jahren eine formale Grundlage für die Lösung dieser Probleme geschaffen hat, ist das Konzept der *Bedienungsanleitungen*. Gegeben einen Service, charakterisiert seine Bedienungsanleitung all seine mit ihm sinnvoll interagierenden Partner, wobei sinnvolle Kommunikation in diesem Fall immer dann vorliegt, wenn zwei Dienste nicht in einen gemeinsamen Verklemmungszustand geraten können. Einen großen Vorteil bietet die endliche Struktur der Bedienungsanleitung, mit der wir eine unendlich große Menge an Partnern beschreiben können. Weiterhin gibt es einen effizienten Matchingalgorithmus, der für einen Service A anhand dessen Bedienungsanleitung OG_A entscheidet, ob ein Partner B sinnvoll mit A kommunizieren kann.

Eine wichtige Voraussetzung spielt in diesem Konzept die asynchrone Kommunikation von Diensten. Eine Sequenz von Nachrichten kann unvorhersagbar in einer beliebigen Reihenfolge beim Empfänger ankommen. Dies muss bei der Charakterisierung aller sinnvollen Partner berücksichtigt werden und führt dazu, dass die Bedienungsanleitung sehr groß im Vergleich zum gegebenen Dienst werden kann.

Da die Bedienungsanleitung mitunter vielfach übertragen und gespeichert werden muss, ist es unser Bestreben, diese möglichst kompakt darzustellen. In dieser Arbeit wollen wir uns mit bestimmten Diamanten- oder Rautenstrukturen beschäftigen, die durch das nebenläufige Eintreten von Ereignissen entstehen. Die Anzahl der Zustände der Struktur ist exponentiell groß im Vergleich zu der Zahl der eintretenden Ereignisse. Es bietet sich hier ein enormes Potential, Speicherplatz zu sparen. Auch wenn im SOA-Konzept menschliches Mitwirken nicht eingeschlossen ist, bietet eine kompakte Struktur durchaus weitere Vorteile, zum Beispiel für die Lesbarkeit einer Bedienungsanleitung, was für Modellierer oder Entwickler zur Kontrolle ihrer Dienste hilfreich ist.

Wenn wir in dieser Arbeit, wie eben geschehen, den Begriff der *Nebenläufigkeit von Ereignissen* benutzen, wollen wir darunter das Eintreten der Ereignisse in beliebiger Reihenfolge verstehen. Ereignisse treten dabei immer nacheinander ein, niemals gleichzeitig.

Ein weit verbreiteter und gut erforschter Ansatz, um die Zustandsraumexplosion durch unabhängige Aktionen (bei uns Ereignisse) zu vermeiden, ist *Partial Order Reduction* (siehe beispielsweise [CGP01]). Dabei wird unabhängigen Aktionen eine Halbordnung aufgezwungen, und somit werden nur bestimmte Reihenfolgen der Aktionen (Abläufe) berücksichtigt. Dies stellt jedoch ein Problem dar, wenn wir versuchen, Partial Order Reduction auf Bedienungsanleitungen anzuwenden. Das Ergeb-

nis muss jeden Ablauf charakterisieren, auch diejenigen, welche durch die Reduktion herausgefallen sind.

Wir werden eine musterbasierte Reduktion für Bedienungsanleitungen einführen. Dabei wollen wir zu definierende Muster in der Bedienungsanleitung erkennen und diese anschließend durch eine kompakte Repräsentation ersetzen, die diese Muster eindeutig beschreibt.

Zuerst werden wir in Kapitel 2 das Umfeld definieren, in dem wir Bedienungsanleitungen verwenden. Zum einen werden wir offene Workflownetze einführen, die wir in Beispielen nutzen wollen. Ihre Umsetzung in Serviceautomaten bereitet dann die Grundlage, Strategien, das heißt sinnvoll agierende Partner von Serviceautomaten, zu definieren. Die Charakterisierung von Strategien bildet das Fundament, mit dem wir das Konzept der Bedienungsanleitung eines Serviceautomaten entwickeln werden. Diese Begriffe beruhen auf Vorarbeiten, zum Beispiel [MS05] oder [MRS05], und liefern die Grundlage für die weiteren Betrachtungen.

Anschließend werden wir in Kapitel 3 als Erweiterung der bisherigen Begriffswelt Serviceautomaten mit nebenläufigen Ereignissen definieren. Diese bauen auf den zuvor definierten Serviceautomaten auf, beinhalten als Neuerung aber die Möglichkeit, mit einem einzigen Zustandsübergang auszudrücken, dass mehrere Ereignisse nebenläufig, also in jeder beliebiger Reihenfolge, stattfinden können. Weiterhin werden wir festlegen, welche Art von (Teil-)Strukturen einer Bedienungsanleitung wir als Diamant verstehen wollen.

In Kapitel 4 werden wir zuerst einfache Muster kennen lernen, in denen Ereignisse in jeder beliebiger Reihenfolge auftreten können. Diese Muster sind in gewisser Weise losgelöst vom Rest einer Bedienungsanleitung, sodass sie leicht durch eine Kurzschreibweise ersetzt werden können. Ein Pseudocodealgorithmus beschreibt die Erkennung der Muster. Wir werden zeigen, dass er die Muster korrekt erkennt, und dass die Ersetzung aufgrund seines Ergebnisses die durch die Bedienungsanleitung charakterisierten Strategien bewahrt.

Aufbauend auf den einfachen Mustern werden wir komplexere Muster einführen. Bei diesen müssen wir zum Beispiel berücksichtigen, dass Kanten in die Diamantenstrukturen hinein oder aus ihr heraus führen. Wir werden den ursprünglichen Algorithmus jeweils an die neue Situation anpassen und entsprechende Ersetzungen der Muster mit einer Kurzschreibweise angeben.

Die Ergebnisse dieser Arbeit werden wir in Kapitel 5 zusammenfassen. Abschließend gibt es einen Ausblick auf Themen, die unseren Betrachtungen folgen könnten.

2 Begriffsbildung

In diesem Kapitel werden wir alle Begriffe kennen lernen, die wir im weiteren Verlauf der Arbeit benötigen werden. Die zentrale Theorie basiert auf kommunizierenden Serviceautomaten. Für anschauliche Beispiele werden wir jedoch vorher offene Workflownetze einführen, mit denen sich auf andere Art Kommunikationsverhalten darstellen lässt. Zudem existiert eine Überführung offener Workflownetze in Serviceautomaten. Für letztere werden wir einen Kompositionsoperator definieren und ein Kriterium in Form eines Lemmas angeben, wann wir die Komposition als sinnvoll betrachten wollen. Letztendlich werden wir in diesem Kapitel sehen, wie sich die Menge aller sinnvoll mit einem Serviceautomaten interagierenden Partner in Form der Bedienungsanleitung beschreiben lässt, und wie diese konstruiert wird. Am Ende des Kapitels wollen wir eine Kurzschreibweise für Zustandsübergänge einführen. Diese soll einen Zustandsübergang mit einer Menge nebenläufiger Ereignisse beschreiben.

2.1 Ein Modell zur Beschreibung von Services

Bevor wir das Konzept der Serviceautomaten erläutern, auf dem die in dieser Arbeit genutzte Theorie basieren wird, werden wir als Beschreibung von Services *offene Workflownetze* (*oWFN*), eine Abwandlung der Petrinetze, einführen. Mit Hilfe der offenen Workflownetze lassen sich einfach nebenläufige Ereignisse modellieren, die das zentrale Problem dieser Arbeit, die Zustandsraumexplosion in Bedienungsanleitungen, verursachen. Neben anschaulichen Beispielen, die wir in dieser Arbeit nutzen werden, sind offene Workflownetze Grundlage für die Analyse von Geschäftsprozessen ([LMSW06]).

Im Gegensatz zu klassischen Petrinetzen (siehe zum Beispiel [Rei82]), deren Kenntnis wir hier voraussetzen möchten, zeichnen oWFNs zwei spezielle Platzmengen aus – die der Eingabe- und der Ausgabekanäle. Marken, die auf diesen Plätzen liegen, sind abstrakt betrachtet Nachrichten, die mit der Umgebung ausgetauscht werden.

Definition 2.1 (Offenes Workflownetz)

Ein offenes Workflownetz (oWFN) ist ein Petrinetz zusammen mit Platzmengen P_I und P_O sowie einer Menge von Endmarkierungen Ω , sodass

- die Mengen der Eingabeplätze $P_I \subseteq P$ (nur ausgehende Kanten) und der Ausgabeplätze $P_O \subseteq P$ (nur eingehende Kanten) disjunkt ($P_I \cap P_O = \emptyset$) sind,
- die Anfangsmarkierung m_0 keine Eingabe- oder Ausgabeplätze markiert ($m_0(p) = 0$ für $p \in P_I \cup P_O$) und
- in jeder Endmarkierung $m \in \Omega$ die Eingabe- und Ausgabeplätze unmarkiert sind ($m(p) = 0$ für $p \in P_I \cup P_O$), und keine Transition aktiviert ist.

┘

Zu fordern, dass die Mengen der Eingabe- und Ausgabeplätze eines oWFN N disjunkt sind, ist insofern notwendig, da wir sonst nicht unterscheiden könnten, ob eine Marke auf einem bidirektionalen Platz von der Umgebung des Netzes N oder von N selbst erzeugt wurde.

Eine Besonderheit in dieser Definition spielt die Menge der Endmarkierungen, die in der klassischen Petrinetztheorie unbedeutend ist. Dort sind zum Beispiel Verklemmungen, also Markierungen, die keine Transition aktivieren, relevant. Die Endmarkierungen eines oWFN separieren die Verklemmungen in solche, die erwünscht sind, und solche, die unerwünscht sind, die also keiner Endmarkierung entsprechen.

In der folgenden Abbildung 2.1 sehen wir ein Beispiel eines oWFN.

2.2 Serviceautomaten

Im vorhergehenden Abschnitt haben wir offene Workflownetze betrachtet. Die Analyse der Kommunikation eines Service mit seiner Umgebung lässt sich jedoch einfacher mit Serviceautomaten bewerkstelligen, für die in dieser Hinsicht eine ausgereifte Theorie vorhanden ist.

Wie der Name Serviceautomat impliziert, betrachten wir eine Automatenstruktur. In unserem Fall sind dabei die Zustandsübergänge mit Nachrichtenkanälen beschriftet, das heißt, der Übergang ist mit dem Senden oder Empfang einer Nachricht verbunden. Alternativ kann ein Übergang auch mit τ beschriftet sein. Das steht für einen internen Zustandsübergang des Serviceautomaten, bei dem keine Kommunikation stattfindet.

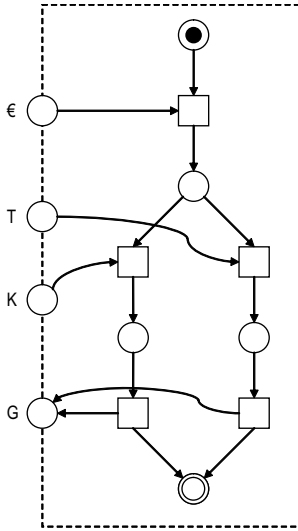


Abbildung 2.1: Links sehen wir das Modell eines Getränkeautomaten. Der Getränkeautomat ist kein praktisches Beispiel eines Webservice, aber einfach verständlich.

Wir sehen ein oWFN mit einer Anfangsmarkierung (oberster Platz mit Marke), den Kanälen € (Geld einwerfen), T (Taste für Tee drücken), K (Taste für Kaffee drücken) und G (Getränk erhalten), und einer Endmarkierung (unterster Platz mit Doppellinie). Die gestrichelte Linie soll das oWFN nach außen abgrenzen, die Interfaceplätze liegen auf dieser Linie.

Ein möglicher Ablauf für die Umgebung des oWFN besteht darin, jeweils eine Marke auf € und T zu legen, damit die entsprechenden Transitionen nacheinander schalten können. Das oWFN kann anschließend selbständig weiter schalten und legt eine Marke auf G, die der Partner aufnehmen kann.

Das oWFN hat dann keine Möglichkeit mehr zu schalten, ist aber nach dem Entfernen der Marke von G in einem Endzustand.

Definition 2.2 (Serviceautomat)

Ein Serviceautomat ist ein nichtdeterministischer Automat $A = (Q, I, O, \delta, q_0, F)$ mit

- einer endlichen Menge Q von Zuständen,
- einem Interface $I \cup O$, mit Eingabekanälen I (input) und Ausgabekanälen O (output) ($I \cap O = \emptyset$),
- einer Übergangsrelation $\delta \subseteq Q \times (I \cup O \cup \{\tau\}) \times Q$,
- einem Anfangszustand $q_0 \in Q$ sowie
- einer Menge von Endzustände $F \subseteq Q$, wobei für alle $q \in F$ gilt, dass $x \in I$ aus $(q, x, q') \in \delta$ folgt.

┘

Notation Die Menge aller Kanäle bezeichnen wir kurz mit $C = I \cup O$. Das Empfangen oder Senden der Nachricht c ist synonym mit dem Empfangen oder Senden einer Nachricht über den Kanal $c \in C$ zu verstehen.

Notation Reden wir über mehrere Serviceautomaten, werden die Bestandteile entsprechend mit dem Namen des Serviceautomaten indiziert, also Q_A für die Zustände des Serviceautomaten A usw.

Definition 2.3 (Vorgänger, Nachfolger)

Sei A ein Serviceautomat und (q_1, x, q_2) ein Zustandsübergang in A . Dann heißt q_1 der Vorgänger von q_2 und entsprechend q_2 der Nachfolger von q_1 . \lrcorner

Um die sinnvolle Kommunikation eines Serviceautomaten beschreiben zu können, müssen wir zuerst seine Zustände charakterisieren. Einen *Wartezustand* kann ein Serviceautomat nicht selbständig (durch Senden einer Nachricht oder einen internen Übergang) verlassen.

Definition 2.4 (Wartezustand)

Sei A ein Serviceautomat. Dann heißt ein Zustand $q \in Q$ genau dann Wartezustand, wenn $x \in I$ für alle $(q, x, q') \in \delta$ gilt. Die Abbildung *wait* sei für Wartezustände wie folgt definiert: $wait(q) = \{x \in I \mid \exists q' \in Q : (q, x, q') \in \delta\}$. \lrcorner

Die Abbildung *wait* gibt für Wartezustände an, durch Empfangen welcher Nachrichten der Zustand verlassen werden kann.

Mit Definition 2.2 sind Endzustände insbesondere Wartezustände.

Wartezustände, die keine Endzustände sind, und in denen auch das Empfangen einer Nachricht nicht mehr möglich ist, möchten wir als *Verklemmungszustände* bezeichnen. Der Serviceautomat stoppt in diesen Zuständen die Abarbeitung, obwohl kein definiertes Ende erreicht wurde.

Definition 2.5 (Verklemmung)

Sei A ein Serviceautomat und $q \in Q$ ein Wartezustand von A . Dann bezeichnen wir q genau dann als Verklemmungszustand, wenn $q \notin F$ und $wait(q) = \emptyset$. \lrcorner

Es ist möglich, offene Workflownetze in Serviceautomaten zu überführen (siehe beispielsweise [LMW07]) und somit alle Fragen, die uns für Serviceautomaten interessieren, auch im Kontext der offenen Workflownetze zu stellen. Im Folgenden werden wir gleichberechtigt Serviceautomaten und offene Workflownetze in Beispielen betrachten, wohl wissend, dass für beide die gleiche Theorie anwendbar ist. So entspricht der Serviceautomat aus Abbildung 2.2 dem oWFN aus Abbildung 2.1.

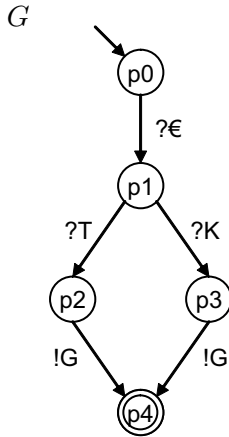


Abbildung 2.2: Der Serviceautomat G ist die Umsetzung des Getränkeautomaten aus Abbildung 2.1.

Im Anfangszustand p_0 (gekennzeichnet durch eingehenden Pfeil) kann der Serviceautomat Geld empfangen ($?€$) und wechselt in den Zustand p_1 . Im Zustand p_1 kann der Serviceautomat durch Empfangen der Nachricht für Tee ($?T$) in den Zustand p_2 bzw. der Nachricht für Kaffee ($?K$) in den Zustand p_3 übergehen. In den beiden Zuständen p_2 und p_3 geht der Serviceautomat durch Senden der Nachricht für die Ausgabe des Getränkes ($!G$) in den Endzustand p_4 (erkennbar an der doppelten Linie) über.

Die Zustände p_0 , p_1 und p_4 sind Wartezustände. Verklemmungszustände gibt es in diesem Serviceautomaten nicht.

In den Beispielen kennzeichnet $?x$ das Empfangen der Nachricht x , und $!x$ das Senden der Nachricht x .

2.3 Kommunikation von Serviceautomaten

Wie schon bei den offenen Workflownetzen liegt bei Serviceautomaten das Augenmerk auf der Kommunikation mit der Umgebung, genauer mit anderen Serviceautomaten.

Nahe liegend ist es, zwei Serviceautomaten komponieren zu können, um dann ihr gemeinsames Verhalten untersuchen zu können. Dafür müssen sie sich grundsätzlich verstehen können, was gewährleistet ist, wenn sie ein entsprechendes Interface besitzen. Damit zwei Serviceautomaten miteinander kommunizieren können, muss der eine genau die Nachrichten empfangen können, die der andere senden kann.

Definition 2.6 (Interfacekompatible Serviceautomaten)

Zwei Serviceautomaten A und B heißen genau dann kompatibel bezüglich ihres Interfaces, wenn $I_A = O_B$ und $O_A = I_B$ gilt. \lrcorner

Jetzt, wo die Voraussetzungen für die Komposition zweier Serviceautomaten gegeben sind, können wir angeben, wie diese funktionieren soll. Einen wichtigen Punkt unserer Theorie stellt die asynchrone Kommunikation dar. Eine Nachricht, die ein Serviceautomat sendet, muss nicht sofort vom anderen Serviceautomaten empfangen werden. Zum einen heißt das, dass der erste Serviceautomat nicht verklemmt, bis die Nachricht empfangen wurde, sondern er kann in seiner Abarbeitung fortfahren, und zum anderen müssen wir uns merken, dass die Nachricht gesendet wurde, solange sie vom zweiten Serviceautomaten nicht empfangen wurde.

Jede möglich Kombination von Nachrichten wird durch das System aller Multimengen $bags(C)$ mit Werten aus der Menge aller Kanäle C beschrieben. Insbesondere können Werte in einer Multimenge mehrfach vorkommen. So können wir mit Multimengen leicht darstellen, dass eine Nachricht mehrfach gesendet wurde. Wir führen für jeden Zustand eine Multimenge $M \in bags(C)$ von Nachrichten ein, die angibt, welche Nachrichten von einem der Serviceautomaten gesendet, aber vom anderen noch nicht empfangen wurden.

Notation Wir werden Multimengen zur besseren Unterscheidung von einfachen Mengen mit eckigen Klammern (z.B. $[a, a, b]$) angeben.

Definition 2.7 (Komponierter Serviceautomat)

Seien A und B zwei interfacekompatible Serviceautomaten. Die Komposition von A und B ist definiert als Serviceautomat

$A \oplus B = (Q_{A \oplus B}, I_{A \oplus B}, O_{A \oplus B}, \delta_{A \oplus B}, q_{0_{A \oplus B}}, F_{A \oplus B})$ mit

- $Q_{A \oplus B} = Q_A \times Q_B \times bags(C)$,
- $I_{A \oplus B} = O_{A \oplus B} = \emptyset$,
- $q_{0_{A \oplus B}} = (q_{0_A}, q_{0_B}, [])$ und
- $F_{A \oplus B} = F_A \times F_B \times \{[]\}$,

wobei die Übergangsrelation die Elemente

- $((q_A, q_B, M), \tau, (q'_A, q_B, M))$ gdw. $(q_A, \tau, q'_A) \in \delta_A$ (interner Übergang in A),
- $((q_A, q_B, M), \tau, (q_A, q'_B, M))$ gdw. $(q_B, \tau, q'_B) \in \delta_B$ (interner Übergang in B),
- $((q_A, q_B, M + [x]), \tau, (q'_A, q_B, M))$ gdw. $(q_A, x, q'_A) \in \delta_A, x \in I_A$ (A empfängt x),
- $((q_A, q_B, M + [x]), \tau, (q_A, q'_B, M))$ gdw. $(q_B, x, q'_B) \in \delta_B, x \in I_B$ (B empfängt x),
- $((q_A, q_B, M), \tau, (q'_A, q_B, M + [x]))$ gdw. $(q_A, x, q'_A) \in \delta_A, x \in O_A$ (A sendet x),
- $((q_A, q_B, M), \tau, (q_A, q'_B, M + [x]))$ gdw. $(q_B, x, q'_B) \in \delta_B, x \in O_B$ (B sendet x)

enthalten soll. ⌋

Abbildung 2.3 zeigt, wie die Komposition des Getränkeautomaten aus Abbildung 2.2 mit einem zu ihm interfacekompatiblen Serviceautomaten aussieht.

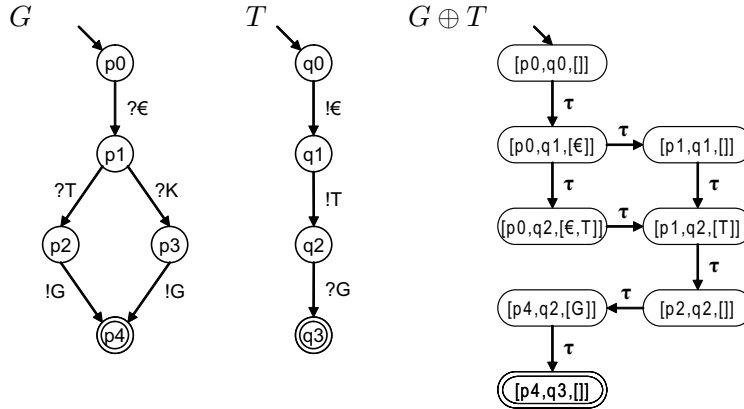


Abbildung 2.3: Die Abbildung zeigt die Komposition $G \oplus T$ des Getränkeautomaten G und eines Serviceautomaten T , der einen Tee haben möchte. Die Tripel in den Zuständen von $G \oplus T$ geben an, in welchem Zustand sich G und T befinden, sowie welche Nachrichten von einem der Serviceautomaten gesendet, vom anderen jedoch noch nicht empfangen wurden. Wir sehen, dass sämtliche Abläufe im Zustand $[p_4, q_3, []]$ enden, der Endzustand ist.

Im weiteren Verlauf wollen wir die Kommunikation der Serviceautomaten einschränken. Dies ist entscheidend bei der Berechnung der Bedienungsanleitung. Wir wollen annehmen, dass eine Nachricht $x \in C$ maximal k -mal gleichzeitig unterwegs sein kann, also maximal k -mal in der Multimenge der Nachrichten M aus Definition 2.7 vorhanden ist. Für alle $(q_A, q_B, M) \in Q_{A \oplus B}$ soll gelten, dass für alle $x \in C$ die Anzahl $M(x) \leq k$ ist. Das System aller Multimengen über C , in dem jedes $x \in C$ maximal k -mal vorhanden ist, wollen wir mit $bags_k(C)$ bezeichnen.

Die Forderung, dass ein Serviceautomat k -beschränkt ist, müssen wir stellen, da wir sonst später bei der Berechnung der Bedienungsanleitung, einen Serviceautomaten mit einem unendlichen Zustandsraum konstruieren müssten.

Definition 2.8 (k -beschränkte Kommunikation)

Seien A und B zwei interfacekompatible Serviceautomaten und $A \oplus B$ ihre Komposition. Dann ist die Kommunikation von A mit B k -beschränkt, wenn $Q_{A \oplus B} \subseteq Q_A \times Q_B \times bags_k(C)$ gilt. \lrcorner

Die Kommunikation der Serviceautomaten G und T in Abbildung 2.3 ist 1-beschränkt, da in jedem Zustand des komponierten Serviceautomaten $G \oplus T$ jeder der Nachrichten aus dem Interface maximal einmal vorkommt.

Im weiteren Verlauf dieses Kapitels wollen wir annehmen, dass wir für einen Serviceautomaten A eine Schranke k angeben können, so dass die Kommunikation mit einem Serviceautomaten B stets k -beschränkt ist.

An Hand der Komposition zweier Serviceautomaten können wir nun festlegen, wann wir ihr Zusammenspiel als sinnvoll betrachten wollen. Wir wollen sicher nicht, dass die Komposition in einen Verklemmungszustand gerät, da dieser Zustand kein definiertes Ende darstellt. Alles weitere kann beliebig bleiben – entweder hat einer der beiden Serviceautomaten die Möglichkeit, in einen anderen Zustand überzugehen, oder aber beide Serviceautomaten befinden sich gemeinsam in einem Endzustand.

Definition 2.9 (Strategie)

Seien A und B zwei interfacekompatible Serviceautomaten. Dann bezeichnen wir B genau dann als Strategie von A , wenn vom Anfangszustand des komponierten Serviceautomaten $A \oplus B$ kein Verklemmungszustand erreichbar ist. \lrcorner

Der Begriff der Strategie ist symmetrisch: Wenn B Strategie von A ist, so ist A auch Strategie von B . Bei der Komposition ist die Reihenfolge der Serviceautomaten nicht von Bedeutung.

Die beiden Serviceautomaten G und T in Abbildung 2.3 sind Strategien für einander, da sie keinen Verklemmungszustand erreichen können.

Notation Sei A ein Serviceautomat. Existiert kein Serviceautomat B , sodass B eine Strategie von A ist, so bezeichnen wir A als *nicht bedienbar*.

Bis hierhin waren die Betrachtungen jeweils symmetrisch. Zwei Serviceautomaten wurden gleichberechtigt komponiert und der Begriff der Strategie ist dementsprechend auf beide der beteiligten Serviceautomaten anwendbar. Nun jedoch wollen wir diese Symmetrie brechen: Wir werden einen Serviceautomaten gegeben haben und wollen zu diesem eine (beziehungsweise alle) Strategien berechnen. Die folgenden Definitionen werden uns dabei helfen, das Wissen, das eine mögliche Strategie über den gegebenen Serviceautomaten haben könnte, zu bestimmen.

Definition 2.10 (Situation)

Sei A ein Serviceautomat, q ein Zustand von A und M eine Multimenge von Nachrichten über dem Interface von A . Dann bezeichnen wir das Tupel (q, M) als Situation von A . \lrcorner

Der Begriff Situation ist losgelöst nicht aussagekräftig. Im Zusammenhang mit einem zweiten Serviceautomaten B können wir jedoch für einen Serviceautomaten A beschreiben, in welchen Zuständen sich A befinden kann, und welche Konfiguration von nicht empfangenen Nachrichten möglich sind, wenn wir wissen, dass B in einem bestimmten Zustand ist.

Definition 2.11 (Wissensfunktion K)

Seien A und B zwei interfacekompatible Serviceautomaten. Dann sei die Wissensfunktion (Knowledge) $K_{(B,A)} : Q_B \rightarrow \mathcal{P}(Q_A \times \text{bags}_k(C))$, die einem Zustand von B eine Menge von Situationen von A zuordnet, definiert durch die Abbildung

$$K_{(B,A)}(q_B) = \{(q_A, M) \mid (q_A, q_B, M) \text{ ist erreichbar von } q_{0_{A \oplus B}} \in Q_{A \oplus B}\}.$$

┘

Das Wissen, dass der Serviceautomat T über den Getränkeautomaten G aus der vorhergehenden Abbildung hat, ist auszugsweise in Abbildung 2.4 dargestellt.

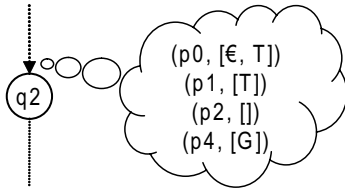


Abbildung 2.4: Wir sehen einen Ausschnitt des Serviceautomaten T , genauer dessen Zustand q_2 . Das Wissen, dass T in diesem Zustand über den Serviceautomaten G hat, ist in der Wolke angegeben. Es sind die Situationen, in denen sich G befinden kann, wenn T in q_2 ist. Das Wissen können wir leicht aus dem komponierten Serviceautomaten $G \oplus T$ in Abbildung 2.3 ableiten.

Mit den soeben eingeführten Definitionen können wir Strategien gemäß [Sch05] folgendermaßen charakterisieren:

Lemma 2.12 (Charakterisierung von Strategien)

Seien A und B zwei interfacekompatible Serviceautomaten. Dann ist B genau dann eine Strategie für A , wenn für alle $q_B \in Q_B$ und alle $(q_A, M) \in K_{(B,A)}(q_B)$ gilt:

- (q_A, q_B, M) ist ein Endzustand von $A \oplus B$; oder
- es existiert ein Übergang $(q_A, x, q'_A) \in \delta_A$ und ein Übergang von (q_A, q_B, M) in $A \oplus B$ mit x gemäß Definition 2.7; oder
- es existiert ein Übergang $(q_B, x, q'_B) \in \delta_B$ und ein Übergang von (q_A, q_B, M) in $A \oplus B$ mit x gemäß Definition 2.7.

┘

Dieses Lemma folgt direkt aus der Definition für *Strategie*.

Wir können nun sagen, welches Wissen ein Serviceautomat über einen anderen haben kann, und haben gesehen, dass wir mit Hilfe dieses Wissens sogar Strategien charakterisieren können. Jedoch sind nicht alle Situationen, in denen sich ein Serviceautomat befinden kann, gleich interessant. Besonders werden uns solche Situationen interessieren, in denen der Serviceautomat von sich aus nicht mehr agieren kann. Weder kann er einen internen Übergang vollziehen, noch kann er eine Nachricht senden, und er kann auch keine von einem anderen Serviceautomaten gesendete Nachricht empfangen. Diese Art von Situation wollen wir als stabil bezeichnen.

Definition 2.13 (Transiente und stabile Situationen)

Sei \mathfrak{M} eine Menge von Situationen. Dann heißt $(q_A, M) \in \mathfrak{M}$ genau dann transient in \mathfrak{M} , wenn ein $(q_A, x, q'_A) \in \delta_A$ existiert, so dass

- $x = \tau$ und $(q'_A, M) \in \mathfrak{M}$ oder
- $x \in I_A$, $M(x) > 0$ und $(q'_A, M - [x]) \in \mathfrak{M}$ oder
- $x \in O_A$ und $(q'_A, M + [x]) \in \mathfrak{M}$.

Anderenfalls heißt (q_A, M) stabil in \mathfrak{M} . ┘

Bei der Berechnung der Bedienungsanleitung eines Serviceautomaten werden wir das Wissen, das eine mögliche Strategie über den Serviceautomaten haben kann, darüber berechnen, welche Ereignisse eingetreten sind. Wir schauen uns die Situationen an, die der Serviceautomat ohne Interaktion mit seiner Umgebung erreichen kann. Ein Ereignis (das Senden oder Empfangen einer Nachricht) das durch die Umgebung des Serviceautomaten verursacht wurden, ändert dann die Menge der möglichen Situationen, in denen sich der Serviceautomat befinden kann.

Definition 2.14 (Ereignis)

Sei $\mathfrak{M} \subseteq Q_A \times \text{bags}_k(C)$ eine Menge von Situationen. Dann erhalten wir durch Eintreten eines Ereignisses (event) eine neue Menge von Situationen:

$$\text{event}(\mathfrak{M}, x) = \begin{cases} \{(q, M + [x]) \mid (q, M) \in \mathfrak{M}\}, & \text{falls } x \in O_A \text{ (Sendeereignis),} \\ \{(q, M) \mid (q, M + [x]) \in \mathfrak{M}\}, & \text{falls } x \in I_A \text{ (Empfangsereignis).} \end{cases}$$
┘

Durch das Eintreten eines Ereignisses ändert sich die Menge der Situationen. Einige dieser neuen Situationen könnten nun bedingen, dass der Serviceautomat selbständig in weitere Situationen gelangen kann, die in der Menge der Situationen nicht vorhanden sind. Da wir grundsätzlich alle Situationen betrachten wollen, die ein Serviceau-

tomat ohne Einwirken der Umgebung erreichen kann, wollen wir alle auf diese Weise erreichbaren Situationen zum Abschluss einer Menge von Situationen hinzufügen.

Definition 2.15 (Abschluss)

Sei \mathfrak{M} eine Menge von Situationen. Dann ist der Abschluss $cl(\mathfrak{M})$ (closure) von \mathfrak{M} induktiv definiert:

Anfang: $\mathfrak{M} \subseteq cl(\mathfrak{M})$

Schritt: Wenn $(q_A, M) \in cl(\mathfrak{M})$ und $(q_A, x, q'_A) \in \delta_A$, dann

- $(q'_A, M) \in cl(\mathfrak{M})$, wenn $x = \tau$,
- $(q'_A, M + [x]) \in cl(\mathfrak{M})$, wenn $x \in O_A$,
- $(q'_A, M - [x]) \in cl(\mathfrak{M})$, wenn $x \in I_A$ und $M(x) > 0$.

┘

Betrachten wir den komponierten Serviceautomaten $G \oplus T$ aus Abbildung 2.3. Das Wissen, das T über G im Zustand $\mathbf{q1}$ hat, ist $K_{(T,G)}(\mathbf{q1}) = \{(\mathbf{p0}, [\mathbf{€}]), (\mathbf{p1}, [])\}$. Das Ereignis $!T$ angewandt auf die Menge ergibt

$$event(K_{(T,G)}(\mathbf{q1}), T) = \{(\mathbf{p0}, [\mathbf{€}, T]), (\mathbf{p1}, [T])\}.$$

Dabei ist die letzte Situation $(\mathbf{p1}, [T])$ zwar stabil in der angegebenen Menge – sie hat keinen Nachfolger – jedoch kann G aus dieser Situation heraus die Situationen $(\mathbf{p2}, [])$ und $(\mathbf{p3}, [G])$ erreichen. Diesen Umstand beschreibt gerade der Abschluss,

$$cl(event(K_{(T,G)}(\mathbf{q1}), T)) = \{(\mathbf{p0}, [\mathbf{€}, T]), (\mathbf{p1}, [T]), (\mathbf{p2}, []), (\mathbf{p3}, [G])\}.$$

Von einem Startzustand aus können wir jetzt bereits für einen beliebigen Serviceautomaten A einen interfacekompatiblen Serviceautomaten B generieren und abhängig davon, welche Ereignisse in B auftreten, sagen, in welchen Situationen sich A befinden kann. So ist es aber auch noch möglich, Verklemmungszustände zu erreichen, oder Zustände, in denen B weiterhin Nachrichten sendet, obwohl A keine mehr empfangen kann. Diese Teile von B repräsentieren dann sicher keine Strategie von A . Für die einzelnen Zustände von B müssen wir entscheiden können, für welche Ereignisse es sinnvoll ist, sie zu verfolgen, um später eine Strategie für A zu erhalten. Dabei helfen uns Annotationen, die den einzelnen Zuständen von B zugeordnet werden.

Definition 2.16 (Annotation, Belegung)

Seien A und B zwei interfacekompatible Serviceautomaten. Dann sei für jedes $q_B \in$

Q_B die Annotation $\phi(q_B)$ folgendermaßen als aussagenlogische Formel über die Literale $C \cup \{\tau, final\}$ (mit $\tau, final \notin C$) definiert:

$$\phi(q_B) := \bigwedge_{(q_A, M) \text{ ist stabil in } K_{(B,A)}(q_B)} (\phi_1(q_A, M) \vee \phi_2 \vee \phi_3(M))$$

wobei

- $\phi_1(q_A, M) := \begin{cases} final, & \text{falls } q_A \in F_A \text{ und } M = [], \\ false, & \text{sonst,} \end{cases}$
- $\phi_2 := \tau \vee \bigvee_{x \in O_B} x,$
- $\phi_3(M) := \bigvee_{x \in I_B, M(x) > 0} x.$

Die Belegung (assignment) $ass_B(q_B) : C \cup \{\tau, final\} \rightarrow \{true, false\}$ ordnet den Literalen $x \in C \cup \{\tau\}$ genau dann den Wert $true$ zu, wenn ein q'_B existiert, so dass $(q_B, x, q'_B) \in \delta_B$, und dem Literal $final$ genau dann den Wert $true$, wenn $q_B \in F_B$. \lrcorner

Die Annotationen spiegeln genau die drei Bedingungen aus Lemma 2.12 wider.

Nun haben wir alle Begriffe zusammen, um die Bedienungsanleitung eines Serviceautomaten A zu berechnen. Wir können einen interfacekompatiblen Serviceautomaten erzeugen, für den wir für jeden Zustand bestimmen können, in welchen Situationen sich A befinden kann. Die Annotationen bilden dann die Entscheidungsgrundlage, ob ein Zustand Teil einer Strategie von A sein kann, bzw. ob es sinnvoll ist, ein bestimmtes Ereignis zu verfolgen. Probieren wir von einem Startzustand aus alle Ereignisse in beliebiger Reihenfolge durch, bis die Schranken der k -beschränkten Kommunikation erreicht sind, und streichen dann alle Zustände weg, welche die Annotationen nicht erfüllen, dann bleiben am Ende nur Zustände übrig, die Teil einer Strategie sind (oder aber kein Zustand, wenn es keine Strategie für A gibt, da A nicht bedienbar ist).

Definition 2.17 (Bedienungsanleitung)

Sei A ein Serviceautomat. Dann definieren wir induktiv eine Folge von Serviceautomaten $S_i = (Q_i, I_i, O_i, \delta_i, q_{0_i}, F_i)$:

Sei $Q_0 = \mathcal{P}(Q_A \times bags_k(C))$ und für alle i sei

- $I_i = O_A,$
- $O_i = I_A,$

- $q_0 = cl(\{(q_{0_A}, [])\})$,
- $(q, x, q') \in \delta_i$, gdw. $q, q' \in Q_i$ und $(q', []) \in cl(event(\{(q, [])\}, x))$ und
- $F_i = \{q \in Q_i \mid q \text{ ist Wartezustand von } S_i \text{ und ist mit final annotiert}\}$.

Dann sei $Q_{i+1} = \{q \mid q \in Q_i, \phi(q) \text{ ist true für die Belegung } ass_{S_i}(q)\}$.

Als Bedienungsanleitung (OG_A , Operating Guideline) von A bezeichnen wir den Serviceautomaten mit dem kleinsten i , für den $S_i = S_{i+1}$ gilt ($OG_A = S_i$), zusammen mit einer Annotation ϕ wie in Definition 2.16. \lrcorner

Hier möchten wir nochmal explizit auf die k -beschränkte Kommunikation verweisen (Def. 2.8). Würden wir keine Schranke vorgeben, wäre die Zahl der Zustände in Q_0 unendlich groß.

Notation Eine Bedienungsanleitung ist strukturell ein Serviceautomat. Zusätzlich ordnen wir jedem Zustand einer Bedienungsanleitung eine Annotation zu. Um diesen Unterschied zu verdeutlichen, werden wir im weiteren Verlauf die Zustände der Bedienungsanleitung *Knoten* und die Übergänge *Ereignisse* oder *Kanten* nennen.

Die Bedienungsanleitung ist sogar ein deterministischer Serviceautomat. Von den einzelnen Knoten gibt es für jedes Ereignis maximal eine Kante, die den entsprechenden Knoten verlässt und es gibt keine mit τ beschrifteten Kante. Die Annotationen geben Aufschluss darüber, welche Ereignisse vorhanden sein müssen oder können, damit die Bedienungsanleitung eine Strategie darstellt. Ist ein Serviceautomat A nicht bedienbar, wird in Definition 2.17 letztendlich der Startknoten entfernt, weil er die Annotation verletzt.

Als Beispiel ist in Abbildung 2.5 die Bedienungsanleitung des Getränkeautomaten G zu sehen.

Die Annotation (Def. 2.16) eines Knotens q in einer Bedienungsanleitung OG_A beschreibt, welche Ereignisse in q eintreten dürfen und welche nicht, damit ein Serviceautomat B eine Strategie von A ist. Jede Klausel der Annotation muss mit *true* belegt werden. Innerhalb einer Klausel sind die Literale disjunktiv angegeben. Das heißt, es muss eine Kante von q weggehen, die mit einem dem Literal entsprechenden Ereignis beschriftet ist. Es können auch Kanten für mehrere der Literale vorhanden sein.

Anstelle aller Kanten einer Bedienungsanleitung genügt es, eine Auswahl an Kanten zu betrachten, um die Annotation in jedem Knoten zu erfüllen. Die Annotation

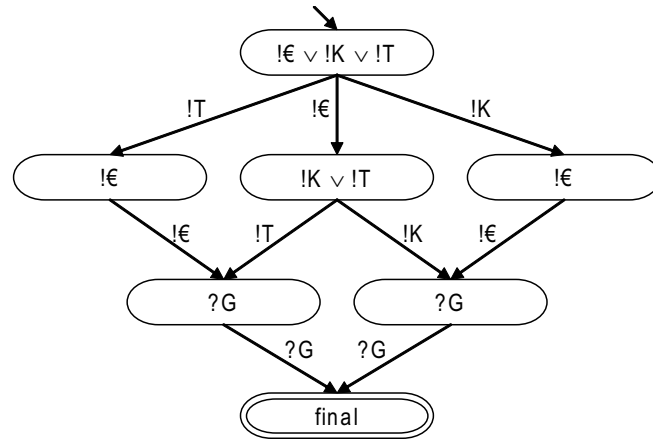


Abbildung 2.5: Dies ist die Bedienungsanleitung des Getränkeautomaten G , wie sie mit dem Werkzeug Fiona¹ ([LMSW06]) berechnet werden kann. Der Startknoten ist durch den eingehenden Pfeil, der Endknoten durch die doppelte Umrandung gekennzeichnet. Die Annotationen bestimmen, welche Ereignisse in den einzelnen Knoten eintreten dürfen. Das Eintreten eines Ereignisses, das nicht in der Annotation genannt wird, führt zu einer Verklemmung. Eine Strategie von G kann im Startknoten ein ϵ , K oder T senden und gelangt dann über die entsprechende Kante in den nächsten Knoten. Im Knoten, der mit $final$ annotiert ist, kann kein Ereignisse mehr eintreten. Die Annotation $final$ besagt, dass eine Strategie von G sich in diesem Knoten in einem Endzustand befinden muss.

eines Knotens spiegelt zudem die Anforderungen eine Strategie wider. Das ist der Grundgedanke, um Strategien eines Serviceautomaten mit Hilfe der Bedienungsanleitung zu identifizieren. Es muss möglich sein, das Verhalten einer Strategie in der Bedienungsanleitung nachzuvollziehen, das heißt die Bedienungsanleitung muss die Strategie schwach simulieren. Dadurch erhalten wir eine Relation der Zustände der Strategie mit den Knoten der Bedienungsanleitung, wobei jeder Zustand der Strategie die Annotationen der mit ihm in Relation stehenden Knoten der Bedienungsanleitung erfüllen muss.

Definition 2.18 (Matching)

Seien A und B zwei Serviceautomaten. Dann gibt es genau dann ein Matching von B mit der Bedienungsanleitung OG_A von A , wenn die Knoten und Kanten von OG_A die Zustände und Übergänge des Serviceautomaten B schwach simulieren, und jeder Zustand von B die Annotationen der mit ihm in Simulation stehenden Knoten von OG_A erfüllt. ┘

¹<http://www2.informatik.hu-berlin.de/top/tools4bpe1/fiona/>

Die Bedeutung des Matching liefert uns folgendes Lemma:

Lemma 2.19 (Matching)

Seien A und B zwei Serviceautomaten. Dann ist B genau dann eine Strategie von A , wenn B mit der Bedienungsanleitung von A matcht. ┘

Der Beweis dieser Aussage folgt sofort aus den gegebenen Definitionen und vor allem aus Lemma 2.12. Ein Serviceautomat B , der mit der Bedienungsanleitung eines anderen matcht, erfüllt die Annotationen, und da diese genau das Lemma repräsentieren, ist B eine Strategie.

Notation Ein Serviceautomat B , für den wir entscheiden wollen, ob er eine Strategie von A ist, muss zum einen in schwacher Simulation mit der Bedienungsanleitung OG_A stehen, zum anderen muss er die Annotationen der Bedienungsanleitung erfüllen. Aufgrund der Simulationsbeziehung kann ein Übergang in B höchste dann ein Literal eines Knotens von OG_A mit *true* belegen, wenn in diesem Knoten das entsprechende Ereignis ebenfalls eintreten kann. Literale, deren Ereignis in einem Knoten nicht vorhanden ist, werden deshalb von Strategien niemals mit *true* belegt. Somit werden wir im Folgenden ausschließlich die Literale in der Annotation eines Knotens betrachten, deren Ereignisse in diesem Knoten vorhanden sind.

An dieser Stelle kennen wir nun die bestehende Theorie zur Berechnung von Bedienungsanleitungen. Im nächsten Kapitel wollen wir uns mit Konsequenzen der asynchronen Kommunikation, die unserer Modellwelt zugrunde liegt, beschäftigen und das Modell der Serviceautomaten abwandeln.

3 Erweiterung der Begriffswelt

In diesem Kapitel wollen wir die Begriffe des Serviceautomaten, beziehungsweise der Bedienungsanleitung erweitern. Wir werden eine Kurzschreibweise einführen, die es uns auszudrücken erlaubt, dass bei einem Zustandsübergang eine Menge von Ereignissen in jeder beliebigen Reihenfolge eintreten kann. Für Serviceautomaten, die diese Kurzschreibweise nutzen, werden wir eine Entfaltung definieren, um den neuen Formalismus auf den bekannten Begriff der Bedienungsanleitung zurückzuführen.

Um exakt definieren zu können, welche Muster wir im nächsten Kapitel finden und durch eine Kurzschreibweise ersetzen wollen, werden wir im zweiten Teil dieses Kapitels den Begriff der Diamantenstruktur einführen.

3.1 Serviceautomaten mit nebenläufigen Ereignissen

Wenn (gleichartige) Ereignisse, zum Beispiel ausschließlich Sendeereignisse, in einem Serviceautomaten in Sequenz angegeben sind, besteht aufgrund der asynchronen Kommunikation für eine Strategie die Möglichkeit, diese in einer beliebigen Reihenfolge zu bedienen. Damit ergibt sich eine Art Nebenläufigkeit dieser Ereignisse – egal in welcher Reihenfolge sie von einer Strategie behandelt werden, sie führen zu dem gleichen Zustand. Diese Nebenläufigkeit kann bis jetzt nicht direkt durch Serviceautomaten ausgedrückt werden, d.h. sie steht einerseits einem Modellierer nicht zur Verfügung, und sie führt in vielen, vor allem praxisrelevanten Beispielen zu Diamantenstrukturen, welche die Bedienungsanleitungen von Serviceautomaten exponentiell groß werden lassen.

In offenen Workflownetzen können wir die Nebenläufigkeit von Ereignissen ausdrücken. Wenn eine Transition von mehreren Eingabepunkten liest, ist es egal, in welcher Reihenfolge diese Nachrichten ankommen, da die Transition erst schalten kann, wenn alle Nachrichten vorhanden sind. Ähnliches gilt für parallele Transitionen, wobei hier egal ist, ob es sendende oder empfangende Transitionen sind. Sie können aufgrund ihrer Nebenläufigkeit in jeder beliebigen Reihenfolge schalten. In der Übersetzung in Serviceautomaten entstehen hier Diamantenstrukturen, die jede beliebige Reihenfolge der Ereignisse beinhalten.

Wie bereits in der Einführung zu diesem Kapitel erwähnt, tritt dieser Effekt auch bei Sequenzen von Ereignissen auf, die durch eine Strategie bedient werden, da sich durch die asynchrone Kommunikation Nachrichten überholen könnten, aber nicht müssen.

Um direkt auszudrücken, dass Ereignisse in jeder beliebigen Reihenfolge stattfinden können, wollen wir die ursprüngliche Definition von Serviceautomaten (siehe Def. 2.2) abwandeln. Anstelle eines einzelnen Ereignisses pro Übergang wollen wir für Zustandsübergänge eine Multimenge von Ereignissen zulassen.

Definition 3.1 (Serviceautomat mit nebenläufigen Ereignissen, SCE)

Ein Serviceautomat mit nebenläufigen Ereignissen (Service automaton with Concurrent Events, SCE) ist ein Automat $A = (Q, I, O, \delta, q_0, F)$ mit

- einer endlichen Menge Q von Zuständen,
- einem Interface $C = I \cup O$, mit Eingabekanälen I (input) und Ausgabekanälen O (output) ($I \cap O = \emptyset$),
- einer Übergangsrelation $\delta \subseteq Q \times \text{bags}(C) \times Q$, wobei für $(q, T, q') \in \delta$ die Multimenge T der Nachrichten endlich ist,
- einem Anfangszustand $q_0 \in Q$,
- einer Menge von Endzuständen $F \subseteq Q$, wobei für alle $q \in F$ gilt, dass $x \in I$ für alle $x \in T$ mit $(q, T, q') \in \delta$ folgt, sowie
- einer Annotation ϕ .

┘

Notation Wenn wir in Analogie zur ursprünglichen Definition von Serviceautomaten den Zustandsübergang als $(q, x, q') \in \delta$ schreiben, meinen wir damit den Übergang mit der einelementigen Multimenge $(q, [x], q') \in \delta$.

Ein Beispiel für solch einen neuen Serviceautomaten ist in Abbildung 3.1 zu sehen.

Die Menge T soll für einen Zustandsübergang $(q, T, q') \in \delta_A$ eines SCE A ausdrücken, dass alle angegebenen Ereignisse nebenläufig stattfinden können. Dies entspricht einem Serviceautomaten nach Definition 2.2, in dem explizit alle möglichen Reihenfolgen dieser Ereignisse modelliert sind.

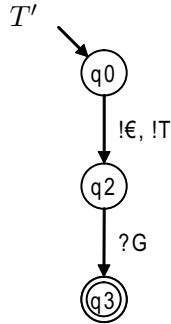


Abbildung 3.1: Wir sehen einen SCE T' , der dem Serviceautomaten T aus Abbildung 2.2 ähnelt. Jedoch sendet T' die Nachrichten € und $!T$ nicht nacheinander, sondern nebenläufig. Das heißt, dass die beiden Nachrichten in jeder beliebigen Reihenfolge gesendet werden können. In Abbildungen von SCE werden wir der Einfachheit halber die Klammern für die Multimengen weglassen, da wir ausschließlich Übergänge mit Multimengen zeigen werden.

Definition 3.2 (Entfalteter SCE)

Sei $A = (Q, I, O, \delta, q_0, F)$ ein SCE. Dann entwickeln wir eine Folge von SCE $A_i = (Q_i, I, O, \delta_i, q_0, F)$ wie folgt:

Anfang:

- $A_0 = A$

Schritt $i \rightarrow i + 1$:

- $Q_{i+1} \supseteq Q_i$
- für alle $(q, T, q') \in \delta_i$ mit T besteht aus mehr als einem Element:
 - für alle $x \in T$ existiert ein Zustand q_x in Q_{i+1} und Übergänge $(q, [x], q_x)$ und $(q_x, T - [x], q')$ in δ_{i+1} ,
 - wenn $(q, T_1, q_1), (q, T_2, q_2) \in \delta_i$ zwei verschiedenen Übergänge mit einem gemeinsamen $x \in T_1 \cap T_2$, dann existiert genau ein $q_x \in Q_{i+1}$ mit $(q, [x], q_x) \in \delta_{i+1}$ (Determinismus der Ereignisse),
- für alle $(q, [x], q') \in \delta_i$ existiert der Übergang $(q, [x], q')$ in δ_{i+1} , und
- für alle $q \in Q_{i+1} \setminus Q_i$ existiert genau ein $q' \in Q_i$ mit $(q', [x], q) \in \delta_{i+1}$ (neue Knoten in Q_{i+1} sind von genau einem alten Knoten aus Q_i über eine Kante erreichbar) und $\phi(q) = (\phi(q'))$ eingeschränkt auf die Ereignisse, die in q eintreten können).

Den SCE mit dem kleinsten i , für den gilt, dass $A_i = A_{i+1}$ wollen wir als die Entfaltung A^* des SCE A bezeichnen. ┘

Die Definition der Entfaltung beschreibt, dass wir Übergänge, die mit einer Multimenge beschriftet sind, schrittweise entfalten. Ist (q, T, q') ein Übergang mit einer mehrelementigen Multimenge T , die wir entfalten wollen, dann führen wir für je-

des Ereignis $x \in T$ statt der Kante (q, T, q') eine Kante zu einen neuen Knoten q_x ein. Von q_x führt eine Kante mit den restlichen Ereignissen $T - [x]$ zu q' . Beim Übergang von q nach q' können so die Ereignisse in T immer noch in jeder beliebigen Reihenfolge stattfinden.

Die Entfaltung soll deterministisch sein. Deshalb fordern wir in der Definition, dass ein Ereignis x von einem Knoten q immer zum gleichen Knoten q_x führt, egal für welche der in q beginnenden Kanten (q, T_1, q_1) und (q, T_2, q_2) wir die Entfaltung betrachten.

Die letzte Forderung der Definition, dass neu eingefügt Knoten genau einen Vorgänger haben, stellen wir, damit wir für die Entfaltung ausschließen können, dass sich zwei Kanten, die wir entfalten, unvorhergesehen Zustände teilen. Die Knoten, die wir durch die Entfaltung einer Kante erhalten, sollen sich von den Knoten aus der Entfaltung einer anderen Kante immer unterscheiden (bis auf die Ausnahme, dass wir Determinismus verlangen). Außerdem weisen wir in jedem Schritt den neu hinzugefügten Knoten eine Annotation zu. Diese leiten wir aus dem eindeutigen Vorgänger ab, indem wir dessen Annotation auf die Literale einschränken, deren Ereignisse in dem neuen Knoten eintreten können. Die Annotationen bereits existierender Knoten werden nicht geändert.

Die Definition 3.1 stellt demnach eine *Kurzschreibweise* dar, deren Semantik wir in Definition 3.2 erklärt haben.

Ein Beispiel dieser Semantik ist in Abbildung 3.2 zu sehen.

Die im Folgenden betrachteten Serviceautomaten werden jeweils ein SCE sein. Bedienungsanleitungen stellen einen Spezialfall eines SCE dar: Jede Kante ist mit einer einelementigen Multimenge beschriftet. Umgekehrt können wir zu jedem SCE A eine Bedienungsanleitung finden, indem wir A entfalten. Deshalb wollen wir einen SCE grundsätzlich auch als Bedienungsanleitung betrachten.

Wenn wir einen Serviceautomaten B mit einem SCE A matchen wollen, dann matchen wir B mit der Entfaltung von A . Die Entfaltung von A ist in ihrer Struktur gerade eine Bedienungsanleitung im klassischen Sinne, und wir können somit die Matchingdefinition 2.18 auf sie anwenden.

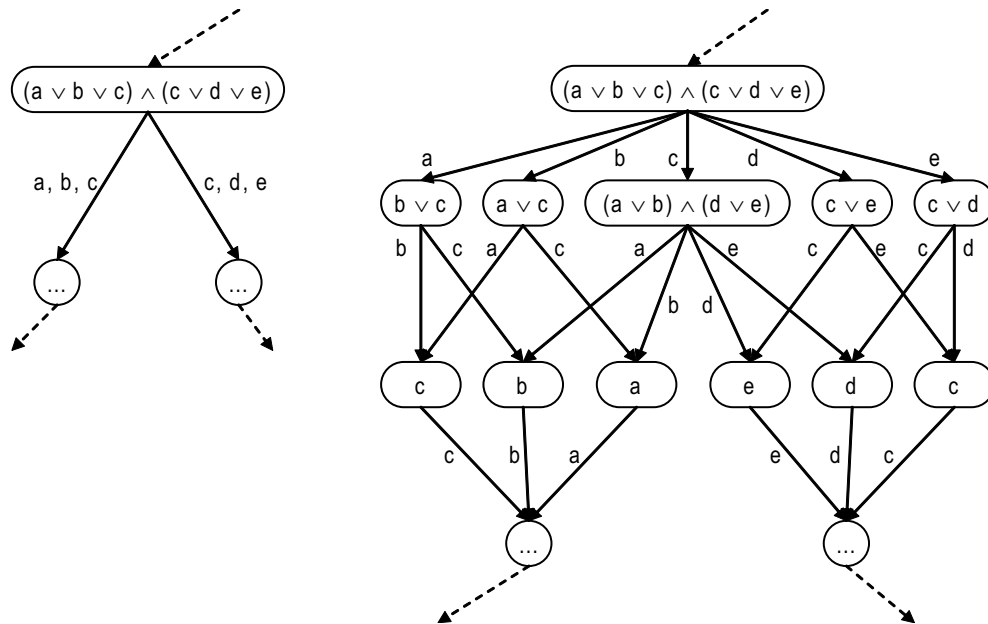


Abbildung 3.2: Wir sehen links einen Teil eines SCE mit Kurzschreibweise. Der SCE kann zwei Übergänge realisieren, zum einen durch das nebenläufige Eintreten von a, b, c und zum anderen durch das nebenläufige Eintreten von c, d, e . Rechts sehen wir die Entfaltung der entsprechenden Übergänge. Die Ereignisse a, b, c und c, d, e treten jeweils in beliebiger Reihenfolge auf. Für das Ereignis c gibt es in der Entfaltung nur eine Kante, obwohl es in der Kurzschreibweise in zwei Übergängen vorkam.

Hinweis

Andreas Kerlin benutzt in seiner Diplomarbeit ([Ker07]) eine Notation für Serviceautomaten, die der hier in Definition 3.1 vorgestellten syntaktisch gleicht. Jedoch unterscheidet sich die Semantik der beiden Definitionen.

Im Laufe seiner Arbeit entwickelt er die gleiche Notation für Serviceautomaten wie wir. In Kerlins Semantik können Sendeereignisse in einem Übergang erst dann eintreten, nachdem die Empfangsereignisse eingetreten sind.

Wir verstehen jedoch unter einem Übergang, der mit mehreren Nachrichten beschriftet ist, dass dieser Übergang durch das Senden und Empfangen dieser Nachrichten in jeder beliebigen Reihenfolge statt finden kann.

Besondere Bedeutung hat dieser Unterschied für die Form der Bedienungsanleitung. Bei Kerlin werden Übergänge mit mehreren Ereignissen eingeführt, es wird aber keine Reduktion der Bedienungsanleitung erreicht. Sind die Ereignisse, mit denen eine Kante beschriftet ist, unabhängig, existiert für jede Reihenfolge, in der diese Ereignisse eintreten können, zusätzlich eine entsprechende Folge von Übergängen in der Bedienungsanleitung. Dies steht im Widerspruch zu unserem Ziel, durch diese Notation eine Reduktion in Bedienungsanleitungen zu bewirken. Deshalb ist es wichtig hervorzuheben, dass wir eine Kurzschreibweise einführen, die *Nebenläufigkeit* von Ereignissen ausdrückt.

3.2 Diamantenstrukturen

In diesem Abschnitt wollen wir den Begriff eines Diamanten festlegen. Wir werden ihn im nächsten Kapitel als Grundlage für jedes der dort vorgestellten Muster verwenden. Die Rechtfertigung dafür, dass wir Diamanten durch eine Kurzschreibweise ersetzen können, liefert ein Lemma am Ende des Abschnitts, das die Entfaltung eines SCE mit Diamantenstrukturen in Verbindung bringt.

Die Diamantenstruktur, die wir in diesem Abschnitt charakterisieren wollen, werden Teilstrukturen eines *SCE* sein. Diese wollen wir Subautomaten nennen.

Definition 3.3 (Subautomat)

Sei $A = (Q_A, I, O, \delta_A, q_{0_A}, F_A)$ ein SCE. Dann heißt $Z = (Q_Z, I, O, \delta_Z, q_{0_Z}, F_Z)$ genau dann Subautomat von A , wenn:

- $Q_Z \subseteq Q_A$,
- $\delta_Z = \delta_A \cap (Q_Z \times \text{bags}(C) \times Q_Z)$ und
- $F_Z = F_A \cap Q_Z$.

┘

Definition 3.4 (Echter Subautomat)

Seien $A = (Q_A, I, O, \delta_A, q_{0_A}, F_A)$ ein SCE und $Z = (Q_Z, I, O, \delta_Z, q_{0_Z}, F_Z)$ ein Subautomat von A . Dann heißt Z genau dann echter Subautomat von A , wenn $Q_Z \subsetneq Q_A$.

┘

Um einen Subautomaten Z in einem SCE A zu eindeutig zu charakterisieren, reicht uns eine Menge von Zuständen $Q_Z \subseteq Q_A$ aus. Die Definition eines Subautomaten (Def. 3.3) legt fest, dass zwei Knoten in Z genau dann durch eine Kante verbunden sind, wenn sie in A durch eine Kante verbunden sind. Die Menge der Endzustände in Z ergibt sich kanonisch aus der Menge der Endzustände in A . Lediglich den Startknoten des Subautomaten Z müssen wir zusätzlich explizit angeben.

Definition 3.5 (Induzierter Subautomat)

Sei $A = (Q_A, I, O, \delta_A, q_{0_A}, F_A)$ ein SCE und $Q_Z \subseteq Q_A$ eine Teilmenge der Zustände von A . Dann heißt der SCE $Z = (Q_Z, I, O, \delta_Z, q_{0_Z}, F_Z)$ mit

- $\delta_Z = \delta_A \cap (Q_Z \times \text{bags}(C) \times Q_Z)$,
- $q_{0_Z} \in Q_Z$ und
- $F_Z = F_A \cap Q_Z$

der von Q_Z in A induzierte Subautomat. ┘

Eine intuitive Vorstellung des Begriffes Diamantenstruktur haben wir bereits. Es ist ein spezielle Subautomat einer Bedienungsanleitung, in dem eine bestimmte Menge an Ereignissen in beliebiger Reihenfolge eintritt. Dabei existiert ein *erster Knoten*, in dem jede Reihenfolge startet, und ein *letzter Knoten*, in dem jede Reihenfolge endet.

Definition 3.6 (Diamant)

Sei A ein SCE und $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein Subautomat von A . Dann bezeichnen wir D genau dann als Diamanten in A , wenn

- es ausgezeichnete Knoten q_{start} und $q_{end} \in Q_D$ gibt,
- eine endliche Multimenge $T = [x_1, x_2, \dots, x_n] \in \text{bags}(C)$ von Ereignissen existiert, sodass
 - jede Folge von n Kanten $(q_{start}, x_1, q_1), (q_1, x_2, q_2), \dots, (q_{n-1}, x_n, q_{end}) \in \delta_D$ in q_{end} endet, also $q_n = q_{end}$, und $[x_1, x_2, \dots, x_n] = T$ gilt,
 - es Knoten $q_1, q_2, \dots, q_{n-1} \in Q_D$ für jede Reihenfolge $\{i_1, i_2, \dots, i_n\} = \{1, 2, \dots, n\}$ der Nachrichten $[x_1, x_2, \dots, x_n] = T$ gibt, sodass $(q_{start}, x_{i_1}, q_1), (q_1, x_{i_2}, q_2), \dots, (q_{n-1}, x_{i_n}, q_{end}) \in \delta_D$,
- für alle Knoten $q \in Q_D \setminus \{q_{start}, q_{end}\}$ in D gilt, dass q weder ein Anfangs- noch ein Endknoten von A ist, also $q \notin \{q_{0_A}\} \cup F_A$.

┘

Notation Sei D ein Diamant mit den Bezeichnungen aus Definition 3.6. Dann nennen wir q_{start} den *ersten Knoten* und q_{end} den *letzten Knoten* des Diamanten. Jeden anderen Knoten in D wollen wir einen *internen Knoten* des Diamanten nennen.

Der letzte Punkt der Definition besagt, dass außer den Knoten q_{start} und q_{end} kein Knoten q des Diamanten D ein Anfangs- oder Endknoten des Serviceautomaten A sein darf, in dem der Diamant liegt. Beim Ersetzen des Diamanten in den nächsten Abschnitten, wird in der Regel der Knoten q aus der Menge der Knoten Q_A entfernt. Wäre q ein End- oder der Anfangsknoten in A , würde sich somit das Verhalten von A ändern.

Die Menge der Endknoten eines Diamanten ist für uns ohne Belang. Insbesondere fordern wir *nicht*, dass q_{end} ein Endknoten ist. Es ist der *letzte* Knoten des Diamanten.

Notation Die Multimenge T aus Definition 3.6 wollen wir die *Ereignismenge* T_D des Diamanten D nennen.

Ein typischer Diamant ist in Abbildung 3.3 zu sehen.

Wenn wir die Definitionen für einen Diamanten in einem SCE (Def. 3.6) mit der Definition für die Entfaltung eines SCE (Def. 3.2) vergleichen, dann können wir feststellen:

Lemma 3.7 (Entfalteter SCE und Diamanten)

Sei A ein SCE und $(q, T, q') \in \delta_A$ ein Übergang in A . Dann sei A^* der entfaltete SCE von A und Q_T die Menge der Zustände, die an der Entfaltung des Übergangs $(q, T, q') \in \delta_A$ beteiligt sind. Dann ist der durch Q_T induzierte Subautomat (mit $q_{start_T} = q$) ein Diamant in A . ┘

Der Beweis dieser Aussage ergibt sich unmittelbar aus dem Vergleich der Definitionen für die Entfaltung (Def. 3.2) und eines Diamanten (Def. 3.6). Diese Aussage bildet die Grundlage für die Reduktion, gibt sie uns doch einen Zusammenhang zwischen Diamanten in Bedienungsanleitungen und der von uns eingeführten Kurzschreibweise für das Eintreten von Ereignissen in beliebiger Reihenfolge.

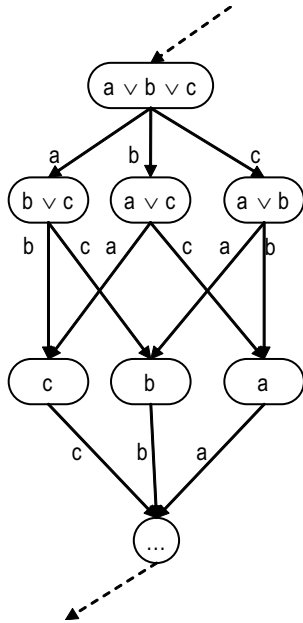


Abbildung 3.3: Im obersten Knoten, dem *ersten Knoten* des Diamanten, sind die Ereignisse a , b , c möglich. Verfolgen wir die einzelnen Ereignisse, sehen wir, dass die Ereignisse in jeder beliebigen Reihenfolgen eintreten können, bis wir zum untersten Knoten kommen, dem *letzten Knoten* des Diamanten.

Die Ereignismenge dieses Diamanten ist dementsprechend $[a, b, c]$.

Entfalten wir mehrere Multimengen T in einem SCE, entstehen so mehrere Diamanten. Insbesondere können so mehrere Diamanten mit dem gleichen ersten Knoten q entstehen, je nachdem wie viele Kanten in q mit mehrelementigen Multimengen beschriftet sind.

Wenn wir in den nächsten Abschnitten die Pattern beschreiben, möchten wir nicht beliebige Diamanten betrachten, sondern solche, die nicht Teil eines anderen Diamanten sind. Die Pattern sollen in diesem Sinne maximal sein.

Definition 3.8 (Maximaler Diamant)

Sei A ein SCE und D ein Diamant in A . Dann heißt D maximal, wenn es keinen Subautomaten Z von A gibt, sodass D ein echter Subautomat von Z ist, und Z ist ein Diamant in A . ┘

4 Reduktion der Bedienungsanleitung

Im vorherigen Kapitel haben wir eine Kurzschreibweise für die Übergänge in einem SCE eingeführt. Mit der Definition der Entfaltung eines SCE A beschreiben wir einen Serviceautomaten A^* , in dem statt der Kurzschreibweise für eine Kante $(q, T, q') \in \delta_A$ explizit jede mögliche Reihenfolge von Nachrichten aus T vorkommt. Die Entfaltung einer Kante haben wir als Diamantenstruktur identifiziert. In diesem Kapitel wollen wir in die andere Richtung gehen, um so eine Reduktion auf Bedienungsanleitungen zu definieren. Wir wollen Teilstrukturen in einer gegebenen Bedienungsanleitung erkennen, in denen eine Menge von Ereignissen in beliebiger Reihenfolge auftritt. Wir werden Muster festlegen, die es uns erlauben, diese Teilstrukturen in zu definierender Weise durch die Kurzschreibweise zu ersetzen und somit die Bedienungsanleitung kompakter darzustellen.

4.1 Elementare Pattern

In diesem Abschnitt beschäftigen wir uns mit elementaren Pattern, von denen wir erwarten können, sie häufig in Bedienungsanleitungen zu finden, und die wir vor allem leicht durch eine Kurzschreibweise ersetzen können.

4.1.1 Pattern eines einfachen Diamanten

Motivation. Wenn wir einen Diamanten D in einem SCE A betrachten, dann enthält dieser laut Definition nur Kanten, welche die Zustände des Diamanten untereinander verbinden. Betrachten wir D als Teil von A , ist es möglich, dass interne Knoten von D mit Knoten, die nicht im Diamanten liegen, durch eine Kante verbunden sind.

Das folgende Muster beschreibt einen Diamanten D in einer Bedienungsanleitung, dessen interne Knoten ausschließlich untereinander verbunden sind. Das heißt, höchstens der erste oder der letzte Knoten von D sind mit einem Knoten außerhalb des Diamanten verbunden.

Pattern. Abbildung 4.1 zeigt das Pattern eines *einfachen Diamanten*.

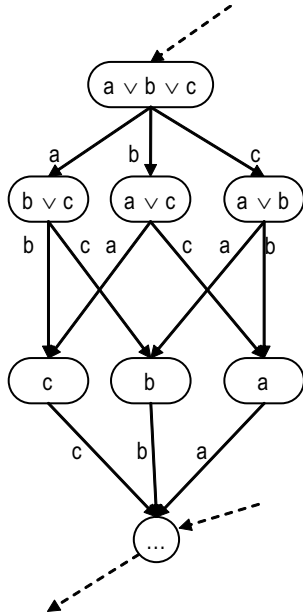


Abbildung 4.1: Das nebenstehende Bild zeigt das Pattern eines *einfachen Diamanten*. Die Annotation des *ersten Knotens* (ganz oben) ist ein Alternative von Ereignissen. Verfolgen wir diese Ereignisse, gelangen wir zum *letzten Knoten* des Diamanten (ganz unten). Die Ereignisse können dabei in jeder beliebigen Reihenfolge eintreten. Die Annotationen der Knoten unterscheiden sich von denen ihrer Vorgänger nur durch das Ereignis, über das sie erreicht wurden.

In die internen Knoten des Diamanten führen weder Kanten von außerhalb hinein, noch führen Kanten heraus zu Knoten, die nicht zum Diamanten gehören.

Die gestrichelte Linie, die zum ersten Knoten führt, soll für die Möglichkeit stehen, dass Kanten von außerhalb zu diesem Knoten führen. Entsprechend dürfen beim letzten Knoten Kanten sowohl hin als auch weg führen.

Definition 4.1 (Pattern: Einfacher Diamant)

Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0OG}, F_{OG})$ die Bedienungsanleitung eines Serviceautomaten. Dann heißt der Subautomat $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein einfacher Diamant in OG , wenn

- (B1) D ein maximaler Diamant in OG ist,
- (B2) für ein Ereignis $(q_1, T, q_2) \in \delta_{OG}$ mit $q_1 \in Q_{OG} \setminus Q_D$ und $q_2 \in Q_D$ folgt, dass $q_2 = q_{start}$ oder $q_2 = q_{end}$ (es gibt von außen keine Kanten zu internen Zuständen),
- (B3) für ein Ereignis $(q_1, T, q_2) \in \delta_{OG}$ mit $q_1 \in Q_D$ und $q_2 \in Q_{OG} \setminus Q_D$ folgt, dass $q_1 = q_{end}$ (es gibt keine Kanten von internen Zuständen nach außen), und
- (B4) für ein Ereignis $(q_1, T, q_2) \in \delta_D$ mit $q_2 \neq q_{end}$ gilt, dass sich die Annotationen von q_1 und q_2 höchstens durch das Literal des Ereignisses x unterscheiden, also $\phi(q_1) = \phi(q_2) \vee x$.

┘

Der im Pattern 4.1 beschriebene Diamant ist in der Form einfach, dass er in gewisser Weise losgelöst von der Bedienungsanleitung OG ist. Es gibt keinen Knoten $q_1 \in Q_{OG} \setminus Q_D$ außerhalb des Diamanten, von dem aus ein Ereignis in den Diamanten führt – maximal führt von q_1 ein Ereignis zum *ersten* oder *letzten Knoten* des Diamanten (Bedingung (B2)). Genauso gibt es keinen Knoten $q_2 \in Q_{OG} \setminus Q_D$, der durch ein Ereignis von einem der Zustände des Diamanten erreichbar ist, mit Ausnahme des *letzten Knotens* (Bedingung (B2)).

Die Bedingung (B4), die wir an die Annotation stellen, ist mit Bedacht auf die Entfaltung gewählt. Wenn wir den Diamanten durch eine Kante mit einer Multimenge von Ereignissen ersetzen, wollen wir sicherstellen, dass die Entfaltung dieser Kante zu Annotationen führt, die denen der Knoten entsprechen, die wir ersetzt haben. Der letzte Knoten eines Diamanten wird in der Regel die Bedingung verletzen. Entweder sind im letzten Knoten neue Ereignisse möglich, sodass die Annotationen des letzten Knotens ohne Zusammenhang zur Annotation eines seiner Vorgänger ist. Oder der letzte Knoten ist mit *final* annotiert, sodass die Bedingung auch verletzt wäre.

Ersetzung. Wie wir gesehen haben, sind alle Knoten eines einfachen Diamanten D , bis auf den ersten und den letzten Knoten von D , ohne Zusammenhang mit den anderen Knoten einer Bedienungsanleitung. Damit können wir das Pattern des einfachen Diamanten folgendermaßen durch einen Übergang mit Kurzschreibweise ersetzen:

Definition 4.2 (Ersetzung einfacher Diamant)

Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ die Bedienungsanleitung eines Serviceautomaten und $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein einfacher Diamant in OG . Dann führt die Ersetzung des einfachen Diamanten D zu $OG' = (Q_{OG'}, I, O, \delta_{OG'}, q_{0_{OG}}, F_{OG})$ mit

- $Q_{OG'} = (Q_{OG} \setminus Q_D) \cup \{q_{start}, q_{end}\}$ und
- $\delta_{OG'} = (\delta_{OG} \setminus (Q_D \times bags(C) \times Q_D)) \cup \{(q_{start}, T_D, q_{end})\}$ für T_D ist die Nachrichtenmenge von D .

┘

Die Ersetzung wird in Abbildung 4.2 visuell umgesetzt. Die internen Knoten des Diamanten werden entfernt und stattdessen existiert eine Kante, die den ersten und letzten Knoten miteinander verbindet, und mit den Ereignissen beschriftet ist, die im Diamanten eintreten können.

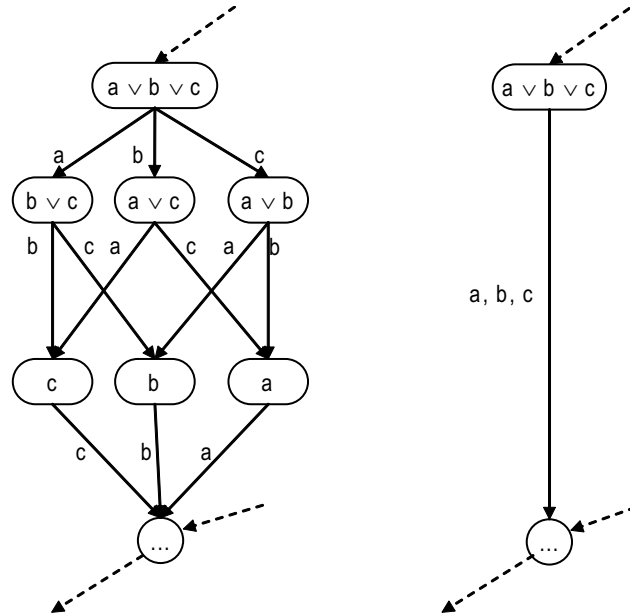


Abbildung 4.2: Links sehen wir das Muster für einen *einfachen Diamanten*, rechts dessen Ersetzung.

In Abbildung 4.3 sehen wir ein Beispiel, in dem das Ereignis **b** mehrfach in jeder Folge von Ereignissen vorkommt. Das geht aus der Annotation des ersten Knotens nicht hervor, fällt aber auch unter die Definition eines einfachen Diamanten.

Ein weiteres Beispiel, das auch eine Diamantenstruktur darstellt, dies aber weniger offensichtlich, ist in Abbildung 4.4 zu sehen.

Korrektheit. Für einen Serviceautomaten A berechnen wir die Bedienungsanleitung, um alle Strategien von A charakterisieren zu können. Laut Definition *matcht* jede Strategie von A mit der Bedienungsanleitung von A . Diese Eigenschaft muss durch die Ersetzung erhalten bleiben.

Satz 4.3 (Korrektheit der Ersetzung)

*Sei A ein Serviceautomat, OG_A dessen Bedienungsanleitung mit einem einfachen Diamanten D und B ein zu A interfacekompatibler Serviceautomat. Sei OG'_A der SCE, in dem D ersetzt wurde. Dann *matcht* B genau dann mit OG'_A , wenn B mit OG_A *matcht*.* ┘

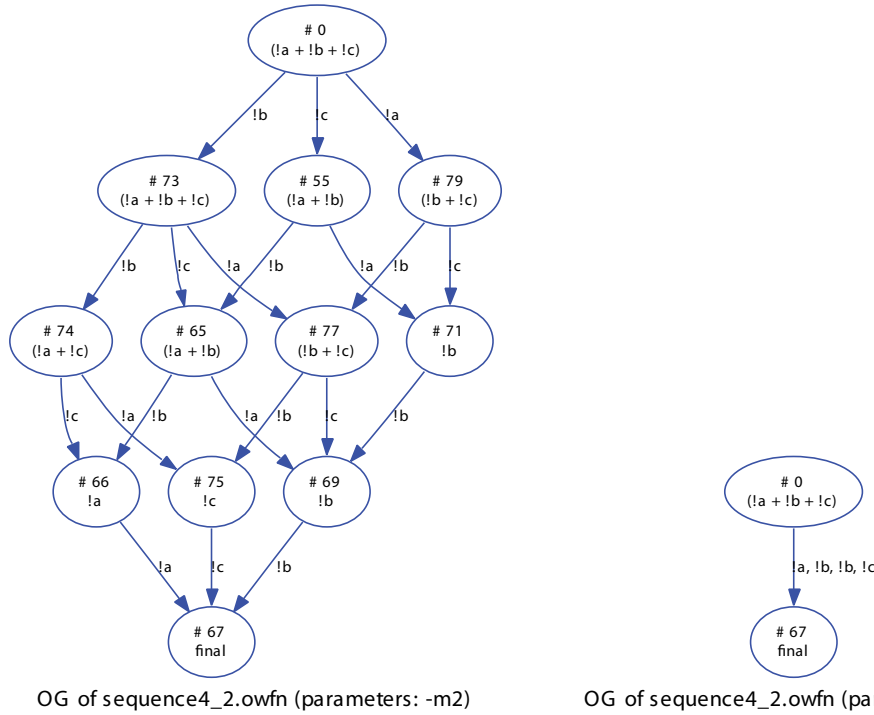


Abbildung 4.3: Links sehen wir eine Diamantenstruktur, die nach der Berechnung eines Serviceautomaten mit dem Werkzeug **Fiona** entstanden ist. In jeder Folge von Ereignissen tritt das Ereignis **b** zweimal auf. Eine Ersetzung durch eine Kurzschreibweise ist auch hier möglich, wie rechts zu sehen ist. (Die Disjunktion von Ereignissen wird in der Ausgabe durch das + in den Annotationen ausgedrückt.)

Beweis

Diese Behauptung folgt unmittelbar aus Lemma 3.7. Sei $(q_{start}, T_D, q_{end}) \in \delta_{OG'_A}$ die Kante, durch die der Diamant D ersetzt wurde. Dann beschreibt die Entfaltung genau einen Diamanten mit den Nachrichten T_D , der isomorph zu D ist, und in dem die Annotationen gleich den Annotationen in D sind. Damit können genau die gleichen Services mit OG'_A wie mit OG_A matchen. \square

An der Ersetzung sehen wir das enorme Reduktionspotential. Gemessen an der Anzahl der Nachrichten, die in einem Diamanten empfangen und gesendet werden, ist der Diamant exponentiell groß. Nach der Ersetzung interessieren uns nur noch die Nachrichten, die in dem Diamant auftreten, und die wir an den Übergang schreiben

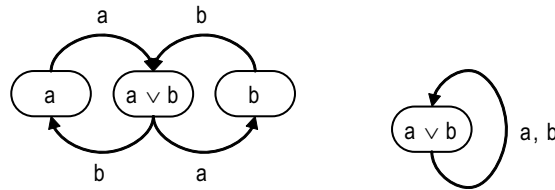


Abbildung 4.4: Links sehen wir eine Diamantenstruktur. Sie wirkt etwas ungewöhnlich, weil der *erste* und der *letzte* Knoten des Diamanten, der mittlere Knoten im Bild, gleich sind. Das ist durch die Definition eines Diamanten abgedeckt. Wir verlangen dort lediglich, dass es einen ersten und letzten Knoten gibt, und dass wir vom ersten Knoten immer zum letzten Knoten gelangen, egal in welcher Reihenfolge die Ereignisse auftreten. Das ist auch hier gegeben.

Auf der rechten Seite sehen wir dann die entsprechende Ersetzung.

müssen. Dies lässt sich mit linearem Speicherplatzaufwand in Vergleich zur Zahl der Nachrichten realisieren.

Implementation. Nachdem wir gezeigt haben, dass wir einen einfachen Diamanten D in einer Bedienungsanleitung OG durch die Kurzschreibweise ersetzen können, stellt sich die Frage, wie wir D algorithmisch bestimmen können. Das folgende Programm in Pascal-ähnlicher Notation beschreibt eine Breitensuche ab einem Knoten q_start . Für die Knoten, die bei der Suche betrachtet werden, überprüfen wir die Eigenschaften, die für die Knoten eines Diamanten allgemein, sowie für die eines *einfachen* Diamanten im Speziellen gelten müssen.

Der hier vorgestellte Algorithmus ist nicht eins-zu-eins in ein Programm umsetzbar, sondern soll eine Vorlage zu Implementation bieten. Er enthält die wichtigen Gedanken für eine praktische Umsetzung.

```

1  PROCEDURE FindeEinfachenDiamanten(Knoten q_start)
    BEGIN
        // benötigte Funktionen/Objekte:
        // Vorgänger(Knoten q), Nachfolger(Knoten q) -
5  // Menge der Vorgänger, Nachfolger eines Knoten q
        // Ereignis(Knoten q, Knoten q') -
        // Das Ereignis, dass q mit q' verbindet
        // (leer, wenn es keine solche Kante gibt)
        // phi(Knoten q) -
10 // die Annotation des Knoten q
    
```

```

// Länge(phi(Knoten q)) -
//     Anzahl Literale in der Annotation des Knoten q
// Nachrichten(Knoten q) -
//     Abbildung auf Multimenge von Nachrichten, initial []
15 Menge D;    // die Knoten des Diamanten
Queue Q;     // Queue für die Breitensuche
Knoten q_end = NULL;

20 D.einfügen(q_start); // q_start im Diamanten
// alle Nachfolger von q_start untersuchen
FOR ALL q' IN Nachfolger (q_start) DO Q.anfügen(q');

// solange Knoten in Q sind, führe die Schleife aus
25 WHILE NOT Q.leer() DO
BEGIN
Knoten q = Q.erstes_Element();
// Keine Zyklen erlauben
IF q IN D THEN DO RETURN FALSE;
30 // Betrachte alle Vorgänger von q
FOR ALL Knoten q' IN Vorgänger(q) DO
BEGIN
// Vorgänger noch nicht gesehen, dann kein einfacher
// Diamant (hineinführende Kante!)
35 IF NOT q' in D THEN DO RETURN FALSE;
// Bedingung an Annotation nicht erfüllt,
// dann kein Diamant
IF NOT phi(q') == (phi(q) OR Ereignis(q', q)) THEN DO
RETURN FALSE;
40 // die eigene Nachrichtenmenge schon gesetzt?
IF Nachrichten(q) == [] THEN DO
Nachrichten(q) = Nachrichten(q') + [Ereignis(q',q)];
ELSE DO
45 IF NOT Nachrichten(q) == Nachrichten(q') + [Ereignis(q',q)]
THEN DO RETURN FALSE;
END
// q verletzt keine Bedingung
D.einfügen(q);

```

```

// Kandidat für letzten Knoten schon gefunden?
50 // nein, alle Nachfolger von q in die Queue packen
// sofern sie nicht drin sind; ja, zeigen sie auf q_end?
FOR ALL q' IN Nachfolger (q) DO
BEGIN
// letzten Knoten gesondert behandeln
55 IF q_end == NULL THEN DO
BEGIN
// ist Nachfolger Kandidat für q_end?
IF (Länge(phi(q)) = 1 AND NOT phi(q) == phi(q')) THEN DO
BEGIN
60 q_end = q';
Nachrichten(q_end) = Nachrichten(q) + [Ereignis(q,q_end)];
END
ELSE
IF NOT q' IN Q THEN Q.anfügen(q');
65 END
ELSE DO
BEGIN
IF NOT q' == q_end THEN DO RETURN FALSE;
IF NOT Nachrichten(q_end) = Nachrichten(q) + [Ereignis(q,q')]
70 THEN DO RETURN FALSE;
END
END
END
D.einfügen(q_end);
75 RETURN D;
END

```

Die Ereignismenge T_D , die den Diamanten charakterisiert, entspricht der Menge $\text{Nachrichten}(q_end)$, die wir für den als q_end identifizierten Knoten berechnet haben.

Der Algorithmus soll einen einfachen Diamanten, der in q_start beginnt, sicher erkennen. Beginnt dagegen in q_start kein einfacher Diamant, muss der Algorithmus abbrechen.

Satz 4.4 (Korrektheit der Implementation)

Sei OG_A die Bedienungsanleitung eines Serviceautomaten A und D ein Subautomat in OG_A . Dann liefert $Q_D = \text{FindeEinfachenDiamanten}(\text{Knoten } q_{start})$ genau dann die Zustände Q_D von D zurück, wenn D ein einfacher Diamant ist. \lrcorner

Beweis

→: Wir wollen zeigen: Seien $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ eine Bedienungsanleitung, $q_{start} \in Q_{OG}$ ein Knoten der Bedienungsanleitung und Q_D die Menge der Knoten, welche die Funktion $\text{FindeEinfachenDiamanten}(q_{start})$ zurück geliefert hat. Dann wird durch die Menge Q_D ein einfacher Diamant D in OG induziert.

Wir müssen nun die einzelnen Punkte der Definition eines einfachen Diamanten (Def. 4.1) überprüfen.

(B1) D ist ein maximaler Diamant:

- * **Der Knoten q_{start} (q_start) existiert in D** , es ist der erste Knoten, den wir betrachten.
- * **Der Knoten q_{end} (q_end) existiert in D** , wir fügen ihn als letztes in die Menge Q_D ein.
- * **Eine Ereignismenge T_D existiert**, es ist die Nachrichtenmultimenge, die q_{end} zugeordnet ist ($\text{Nachrichten}(q_{end})$). Sie enthält genau die Ereignisse, die zwischen q_{start} und q_{end} auftreten:
 - **Jeder Folge von Ereignissen in D führt von q_{start} zu q_{end}** und ist somit endlich. Angenommen, nicht jede Folge führt zu q_{end} : Da die Struktur der Bedienungsanleitung endlich ist, finden wir entweder einen Zyklus, einen Knoten ohne Nachfolger, der die Bedingung für q_{end} nicht erfüllt, – der Algorithmus bricht jeweils ab – oder wir finden einen Kandidaten für q_{end} . Sei γ eine Folge von Übergängen in D , die nicht in q_{end} endet. Jeder Knoten in γ erfüllt die Bedingungen bezüglich der Annotation, da sonst der Algorithmus abgebrochen hätte. Da γ endlich sein muss (gleiches Argument wie zuvor), heißt das, dass es einen Knoten q gibt, dessen Nachfolger die Bedingung für q_{end} erfüllt. Jedoch haben wir q_{end} zuvor schon gefunden, dann muss der Algorithmus aber

in Zeile 67 abbrechen, da der Nachfolger von q nicht q_{end} ist. Das ist im Widerspruch zur Annahme.

Jede Folge von Ereignissen von q_{start} nach q_{end} ergibt die Menge T_D . Anderenfalls bricht der Algorithmus in Zeile 68/69 ab, da wir spätestens für q_{end} unterschiedlich Nachrichtenmengen durch die verschiedenen Vorgänger erhalten.

- **Jede Reihenfolge der Nachrichten $T_D = [x_1, x_2, \dots, x_n]$ kann durch eine Folge von Ereignissen in D realisiert werden.** Sei $\{i_1, i_2, \dots, i_n\} = \{1, 2, \dots, n\}$ eine beliebige Reihenfolge der Nachrichten. Angenommen, diese Reihenfolge lässt sich in D realisieren. Sei nun γ eine beliebige Folge von Ereignissen in D . Dann gibt es einen Präfix $\{i_1, i_2, \dots, i_{k-1}\}, k < n$, sodass die Ereignisse $x_{i_1}, x_{i_2}, \dots, x_{i_{k-1}}$ in γ stattfinden können und zu einem Knoten $q_k \in Q_D$ führen, in dem das Ereignis x_{i_k} nicht eintreten kann. Da wir jedoch wissen, dass γ zu q_{end} führt, und dort das Ereignis eingetreten sein muss (alle Nachrichten in T_D müssen auftreten), gibt es einen Knoten q_l in der Folge γ , in dem das Ereignis x_{i_k} eintritt. Laut Definition enthält die Annotation von q_l das Literal x_{i_k} . Laut Voraussetzung enthält auch jeder Vorgänger von q_l das Literal x_{i_k} . Iterieren wir weiter zurück, gelangen wir zum Knoten q_k , dessen Annotation auch das Literal x_{i_k} enthalten muss. Damit gibt es aber auch eine Kante, die das Ereignis x_{i_k} realisiert. Das ist ein Widerspruch zur Annahme, dass dieses Ereignis in q_k nicht möglich ist.

Die **Maximalität** ist dadurch gegeben, dass der Algorithmus keine eingehenden oder ausgehenden Kanten aufgezeigt hat (siehe die nächsten beiden Punkte).

- (B2) **Es gibt kein Ereignis, das in die Menge Q_D führt**, genauer, sei $q \in Q_D \setminus \{q_{start}, q_{end}\}$ ein Knoten von D , und $q_1 \in Q_{OG} \setminus Q_D$ ein Knoten der Bedienungsanleitung, der nicht zu D gehört. Dann gibt es kein Ereignis $(q_1, T, q) \in \delta_{OG}$. Anderenfalls hätte der Algorithmus in Zeile 35 abgebrochen, da $q_1 \notin Q_D$, aber Vorgänger von q .
- (B3) **Es gibt kein Ereignis, das aus der Menge Q_D führt**, genauer, sei $q \in Q_D \setminus \{q_{end}\}$ ein Knoten von D , und $q_2 \in Q_{OG} \setminus Q_D$ ein Knoten der Bedienungsanleitung, der nicht zu D gehört. Dann gibt es kein Ereignis $(q, T, q_2) \in \delta_{OG}$. Anderenfalls gibt es Knoten $q, q' \in Q_D$ in D , sodass q'

Vorgänger von q ($(q', y, q) \in \delta_D$), und das Ereignis x ist in q möglich, in q' aber nicht. Dann unterscheiden sich die Annotationen von q und q' zusätzlich zum y um dieses x , weshalb der Algorithmus in Zeile 38 abbricht.

- (B4) **Die Annotationen erfüllen die Bedingungen**, das heißt für ein Ereignis $(q', T, q) \in \delta_D$ mit $q_2 \neq q_{end}$ gilt, dass sich die Annotationen von q_1 und q_2 höchstens durch das Literal des Ereignisses x unterscheiden, also $\phi(q_1) = \phi(q_2) \vee x$. Das wird gerade durch Zeile 38 sicher gestellt, in welcher der Algorithmus abbricht, wenn die Bedingung verletzt ist.

Somit induziert Q_D einen einfachen Diamanten in der Bedienungsanleitung OG .

- ←: *Wir wollen zeigen:* Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ eine Bedienungsanleitung. Wenn $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein einfacher Diamant in OG ist, dann liefert `FindeEinfachenDiamanten(q_{start})` die Menge Q_D zurück.

Mindestens jeder Knoten von Q_D wird vom Algorithmus betrachtet. Für q_{start} ist dies offensichtlich, weil dort der Algorithmus gestartet wird. Sei $q \in Q_D \setminus \{q_{start}, q_{end}\}$ ein beliebiger Knoten des Diamanten D . Dieser wird durch den Diamanten erreicht, denn q liegt in keinem Zyklus (kein Abbruch in Zeile 29; Eigenschaft des Diamanten), alle Vorgänger von q wurden bereits betrachtet (kein Abbruch in Zeile 35; Eigenschaft der Breitensuche, da alle Vorgänger von q mit einem Ereignis weniger erreicht werden), die Annotation von q unterscheidet sich von der Annotation von eines Vorgängers q' nur durch das Ereignis x , wenn $(q', x, q) \in \delta_D$ (kein Abbruch in Zeile 38; Forderung in der Definition eines Diamanten), und der Knoten ist nur über eine bestimmte Teilmenge von Ereignissen $T \subset T_D$ erreichbar (kein Abbruch in Zeile 44; Eigenschaft, des Diamanten).

Die Forderungen an `q_end` im Algorithmus sind durch $q_{end} \in Q_D$ des Diamanten erfüllt. Für jeden der Vorgänger von q_{end} ist die Bedingung in Zeile 58 erfüllt, aber für keinen Knoten vorher. Da alle Folgen von Ereignissen von q_{start} nach q_{end} jeweils die gleichen Ereignisse beinhalten, wird die Abbruchbedingung in Zeile 69 nie erfüllt.

Folglich terminiert der Algorithmus mit Rückgabe der Menge Q_D .

□

Nachdem wir gezeigt haben, dass der Algorithmus das Muster erkennt und wir es folglich ersetzen können, interessiert uns der zeitliche Aufwand, um es zu erkennen. Sinnvollerweise wollen wir ein Muster möglichst schnell erkennen, da wir das Problem der Größe einer Bedienungsanleitung nicht in ein Zeitproblem umwandeln wollen.

Die Funktion `FindeEinfachenDiamanten` basiert auf einer Breitensuche. Deren Laufzeit ist linear in der Anzahl der betrachteten Knoten und Ereignisse. Für jeden Knoten, den der Algorithmus betrachtet, werden zusätzlich noch höchstens dessen Vorgänger und Nachfolger betrachtet. Das liegt aber im Rahmen der Zeitkomplexität der Breitensuche.

Mit dem Muster *einfacher Diamant* beschreiben wir sogar drei Fälle. Die Ereignisse in q_{start} könnten ausschließlich aus Empfangsereignissen, ausschließlich aus Sendeereignissen als auch aus beiden Arten von Ereignissen bestehen. Diese drei Fälle sind durch das Muster abgedeckt. Da der Algorithmus das Muster erkennt, unabhängig davon, welche Ereignisse möglich sind, ist eine Unterscheidung dieser Fälle nicht zwangsläufig notwendig. Jedoch können wir den Algorithmus für den Spezialfall, dass die Annotation nur aus Empfangsereignissen besteht, effizienter gestalten.

Einfacher Diamant für Empfangsereignisse

Implementation. Sei $q_{start} \in Q_{OG_A}$ ein Knoten in einer Bedienungsanleitung OG_A , auf den wir den Algorithmus `FindeEinfachenDiamanten` anwenden wollen, und seien in q_{start} ausschließlich Empfangsereignisse möglich. Steht uns das Wissen für die Zustände von OG_A zur Verfügung, ist es sinnvoll, nicht die Annotation, sondern die stabile Situation eines Knotens zu betrachten. Im Wissen $K_{(OG_A, A)}(q_{start})$ des Knotens q_{start} gibt es dann eine stabile Situation (q_A, M) , wobei die Nachrichtenmenge M alle Nachrichten $M_R \subseteq M$ enthält, die von A gesendet und von OG_A noch nicht empfangen wurden, sowie die Nachrichten $M \setminus M_R$, die von OG_A gesendet und von A noch nicht empfangen werden konnten. Uns interessiert die Multimenge M_R , bestimmt sie doch die Annotation von q_{start} . Jedes Empfangsereignis $x \in I_{OG_A}$ steht genau dann in der Annotation von q_{start} , wenn es in der Multimenge $x \in M_R$ enthalten ist.

Im Gegensatz zur Annotation können wir mit der Multimenge M_R genau sagen, wie oft ein bestimmtes Empfangsereignis x eintreten muss, da x entsprechend oft in M_R enthalten ist. Damit vereinfacht sich das Kriterium, das für die Annotation eines Knoten und dessen Vorgänger in einem Diamanten gelten muss, auf einen

Vergleich der stabilen Situationen der beiden Knoten. Sei $(q', x, q) \in \delta_D$ ein Ereignis in einem Diamanten D , der nur aus Empfangsereignissen besteht. Dann gilt für die stabilen Situationen $(q_A, M) \in K_{(OG_A, A)}(q)$ und $(q'_A, M') \in K_{(OG_A, A)}(q')$ von q und q' , dass $(q'_A, M') = (q_A, M + [x])$ gilt. Die stabilen Situationen unterscheiden sich nur durch das Ereignis x in der zugehörigen Nachrichtenmultimenge (vergleiche *event* Def. 2.14).

Entsprechend ist der Nachfolger eines Knotens q höchstens dann der letzte Knoten des Diamanten q_{end} , wenn in der stabilen Situation des Knotens q noch genau ein Empfangsereignis enthalten ist.

Mit diesen Betrachtungen können wir den Algorithmus `FindeEinfachenDiamanten` nicht um Größenordnungen beschleunigen. Jedoch ist der Umgang mit Multimengen einfacher und effizienter als die Handhabung von aussagenlogischen Formeln, wie sie in den Annotationen genutzt werden.

4.2 Erweiterte Pattern

In diesem Abschnitt werden wir Muster von Diamanten kennen lernen, die wir nicht einfach durch die Kurzschreibweise ersetzen können, sondern bei denen wir bestimmte Randbedingungen beachten müssen, die wir jeweils vorstellen werden.

Wir werden sehen, dass wir den Algorithmus aus Abschnitt 4.1.1 wiederverwenden können und ihn nur geringfügig an die neuen Gegebenheiten anzupassen brauchen.

4.2.1 Pattern eines Diamanten mit hineinführenden Kanten

Motivation. Im vorhergehenden Abschnitt haben wir nur Diamantenstrukturen betrachtet, die losgelöst von der restlichen Bedienungsanleitung sind. Das heißt, dass die internen Knoten des Diamanten nur über den Knoten q_{start} erreicht werden können, und dass man von diesen immer in den Knoten q_{end} gelangt. In diesem Muster wollen wir jedoch auch zulassen, dass eine Kante in den Diamanten hineinführt. Das vorhergehende Muster erweitern wir, in dem wir eine der Bedingungen weglassen.

Pattern. Abbildung 4.5 zeigt das Pattern eines *Diamanten mit hineinführenden Kanten*. In dem Pattern ist es möglich, dass interne Zustände des Diamanten über eine hineinführende Kante mit Zuständen außerhalb des Diamanten verbunden sind.

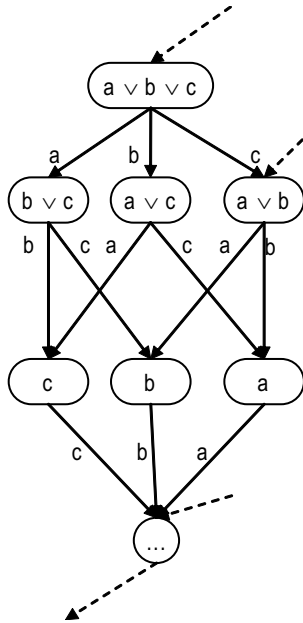


Abbildung 4.5: Das abgebildete Pattern eines *Diamanten mit hineinführenden Kanten* gleicht in Wesentlichen einem *einfachen Diamanten*. Der mit $a \vee b \vee c$ annotierte Knoten ist der *erste Knoten* des Diamanten. Die einzelnen Ereignisse a , b , c können von dort aus wieder in beliebiger Reihenfolge stattfinden und führen zu einem gemeinsamen Knoten, dem *letzten Knoten* des Diamanten. Anders als bisher dürfen Kanten hier von einem Knoten außerhalb des Diamanten zu einem internen Knoten führen. Kanten, die aus dem Diamanten herausführen, wollen wir noch nicht zulassen.

Der erste und der letzte Knoten des Diamanten können über die angedeuteten Kanten (gestrichelt) mit anderen Knoten außerhalb des Diamanten verbunden sein.

Wenn wir die Definition des Patterns mit der eines einfachen Diamanten (Def. 4.1) vergleichen, fällt uns auf, dass sie sich nur in einem Punkt unterscheiden. Den Fall, dass eine Ereignis von einem Knoten, der nicht zum Diamanten gehört, zu einem Knoten führt, der zum Diamanten gehört, schließen wir beim einfachen Diamanten explizit aus (Bedingung (B2)). In Definition 4.5 ist dies jedoch möglich, da wir solche Kanten in diesem Fall nicht aus dem Muster ausschließen wollen. Entsprechend geben wir die Bedingung (B2) nicht an.

Definition 4.5 (Pattern: Diamant mit hineinführenden Kanten)

Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ die Bedienungsanleitung eines Serviceautomaten. Dann heißt der Subautomat $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein *Diamant mit hineinführenden Kanten* in OG , wenn

- (B1) D ist ein maximaler Diamant in OG ,
- (B3) für ein Ereignis $(q_1, T, q_2) \in \delta_{OG}$ mit $q_1 \in Q_D$ und $q_2 \in Q_{OG} \setminus Q_D$ folgt, dass $q_1 = q_{end}$, und
- (B4) für ein Ereignis $(q_1, T, q_2) \in \delta_D$ mit $q_2 \neq q_{end}$ gilt, dass sich die Annotationen von q_1 und q_2 höchstens durch das Literal des Ereignisses x unterscheiden, also $\phi(q_1) = \phi(q_2) \vee x$.

┘

Ersetzung. Wenn wir das Muster 4.5 des Diamanten mit hineinführenden Kanten ersetzen, müssen wir die Zustände des Diamanten gesondert betrachten, die gerade eine hineinführende Kante besitzen, also mit einem Knoten verbunden sind, der nicht im Diamanten liegt.

Definition 4.6 (Ersetzung Diamant mit hineinführenden Kanten)

Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ die Bedienungsanleitung eines Serviceautomaten und $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein Diamant mit hineinführenden Kanten in OG , wobei $Q_E = \{q \in Q_D \setminus \{q_{start}, q_{end}\} \mid \exists q' \in Q_{OG} \setminus Q_D : \exists (q', T, q) \in \delta_{OG}\}$ die Menge der Zustände in D ist, die eine hineinführende Kante besitzen. Dann führt die Ersetzung des Diamanten mit hineinführenden Kanten D zu $OG' = (Q_{OG'}, I, O, \delta_{OG'}, q_{0_{OG}}, F_{OG})$ mit

- $Q_{OG'} = (Q_{OG} \setminus Q_D) \cup \{q_{start}, q_{end}\} \cup Q_E$ für q_{start} ist der erste Knoten und q_{end} ist der letzte Knoten des Diamanten und
- $\delta_{OG'} = (\delta_{OG} \setminus (Q_D \times \text{bags}(C) \times Q_D)) \cup \{(q_{start}, T_D, q_{end})\} \cup \{(q_e, T, q_{end}) \mid q_e \in Q_E, T \subset T_D \text{ Menge der Ereignisse, die in } q_e \text{ noch eintreten können}\}$ für T_D ist die Nachrichtenmenge von D .

┘

Die Ersetzung des Pattern eines Diamanten mit hineinführenden Kanten ist in Abbildung 4.6 angegeben.

Die Ersetzung besagt, dass wir alle Knoten bis auf die mit eingehenden Kanten aus der Knotenmenge der Bedienungsanleitung OG entfernen. Dafür führen wir eine Kante $(q_{start}, T_D, q_{end})$ zwischen q_{start} und q_{end} ein. Für jeden Knoten $q_e \in Q_E$ mit einer eingehenden Kante müssen wir zudem die Kante (q_e, T, q_{end}) hinzufügen. In OG ist es möglich, von q_e zu q_{end} gelangen, dass muss nach der Ersetzung auch noch gewährleistet sein. Da q_e Knoten in einem Diamanten ist, können die Ereignisse T , die zwischen q_e und q_{end} noch stattfinden können, in jeder beliebigen Reihenfolge auftreten. Somit brauchen wir nicht den Teil der Diamantenstruktur zu behalten, der von q_e aus erreichbar ist, sondern wir können direkt die Kante (q_e, T, q_{end}) hinzufügen.

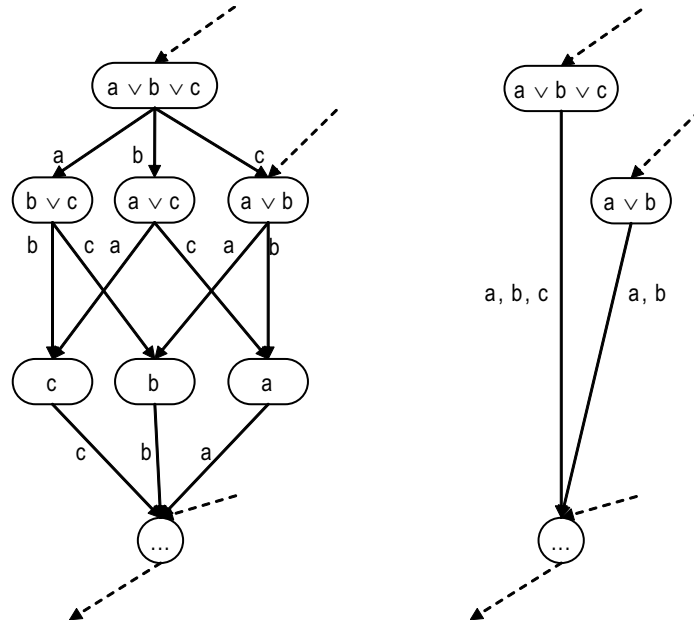


Abbildung 4.6: Links sehen wir das Muster für den *Diamanten mit hineinführenden Kanten*, rechts dessen Ersetzung.

Korrektheit. Für das Muster eines Diamanten mit hineinführenden Kanten müssen wir zeigen, dass die Ersetzung die Matchingeigenschaft der Bedienungsanleitung bewahrt.

Satz 4.7 (Korrektheit der Ersetzung)

Sei A ein Serviceautomat, OG_A dessen Bedienungsanleitung mit einem Diamanten mit hineinführenden Kanten D und B ein zu A interfacekompatibler Serviceautomat. Sei OG'_A der SCE, in dem D ersetzt wurde. Dann matcht B genau dann, mit OG'_A , wenn es mit OG_A matcht. \lrcorner

Beweis

Wenn B mit dem *ersten* Knoten des Diamanten D in OG matcht, dann auch mit OG'_A , und umgekehrt. Das geht aus den Betrachtungen für den einfachen Diamanten hervor. Eine herein führende Kante $(q', y, q) \in \delta_{OG_A}$ hat darauf keinen Einfluss. Wenn B aber auch mit den Zuständen q' und q in OG_A matcht, dann ist dies genau so auch in OG'_A möglich. Das Matching der Zustände zwischen q und q_{end} geht wieder

auf die Argumentation eines einfachen Diamanten zurück. Es war vor der Ersetzung möglich, mit den Nachfolgern von q zu matchen, dies ist in der Entfaltung von OG'_A genauso möglich. \square

Implementation. Wie wir gesehen haben, unterscheidet sich die Definition eines *Diamanten mit hineinführenden Kanten* nur in dem Punkt von der Definition eines *einfachen Diamanten*, der eingehende Kanten bei einfachen Diamanten verbietet. Entsprechend können wir den Algorithmus aus Abschnitt 4.1.1 wiederverwenden und müssen ihn nur an einer Stelle anpassen.

```
1  PROCEDURE FindeDiamantenEingehendeKanten(Knoten q_start)
   BEGIN
```

Der Algorithmus ist genau der gleiche wie `FindeEinfachenDiamanten`, wir ändern *ausschließlich* die hier gezeigten Zeilen 31 bis 35. Alle anderen Programmzeilen sind identisch zum Algorithmus `FindeEinfachenDiamanten`.

Anstatt abzubrechen, wenn wir eine hineinführende Kante in den Diamanten finden, merken wir uns den Knoten, zu dem diese Kante führt, in einer Menge Q_E .

```

   FOR ALL Knoten q' IN Vorgänger(q) DO
   BEGIN
     // Vorgänger noch nicht gesehen,
     // dann hineinführende Kante
35  IF NOT q' in D THEN DO Q_E.einfügen(q);
```

Am Ende des Algorithmus geben wir die zusätzlich berechnete Menge Q_E zurück, um diese Zustände entsprechend der Ersetzung eines Diamanten mit hineinführenden Kanten behandeln zu können.

```
75  RETURN D, Q_E;
   END
```

Satz 4.8 (Korrektheit der Implementation)

Sei OG_A die Bedienungsanleitung eines Serviceautomaten A und D ein Subautomat in OG_A . Dann liefert `FindeDiamantenEingehendeKanten(q_{start})` genau dann die Knoten Q_D von D zurück, wenn D ein Diamant mit eingehenden Kanten ist. \lrcorner

Beweis

Der Beweis verläuft vollkommen analog zu der Argumentation in Abschnitt 4.1.1. Wir haben genau den Punkt im Algorithmus geändert, in dem ausgeschlossen werden sollte, dass Kanten in den Diamanten hineinführen. Somit finden wir mit dem angegebenen Algorithmus genau die Diamanten mit hineinführenden Kanten. \square

4.2.2 Pattern eines Diamanten mit herausführenden Kanten

Motivation. Genauso gut, wie Kanten in einen Diamanten hineinführen können, soll es auch möglich sein, dass Kanten aus einem Diamanten herausführen. Wenn wir die Ereignisse in einem Diamanten betrachten, ist es möglich, dass durch das Eintreten eines oder mehrerer dieser Ereignisse, in einem internen Zustand des Diamanten ein Ereignis eintreten kann, das aus dem Diamanten herausführt. Genauso wie das Muster eines Diamanten mit hineinführenden Kanten ist dieses Muster eine Erweiterung des einfachen Diamanten, in dem wir eine Bedingung, die an einen einfachen Diamanten gestellt wird, weglassen.

Pattern. Abbildung 4.7 zeigt das Pattern eines *Diamanten mit herausführenden Kanten*. In diesem ist es möglich, dass aus internen Zuständen Ereignisse aus dem Diamanten herausführen.

Im Vergleich mit dem einfachen Diamanten (Def. 4.1) fehlt hier wieder eine Bedingung, nämlich genau die, welche herausführende Kanten verbietet (Bedingung (B3)). Andererseits müssen wir die Bedingung an die Annotation eines Knoten (Bedingung (B4)) anders fassen, da die Kanten, die aus dem Diamanten herausführen, entsprechend in der Annotation enthalten sind. Damit verletzen wir die ursprüngliche Bedingung.

Definition 4.9 (Pattern: Diamant mit herausführenden Kanten)

Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ die Bedienungsanleitung eines Serviceautomaten. Dann heißt der Subautomat $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein Diamant mit herausführenden Kanten in OG , wenn

- (B1) D ist ein maximaler Diamant in OG ,
- (B2) für ein Ereignis $(q_1, T, q_2) \in \delta_{OG}$ mit $q_1 \in Q_{OG} \setminus Q_D$ und $q_2 \in Q_D$ folgt, dass $q_2 = q_{start}$ oder $q_2 = q_{end}$ (siehe Def. 3.6 eines Diamanten),

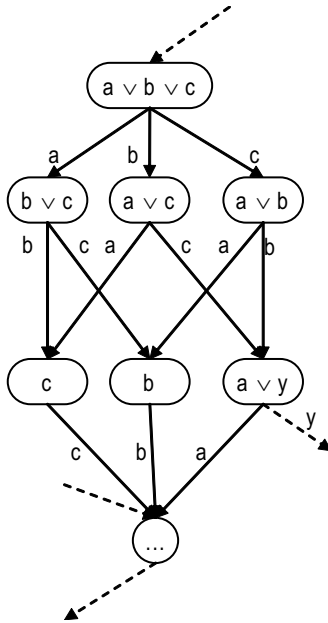


Abbildung 4.7: Das abgebildete Pattern eines *Diamanten mit herausführenden Kanten* gleicht in Wesentlichen wieder einem *einfachen Diamanten*. Der mit $a \vee b \vee c$ annotierte Knoten ist der *erste Knoten* des Diamanten. Die einzelnen Ereignisse a, b, c können von dort aus wieder in beliebiger Reihenfolge statt finden und führen zu einem gemeinsamen Knoten, dem *letzten Knoten* des Diamanten.

Jedoch ist es nun möglich, von dem mit $a \vee y$ annotierten Knoten aus den Diamanten mit dem Ereignis y zu verlassen.

(B_4^*) für ein Ereignis $(q_1, [x], q_2) \in \delta_D$ mit $q_2 \neq q_{end}$ folgt, dass $\phi(q_1) = \phi(q_2) \vee x$, wenn wir in ϕ nur den Teil der Klausel betrachten, dessen Literale Ereignissen im Diamanten entsprechen.

┘

Wenn in einem Diamanten ein Ereignis y aus einem internen Knoten aus dem Diamanten herausführt, dann umfasst die Annotation dieses y . Der Knoten q , für den wir als erstes feststellen, dass dort das Ereignis y eintreten kann, verletzt unsere Anforderung an die Annotation. Für ein $(q', [x], q) \in \delta_D$ Ereignis, über das wir q erreichen, gilt dann

$$\phi(q') \neq \phi(q') \vee y = \phi(q_2) \vee x.$$

Deshalb schauen wir uns in der Annotation nur die Literale der Klausel an, die bereits im ersten Knoten eines Diamanten enthalten waren – die anderen Literale streichen wir quasi zur Überprüfung der Annotation weg.

Aus dieser Definition folgt, dass wir als Ereignisse für herausführende Kanten, nur solche zulassen, die nicht im Diamanten selbst eintreten können. Dafür können beliebig viele Ereignisse aus einem Knoten des Diamanten herausführen.

Ersetzung. Bei der Ersetzung müssen wir beachten, dass wir jeden Knoten, von dem aus eine Kante aus einem Diamanten herausführt, erhalten bleibt. Der Knoten muss weiterhin von q_{start} aus erreichbar sein. Ansonsten können wir wieder die gleiche Ersetzung wie bisher durchführen.

Definition 4.10 (Ersetzung Diamant mit herausführenden Kanten)

Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ die Bedienungsanleitung eines Serviceautomaten und $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein Diamant mit herausführenden Kanten in OG , wobei $Q_E = \{q \in Q_D \setminus \{q_{start}, q_{end}\} \mid \exists q' \in Q_{OG} \setminus Q_D : \exists (q, T, q') \in \delta_{OG}\}$ die Menge der Zustände in D ist, die eine herausführende Kante besitzen. Dann führt die Ersetzung des Diamanten mit herausführenden Kanten D zu $OG' = (Q_{OG'}, I, O, \delta_{OG'}, q_{0_{OG}}, F_{OG})$ mit

- $Q_{OG'} = (Q_{OG} \setminus Q_D) \cup \{q_{start}, q_{end}\} \cup Q_E$ für q_{start} ist der erste Knoten und q_{end} ist der letzte Knoten des Diamanten und
- $\delta_{OG'} = (\delta_{OG} \setminus (Q_D \times \text{bags}(C) \times Q_D) \cup \{(q_{start}, T_D, q_{end})\} \cup \{(q_{start}, T, q_e) \mid q_e \in Q_E, T \subset T_D \text{ Menge der Ereignisse, die in } q_e \text{ seit } q_{start} \text{ eingetreten sind}\})$ für T_D ist die Nachrichtenmenge von D .

┘

Wichtig an der Definition ist, dass sie nichts über die Annotationen der Knoten aussagt. Jeder Knoten in der Ersetzung OG' soll genau die Annotation besitzen, die er in der Bedienungsanleitung OG hatte. Wenn ein Knoten q_e ein Knoten des Diamanten ist, von dem eine Kante aus dem Diamanten herausführt, dann enthält seine Annotation auch die Ereignisse des Diamanten, die in q_e eintreten können. In der Ersetzung sind diese Kanten nicht mehr enthalten. Wenn wir jedoch OG' entfalten (Def. 3.2), wird die Kante wiederhergestellt. Dann ist es wichtig, dass die Annotation unverändert vorhanden ist, um die Äquivalenz der Bedienungsanleitung und der Entfaltung von OG' bezüglich der Matchingdefinition zu gewährleisten.

Die Ersetzung des Pattern eines Diamanten mit hineinführenden Kanten ist in Abbildung 4.8 angegeben. Der Knoten, der mit $a \vee y$ annotiert ist, besitzt in der Ersetzung keine mit a beschriftete Kante mehr. Strichen wir a in diesem Fall heraus, führte die Entfaltung nicht zu dem Muster, dass wir ersetzt haben, sondern zu einer Struktur, in der die entsprechende Annotation nur noch y lautet. Eine Strategie, die nur den Diamanten, aber nicht die Kante y enthält, matcht dann nicht mit der Entfaltung.

Korrektheit. Diese Ersetzung bewahrt wieder die Menge der Strategien.

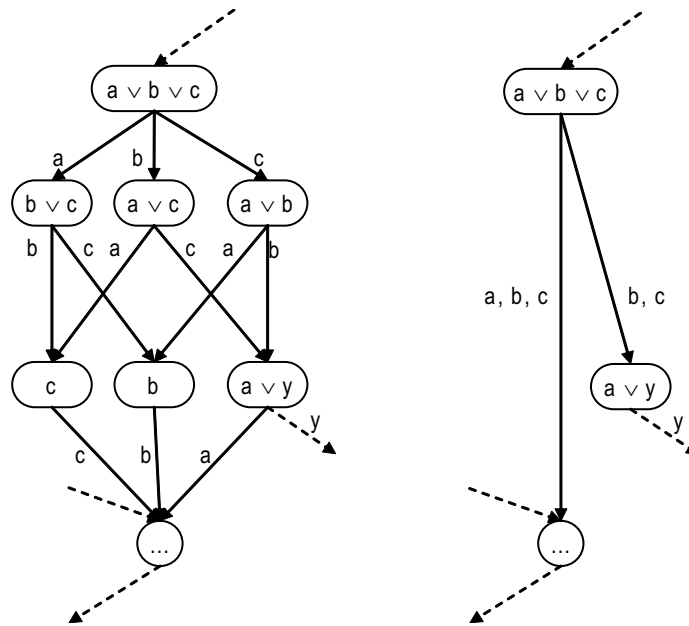


Abbildung 4.8: Links sehen wir das Muster für den *Diamanten* mit herausführenden Kanten, rechts dessen Ersetzung.

Satz 4.11 (Korrektheit der Ersetzung)

Sei A ein Serviceautomat, OG_A dessen Bedienungsanleitung mit einem *Diamanten* mit herausführenden Kanten D und B ein zu A interfacekompatibler Serviceautomat. Sei OG'_A der SCE, in dem D ersetzt wurde. Dann matcht B genau dann, mit OG'_A , wenn es mit OG_A matcht. \lrcorner

Beweis

Analog zu den bisherigen Beweisen lässt sich die Frage des Matching auf den Fall des *einfachen Diamanten* zurückführen. Das Matching mit den Knoten des *Diamanten* funktioniert in der Bedienungsanleitung OG_A genauso wie in der reduzierten Form OG'_A , solange wir nur die im ersten Knoten möglichen Ereignisse betrachten. Sei (q, y, q') eine Kante, die aus dem *Diamanten* herausführt. Diese ist dann beispielsweise mit $a \vee y$ annotiert (siehe Abbildung 4.8). Nach der Ersetzung gibt es im Knoten q explizit keine Kante a mehr. Implizit ist sie jedoch vorhanden. Beim Entfalten von $(q_{start}, T_D, q_{end})$ wird der Knoten wieder in den *Diamanten* D mit einbezogen, da die Entfaltung der Kante (q_{start}, T, q) mit $T \subset T_D$ deterministisch in die Entfaltung von

$(q_{start}, T_D, q_{end})$ eingegliedert werden muss. Damit ist es möglich, in q das Matching in OG'_A genau dann zu erfüllen, wenn wir es in OG_A erfüllen können. \square

Implementation. Genauso wie bei dem Muster für Diamanten mit hineinführenden Kanten brauchen wir den Algorithmus für *einfache Diamanten* nur geringfügig abzuändern, um die Unterschiede in den Definitionen zu berücksichtigen.

```
1  PROCEDURE FindeDiamantenAusgehendeKanten(Knoten q_start)
    BEGIN
```

Der Algorithmus ist im Wesentlichen der gleiche wie `FindeEinfachenDiamanten`, diesmal ersetzen wir jedoch die Zeile 36 bis 38 durch die hier angegebenen. Eine ausgehende Kante haben wir gefunden, wenn neue Ereignisse möglich sind. Den Knoten, für den das eintritt, und die entsprechenden Ereignisse merken wir uns in einer Menge Q_E .

```

        // Bedingung an Annotation nicht erfüllt,
        // dann kein Diamant
        IF NOT phi(q') == (phi(q) OR Ereignis(q', q)) THEN DO
        BEGIN
40      IF NOT (phi(q') OR Klausel) == (phi(q) OR Ereignis(q', q))
          THEN DO Q_E.einfügen(q);
          ELSE DO RETURN FALSE;
        END
```

Die Variable `Klausel` soll dabei für die Klausel der neuen Ereignisse stehen. Für den Knoten q kennen wir zudem die Multimenge der Nachrichten `Nachrichten(q)`, also die Ereignisse, über die wir von q_{start} aus zu q gelangt sind. Diese Menge beschreibt genau den Übergang von q_{start} nach q . Am Ende des Algorithmus geben wir die zusätzlich berechnete Menge Q_E zurück, um diese Zustände entsprechend der Ersetzung eines Diamanten mit herausführenden Kanten behandeln zu können.

```
75  RETURN D, Q_E;
    END
```

Satz 4.12 (Korrektheit der Implementation)

Sei OG_A die Bedienungsanleitung eines Serviceautomaten A und D ein Subautomat

in OG_A . Dann liefert $Q_D = \text{FindeDiamantenAusgehendeKanten}(\text{Knoten } q_{\text{start}})$ genau dann die Knoten Q_D von D zurück, wenn D ein Diamant mit eingehenden Kanten ist. \lrcorner

Beweis

Genauso wie für das Pattern des Diamanten mit hineinführenden Kanten ist der Beweis analog zu Abschnitt 4.1.1. Den Algorithmus haben wir in einem Punkt geändert. Anstatt herausgehende Kanten zu verbieten, erlauben wir sie, indem die Bedingung an die Annotation in Zeile 38 die Ereignisse, die durch die herausführenden Kante eintreten können, gesondert betrachtet. \square

Die beiden Muster *Diamant mit hinein führender Kante* und *Diamant mit heraus führender Kante* können natürlich kombiniert werden. Dazu werden die jeweils vorgestellten Veränderungen am ursprünglichen Algorithmus eingefügt und die Knoten, die hineinführende, beziehungsweise herausführende Kanten haben, getrennt gemerkt.

4.2.3 Pattern Diamant einer Klausel

Motivation. Bis jetzt haben wir nur Pattern betrachtet, in denen der Startknoten aus genau einer Klausel bestand. Nun ist es durchaus möglich, dass die Annotation aus einer Konjunktion mehrerer Klauseln besteht. Dies ist immer dann der Fall, wenn wir im Wissen des Startknotens mehrere stabile Situationen finden, die dann jeweils als ein Konjunktionsglied in die Annotation eingehen (vergleiche Def. 2.16). Jede Klausel für sich betrachtet, könnte dann mit den entsprechenden Ereignissen einen Diamanten aufspannen. Die Ereignisse, die in den Klauseln angegeben sind, können sich überschneiden – die Sendeereignisse in jeder Klausel sind sogar gleich – sodass sich auch die Diamanten überlappen können.

Pattern. In Abbildung 4.9 zeigen wir das Muster für einen *Diamanten einer Klausel*. Es ähnelt dem zuvor vorgestellten Muster eines Diamanten mit herausführenden Kanten. Jedoch werden hier im ersten Knoten des Diamanten nicht alle Ereignisse verfolgt, sondern nur die, welche in der betrachteten Klausel vorkommen.

Definition 4.13 (Pattern: Diamant einer Klausel)

Sei $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ die Bedienungsanleitung eines Serviceauto-

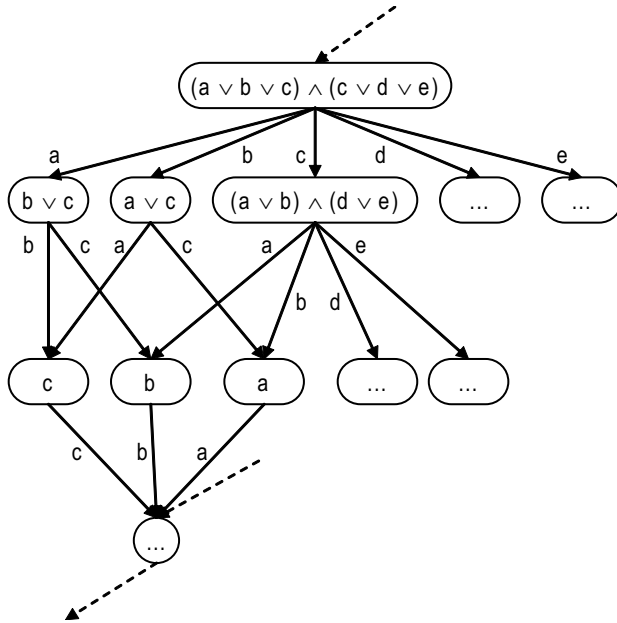


Abbildung 4.9: Das Pattern zeigt einen Diamanten, der durch die Ereignisse a, b, c aufgespannt wird. Der erste Knoten des Diamanten ist der mit $(a \vee b \vee c) \wedge (c \vee d \vee e)$ annotierte Knoten.

Der Diamant enthält Knoten mit herausführenden Kanten (nach Ereignis c). Die Ursache hier liegt aber in der Tatsache, dass das Ereignis c auch in der zweiten Klausel $(c \vee d \vee e)$ möglich ist.

maten. Dann heißt der Subautomat $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein Diamant einer Klausel in OG bezüglich ein Klausel ψ der Annotation von $\phi(q_{start})$, wenn

- (B1) D ist ein maximaler Diamant in OG,
- (B2) für ein Ereignis $(q_1, T, q_2) \in \delta_{OG}$ mit $q_1 \in Q_{OG} \setminus Q_D$ und $q_2 \in Q_D$ folgt, dass $q_2 = q_{start}$ oder $q_2 = q_{end}$,
- (B3**) für ein Ereignis $(q_1, T, q_2) \in \delta_{OG}$ mit $q_1 \in Q_D$ und $q_2 \in Q_{OG} \setminus Q_D$ folgt, dass $q_1 = q_{end}$, oder es existiert eine weitere Klausel ψ' in der Annotation $\phi(q_{start})$ und q_1 ist erreichbar von q_{start} wegen ψ' , und
- (B4**) für ein Ereignis $(q_1, [x], q_2) \in \delta_D$ mit $q_2 \neq q_{end}$ folgt, dass $\phi(q_1) = \phi(q_2) \vee x$, wenn wir in ϕ nur den Teil der Klausel ψ betrachten.

┘

Wenn es mehrere Klauseln in der Annotation eines Knotens q_{start} gibt, ist es möglich, dass die Klauseln teilweise aus gleichen Literalen bestehen. So wie die Annotation für Bedienungsanleitungen definiert ist (Def. 2.16), besteht zum Beispiel jede Klausel aus genau den gleichen Literalen für die entsprechenden Sendeereignisse, da diese unabhängig von der stabilen Situation zu jeder Klausel hinzugefügt werden. Wenn nun

aber ein Knoten erreicht werden kann, weil zwei verschiedene Klauseln die gleichen Ereignisse beinhalten, dann könnten die Kanten, die nicht zum Diamanten gehören, nicht einfach hineinführende oder herausführende Kanten wie zuvor, sondern Kanten die zu dieser zweiten Klausel gehören, sein. Dementsprechend haben wir die Bedingung (B3) angepasst.

Wie beim einfachen Diamanten fordern wir hier erstmal, dass es keine hinein- oder herausführenden Kanten gibt. Später können wir das Pattern analog zum einfachen Diamanten um diese Fälle erweitern.

Ähnlich wie beim Pattern für den Diamanten mit herausführenden Kanten, mussten wir für dieses Muster die Bedingung (B4) anpassen. Anstelle der gesamten Annotation wollen wir einen Teil betrachten. Dieser Teil soll genau der Klausel ψ entsprechen, für die wir in q_{start} überprüfen, ob die enthaltenen Ereignisse einen Diamanten aufspannen.

Ersetzung. Die Kanten, die aufgrund einer zweiten Klausel ψ' , von oder zu Knoten im Diamanten führen, zeigen uns, welche Knoten erhalten bleiben müssen, da sie über diese zweite Klausel erreichbar sind. Wir brauchen zu diesen Zuständen jedoch keine Kanten einzufügen, da alle Kanten, über die wir diese Knoten erreichen können, aufgrund von ψ' erhalten bleiben. Eine Kante wird bei der Ersetzung entfernt, wenn einer der beiden Knoten, die sie verbindet, entfernt wird. Da aber alle Knoten, die durch ψ' erreicht werden können, erhalten bleiben, gibt es auch die Kanten dazwischen weiterhin.

Definition 4.14 (Ersetzung Diamant einer Klausel)

Sei A ein Serviceautomat, $OG = (Q_{OG}, I, O, \delta_{OG}, q_{0_{OG}}, F_{OG})$ dessen Bedienungsanleitung und $D = (Q_D, I, O, \delta_D, q_{start}, F_D)$ ein Diamant einer Klausel in OG , wobei $Q_E \subset Q_D \setminus \{q_{start}, q_{end}\}$ die Zustände von D sind, die aufgrund einer zweiten Klausel in der Annotation von D erreicht werden können. Dann führt die Ersetzung des Diamanten einer Klausel D zu $OG' = (Q_{OG'}, I, O, \delta_{OG'}, q_{0_{OG}}, F_{OG})$ mit

- $Q_{OG'} = (Q_{OG} \setminus Q_D) \cup \{q_{start}, q_{end}\} \cup Q_E$ für q_{start} ist der erste Knoten und q_{end} ist der letzte Knoten des Diamanten und
- $\delta_{OG'} = (\delta_{OG} \cap (Q_{OG'} \times bags(C) \times Q_{OG'})) \cup \{(q_{start}, T_D, q_{end})\}$ für T_D ist die Nachrichtenmenge von D .

┘

Die folgenden beiden Abbildungen 4.10 und 4.11 zeigen jeweils die Ersetzung des Pattern eines Diamanten einer Klausel; einmal für den Fall, dass nur eine Klausel einen Diamanten hervorruft, und einmal für den Fall, dass alle Klauseln Diamanten verursachen.

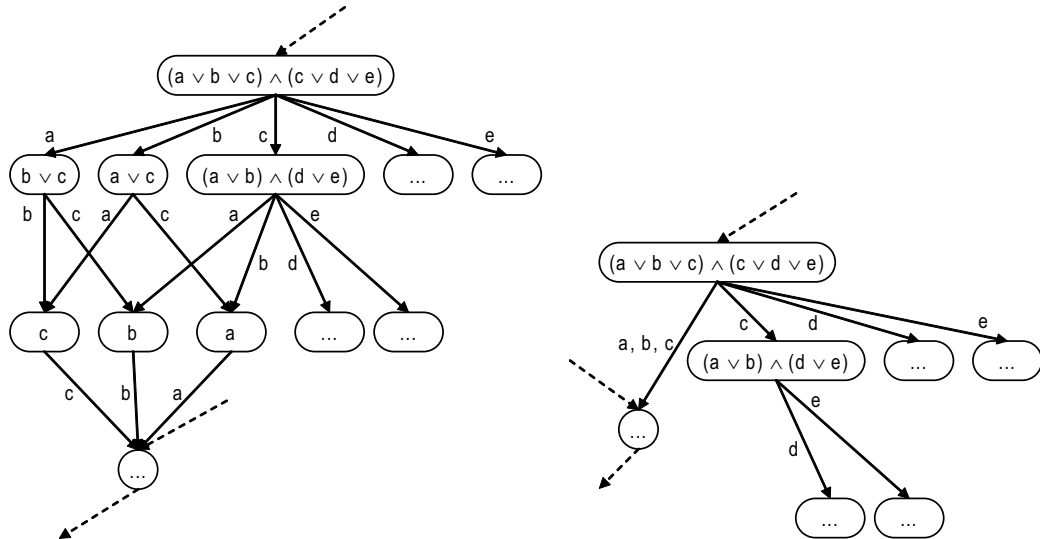


Abbildung 4.10: Links sehen wir das Muster für einen *Diamanten einer Klausel* für den Fall, dass es eine Klausel gibt ($c \vee d \vee e$), die keinen Diamanten aufspannt; rechts dessen Ersetzung.

In der ersten Abbildung (4.10) sehen wir, dass wie im Fall eines Diamanten mit herausführenden Kanten, die Annotation für Knoten, die wir nicht entfernen, erhalten bleiben. Der Grund ist der gleiche wie zuvor: Die entsprechenden Kanten, die den Literalen entsprechen, werden bei der Entfaltung wiederhergestellt, und die Annotation soll das Matching für die Strategien bewahren.

Korrektheit. Eine Strategie eines Serviceautomat muss nun auch mit der Ersetzung des Diamanten matchen.

Satz 4.15 (Korrektheit der Ersetzung)

Sei A ein Serviceautomat, OG_A dessen Bedienungsanleitung mit einem Diamanten einer Klausel D und B ein zu A interfacekompatibler Serviceautomat. Sei OG'_A der SCE, in dem D ersetzt wurde. Dann matcht B genau dann, mit OG'_A , wenn es mit OG_A matcht. ┘

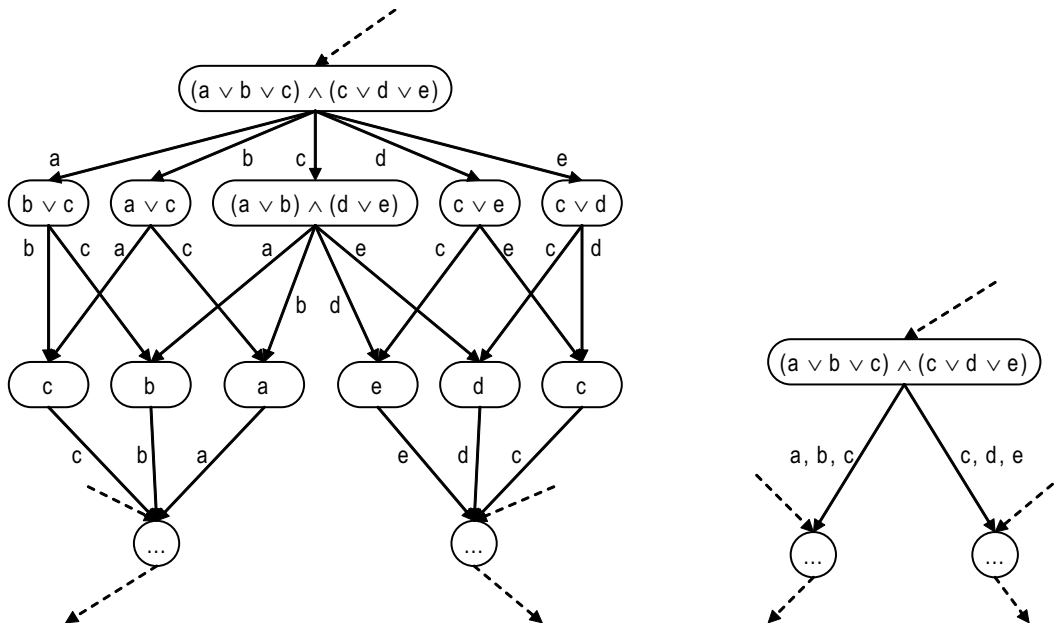


Abbildung 4.11: Links sehen wir das Muster für einen *Diamanten einer Klausel* für den Fall, dass von allen Klauseln aus ein Diamant aufgespannt wird; rechts dessen Ersetzung.

Beweis

Eine Strategie B muss im Knoten q_{start} eine Übergang für jede Klausel der Annotation von q_{start} besitzen. Sei ψ die Klausel, deren Ereignisse den Diamanten D aufgespannt haben, den wir ersetzt haben. Dann ist wie bisher jedes Matching bezüglich ψ genauso möglich wie bei den bisherigen Pattern. Ist ψ' eine weitere Klausel, dann existieren alle Nachfolger von q_{start} , die aufgrund dieser Klausel erreichbar sind, mit den entsprechenden Annotationen. Für Ereignisse x, y, \dots die aufgrund mehrerer Klauseln möglich sind, entsteht bei der Entfaltung wieder genau ein Knoten, zu dem eine entsprechende Reihenfolge von x, y, \dots führt. Gemäß Definition der Entfaltung (Def. 3.2) lauten die Annotationen dieser Zustände so, dass Ereignisse die in q_{start} in unterschiedlichen Klauseln liegen, auch in allen Nachfolgern in unterschiedlichen Klauseln liegen. Somit ist hier das Matching in OG'_A genauso wie in OG_A . \square

Implementation. Die algorithmische Umsetzung wollen wir hier nur als Idee angeben. Sie unterscheidet sich nur gering von den bisher vorgestellten Algorithmen. Der Änderung betrifft das Betrachten der Annotation für q_{start} . Wir betrachten nun

nicht mehr die komplette Annotation, sondern lediglich eine bestimmte Klausel. Entsprechend müssen wir Nachfolger von Knoten, die nicht zum Diamanten gehören können, danach unterscheiden, ob sie über Ereignisse aus anderen Klauseln der Annotation von q_{start} mit den Knoten des Diamanten verbunden sind.

4.2.4 Pattern mit sich gegenseitig ausschließenden Sendeereignissen

Für das letzte in dieser Arbeit vorgestellte Pattern wollen wir lediglich die Grundlagen erläutern und dann auf die bereits vorhandenen Pattern verweisen.

Motivation. In den bisherigen Mustern haben wir Diamanten betrachtet, die durch alle Ereignisse einer Klausel aufgespannt wurden. Trotzdem wollen wir, wenn möglich beziehungsweise praktikabel, auch Diamanten erkennen, die nur durch einige der in einer Klausel vorhandenen Ereignisse aufgespannt werden. Ein entsprechendes Muster, das entstehen kann, wenn sich Sendeereignisse in einer Klausel gegenseitig ausschließen, wollen wir hier beschreiben.

Pattern und Ersetzung. In Abbildung 4.12 können wir sehen, wie durch eine Klausel zwei verschiedene (sich überlappende) Diamanten aufgespannt werden, die wir mit den bisherigen Mustern nicht beschreiben können. Die entsprechende Ersetzung ist ebenfalls in der Abbildung enthalten.

Dass wir Muster, die nur Teile einer Klausel einbeziehen, bisher nicht betrachtet haben, hat vor allem einen algorithmischen Hintergrund. Denn die Frage, welche Ereignisse einer Annotation einen Diamanten aufspannen, führt im Allgemeinen zu einem Auswahlproblem. Sind n verschiedene Literale in einer Annotation enthalten, gibt es 2^n verschiedene Teilmengen der entsprechenden Ereignisse. Für jede Teilmenge zu prüfen, ob sie einen Diamanten aufspannt, dauert dann exponentiell lang.

Wir können im Vorhinein nur anhand der Annotation des betrachteten Knotens nicht sagen, welche Sendeereignisse sich gegenseitig ausschließen. Das ist auch der Grund, warum wir dieses Muster bisher nicht erkennen konnten – wir haben bisher verlangt, dass, wenn wir einen Diamanten finden, *alle* Ereignisse einer Klausel nebenläufig sind, um das genannte Problem zu umgehen.

Implementation. Wir können jedoch abschätzen, ob sich Sendeereignisse gegenseitig ausschließen. Sei q ein Knoten, dessen Annotation mehrere Sendeereignisse

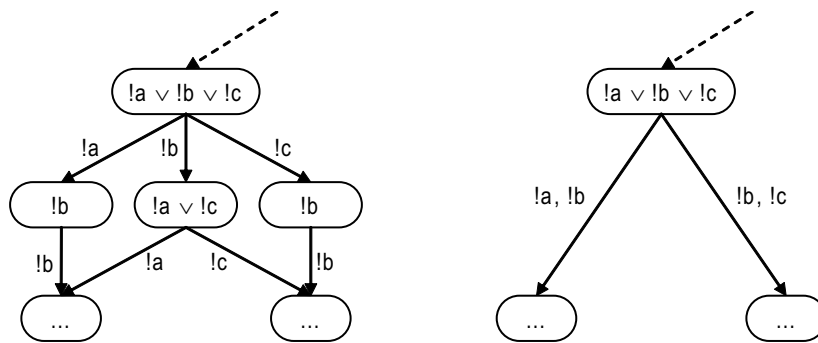


Abbildung 4.12: Auf der linken Seite sehen wir ein Beispiel einer Struktur, in der sich Sendeereignisse gegenseitig ausschließen. Tritt das Ereignis a ein, so kann c nicht mehr eintreten und umgekehrt. Beide Ereignisse können aber nebenläufig zu b stattfinden. Das heißt, es gibt zwei Diamanten, nämlich mit den Ereignissen a, b respektive b, c . Die gezeigte Struktur passt jedoch auf keines der bisher beschriebenen Muster.

Rechts sehen wir eine entsprechende Ersetzung dieser Struktur.

enthält. Dann schauen wir uns alle Nachfolger von q an, die wir über diese Sendeereignisse erreichen, und anhand derer Annotationen sehen wir, ob nun andere Sendeereignisse nicht mehr möglich sind. So können wir die Klausel von q so auf Ereignisse einschränken, die sich nicht gegenseitig ausschließen, und dann ähnlich wie in Abschnitt 4.2.3, wo wir einzelne Klauseln einer Annotation betrachtet haben, die Ereignisse einzelner Teilklauseln dahingehend überprüfen, ob sie einen Diamanten aufspannen.

4.3 Geeignete Wahl des ersten Knotens eines Diamanten

Ein wichtiger Punkt bei der Umsetzung der Algorithmen ist die Wahl von q_{start} . Das wahllose Anwenden der Algorithmen auf alle Zustände einer Bedienungsanleitung ist nicht nur mit einer hohen Zeitkomplexität verbunden, sondern bürgt auch andere Nachteile. So gibt es in jedem einfachen Diamanten D Teilmengen von Knoten, welche die Definitionen für die erweiterten Pattern erfüllen. Zum Beispiel können wir eine Diamanten mit eingehenden Kanten D' erkennen. Jedoch gehen dann in jeden Knoten von D' Kanten hinein. Somit ist die Reduktion sehr gering bezüglich D' , und zudem erkennen wir den einfachen Diamanten D nicht mehr.

Wenn wir die komplette Bedienungsanleitung gegeben haben, können wir vom Startknoten der Bedienungsanleitung ausgehen, und dann jeweils alle Nachfolger betrachten. Für jeden Knoten können wir in Abhängigkeit von der Annotation (Konjunktion von Klauseln, sich ausschließende Sendeereignisse) ein entsprechendes Pattern auswählen und die einzelnen Algorithmen anwenden. Hierbei ist die Gefahr geringer, Diamanten zu finden, die in größeren enthalten sind. Im Allgemeinen sollten wir größere Diamanten immer zuerst finden, da alle in ihnen enthaltenden Diamanten in einem Nachfolger des *ersten* Knotens beginnen. Ausschließen können wir diesen Fall jedoch nicht.

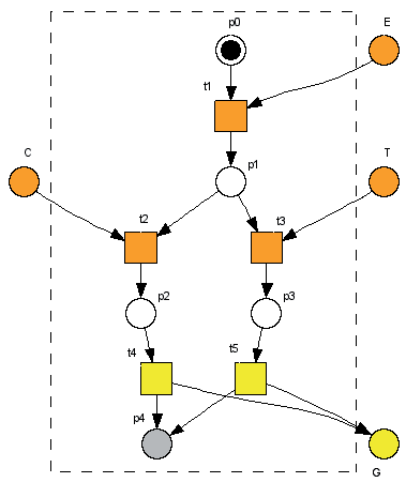
Wenn wir zur Berechnungszeit einer Bedienungsanleitung OG die Algorithmen anwenden wollen, um genügend Speicherplatz für die weitere Berechnung von OG zu haben, wird die Wahl eines geeigneten *ersten* Knotens schwierig. Es sollte dann Kriterien geben, wann es sich lohnt, die Algorithmen auf einen Knoten anzuwenden. Die komplexeren Muster gehen von Knoten q_{start} aus, die entweder aus einer Konjunktion von Klauseln oder aus Sendeereignissen, die sich gegenseitig ausschließen, bestehen. Dies können wir leicht feststellen, und dann die entsprechenden Algorithmen anwenden. Ansonsten können wir ausschließen, dass ein Knoten q Teil eines (einfachen) Diamanten ist, wenn er die Bedingung an die Annotation eines Knoten in einem Diamanten nicht erfüllt. Wenn es einen Vorgänger q' von q gibt, (q', x, q) die entsprechende Kante ist, über die q erreicht wird, und sich die Annotationen von q und q' nicht nur um x unterscheiden, also $\phi(q') \neq \phi(q) \vee x$ (siehe Def. 3.6), dann ist q zumindest kein Teil eines einfachen Diamanten. Hier müssen Fallstudien zeigen, welche Heuristiken eine Reduktion zur Berechnungszeit einer Bedienungsanleitung unterstützen.

4.4 Beispiele

Zum Abschluss dieses Kapitel wollen wir noch einige wenige Beispiele vorstellen, welche die Möglichkeit zur Reduktion von Bedienungsanleitungen zeigen sollen.

Beispiel1 In Kapitel 2 haben wir als einführendes Beispiel das oWFN eines Getränkeautomaten gezeigt, der in Abbildung 4.13 nochmal dargestellt ist.

Die Bedienungsanleitung des Getränkeautomaten ohne und mit Ersetzung sind in Abbildung 4.14 zu sehen. Da das Beispiel des Getränkeautomaten nur eine kleine Bedienungsanleitung mit wenig nebenläufigen Ereignissen hat, fällt die Reduktion



Petri net generated from kaffee.owfn | Final Marking: p4: 1

Abbildung 4.13: Das Bild zeigt den Getränkeautomaten aus Abbildung 2.1, wie es automatisch mit dem Werkzeug Fiona (siehe [LMSW06]) für ein oWFN generiert werden kann.

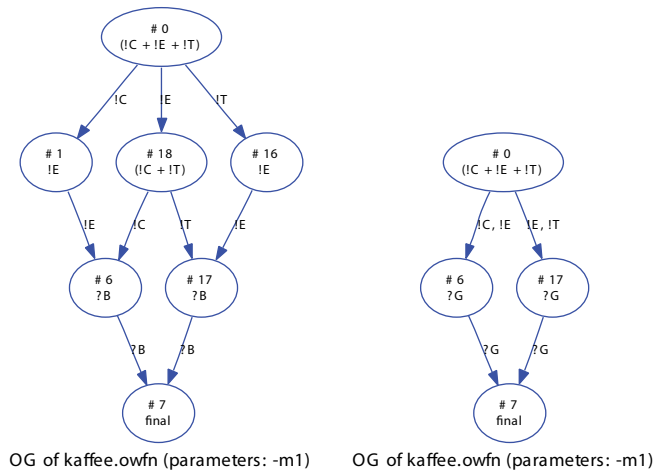
Zur Erinnerung: Der Getränkeautomat erwartet Geld (E, Euro) und die Wahl für Kaffee (C) oder Tee (T) und gibt dann das Getränk (G) aus.

Wie wir an diesem Bild sehen können, ist es egal, ob wir zuerst das Geld einwerfen (E markieren), oder die Wahl für Tee oder Kaffee treffen (T resp. C markieren). Schalten kann G aber erst, wenn eine Marke auf E liegt.

Die Marke von G können wir erst entfernen, nachdem wir E und T beziehungsweise E und C markiert haben. Das Ereignis G tritt immer erst nach den anderen genannten Ereignissen ein.

Entsprechend sieht die Bedienungsanleitung von G aus.

durch die Ersetzung gering aus. Wir können aber immerhin drei Zustände entfernen.



OG of kaffee.owfn (parameters: -m1) OG of kaffee.owfn (parameters: -m1)

Abbildung 4.14: Beispiel: Bedienungsanleitung des Getränkeautomaten

Die beiden Bilder in Abbildung 4.14 zeigen jeweils die Bedienungsanleitung des Getränkeautomaten G . Links sehen wir die klassische Form, in der jede Kante mit genau einem Ereignis beschriftet ist. Rechts sehen wir die Variante, in der die Diamanten durch Kanten mit mehrelementigen Multimengen ersetzt wurden.

Beispiel2 Das oWFN aus Abbildung 4.15 enthält zwar parallele, unabhängige Transitionen, jedoch führen diese nicht zu Diamantenstrukturen in der Bedienungsanleitung des oWFN, deren Ersetzung zwangsläufig einen Speicherplatzgewinn bringt.

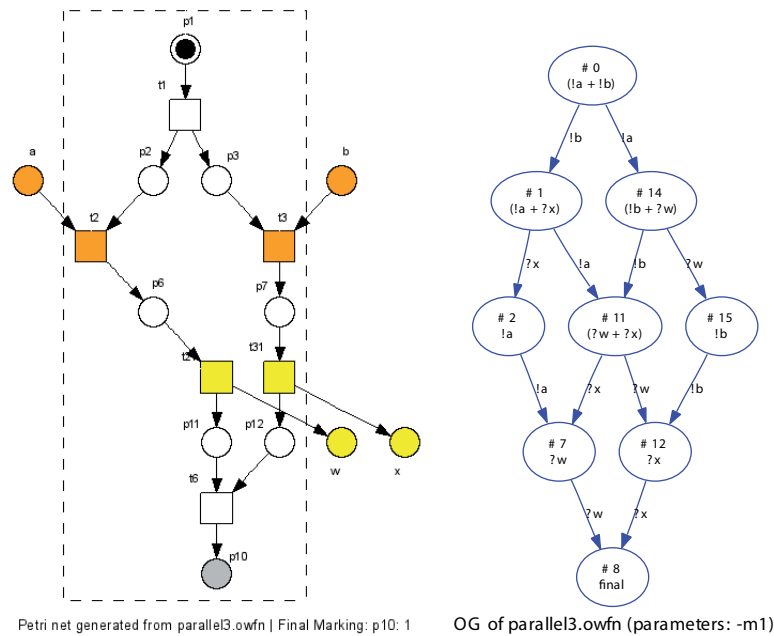


Abbildung 4.15: Das gezeigte oWFN (links) besteht aus zwei parallele Komponenten. Nach der Initialisierung durch die erste Transition, ist das Empfangen von a und Senden von w vollkommen unabhängig vom Empfangen von b und Senden von x . Entsprechend besitzt die Bedienungsanleitung (rechts) vier verschiedene Diamanten (a,b ; a,x ; b,w ; w,x).

In Abbildung 4.16 sehen wir die Bedienungsanleitung mit einer jeweils unterschiedlichen Reihenfolge der Ersetzungen der Diamanten. In einem Fall sparen wir keine Zustände ein, im anderen immerhin zwei.

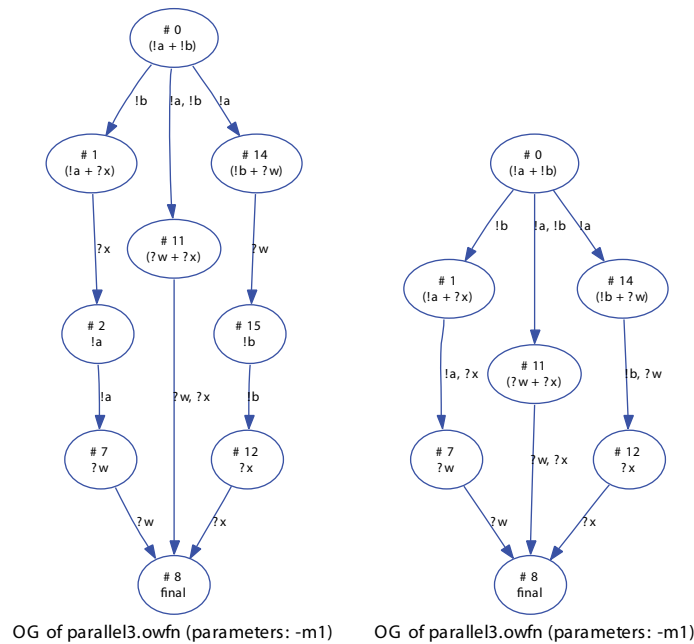


Abbildung 4.16: Beispiel: Bedienungsanleitung Parallele Transitionen mit Ersetzung

Links in Abbildung 4.16 sehen wir das Ergebnis der Reduktion, wenn wir zuerst die Diamanten a, b und w, x ersetzen. Dann sind außer diesen Ersetzungen keine weiteren möglich und wir haben keine Speicherplatzersparnis. Ersetzen wir jedoch zuerst die Diamanten a, x und b, w , sind die anderen beiden Ersetzungen a, b und w, x zusätzlich möglich, und wir erhalten das Resultat, das rechts in der Abbildung zu sehen ist. Das zeigt deutlich, dass die Reihenfolge, in der die Knoten betrachtet werden, und in der die Muster angewandt werden, von großer Bedeutung für die Reduktion ist.

Beispiel3 Das dritte Beispiel in Abbildung 4.17 ist ein einfaches Modell eines Onlineshops. Ein Nutzer des Shops hat die Möglichkeit, sich mit seiner Kennung beim Shop anzumelden, und Bestellungen aufzugeben. Ist der Nutzer bereits beim Shop registriert, erhält er auf die Bestellung die Rechnung und die bestellte Ware. Ist er noch nicht registriert, muss er vor Erhalt von Rechnung und Ware noch die Geschäftsbedingungen des Onlineshops bestätigen.

4 Reduktion der Bedienungsanleitung

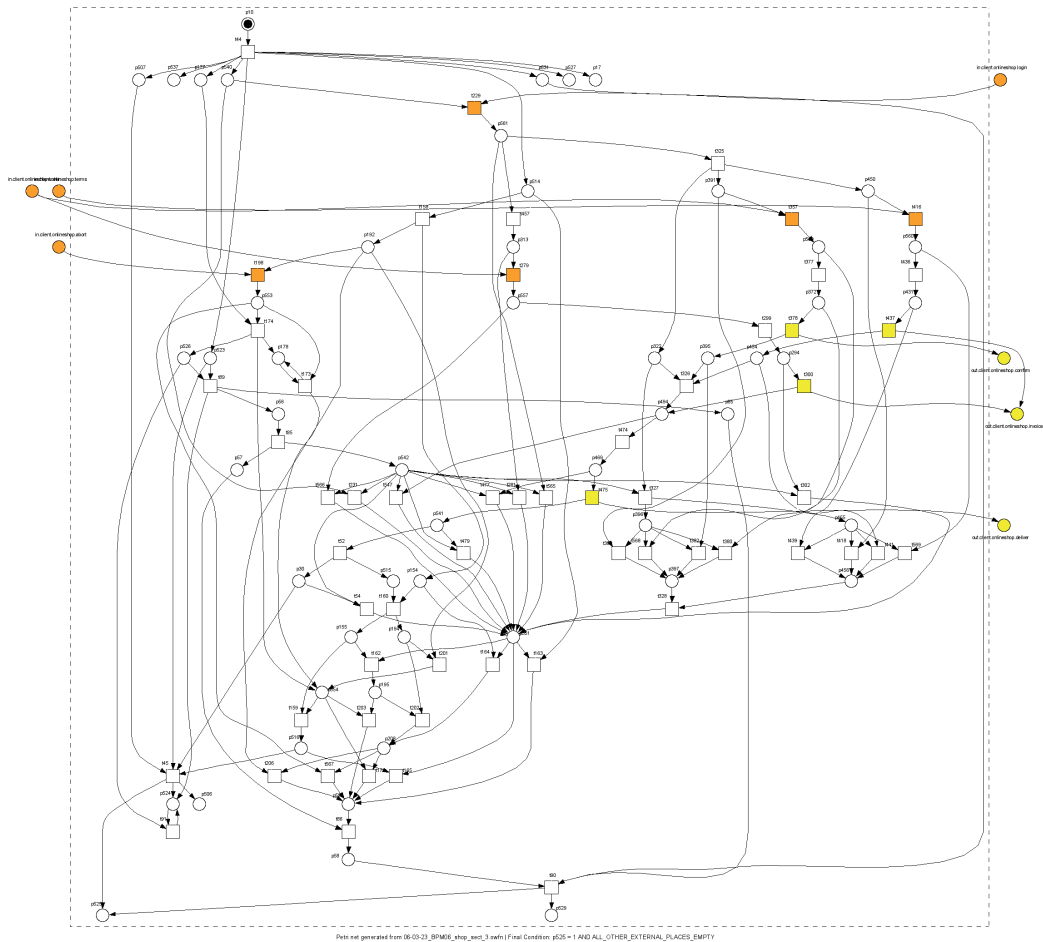


Abbildung 4.17: Dieses Beispiel ist aufgrund seiner internen Struktur bereits so komplex, dass wir es hier kaum lesbar darstellen können. Schemenhaft können wir die Eingabe- und Ausgabeplätze erkennen.

Trotz der Größe des oWFN in Abbildung 4.17 entsteht eine relativ kleine Bedienungsanleitung für den Onlineshop, die wir zusätzlich noch reduzieren können.

Die Bedienungsanleitung und ihre Reduktion ist in Abbildung 4.18 zu sehen.

In der Bedienungsanleitung können wir drei verschiedene Diamanten finden. Zum einen bilden die Ereignisse !order und !login (oben) einen Diamanten. Diese beiden

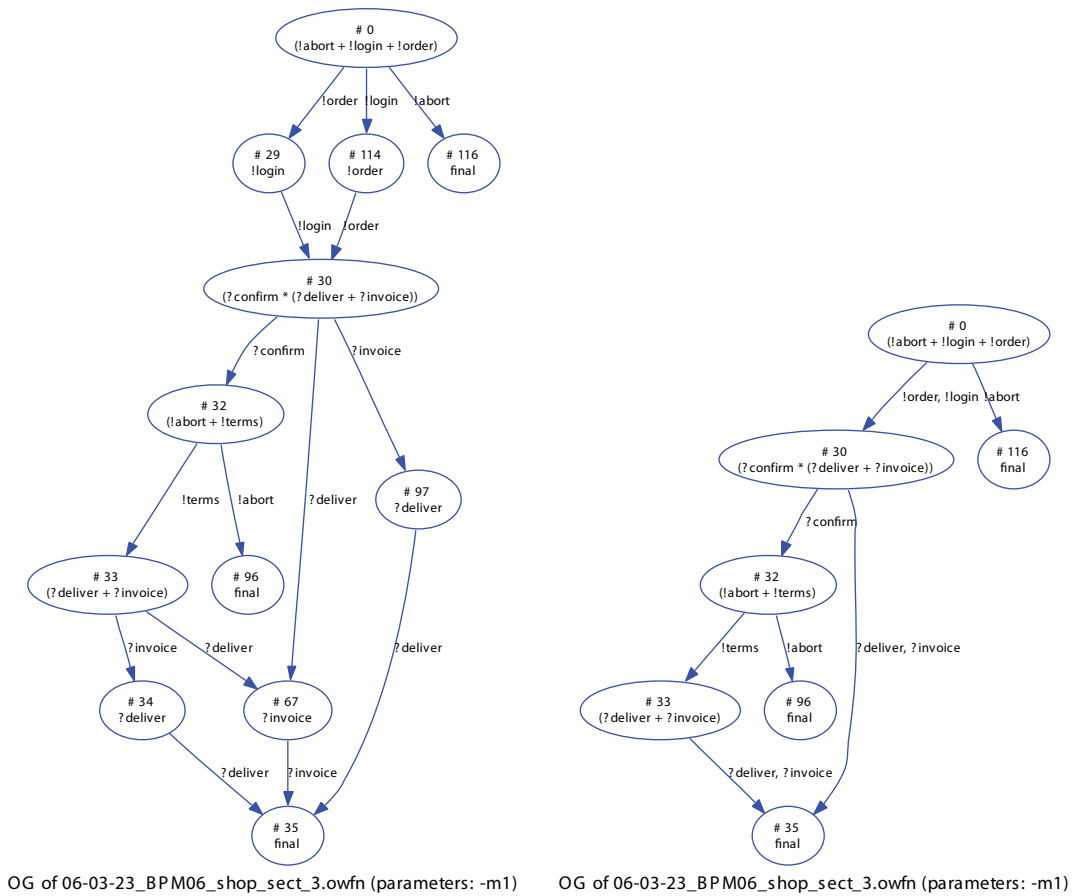


Abbildung 4.18: Beispiel: Bedienungsanleitung Onlineshop

Ereignisse schließen das Ereignis !abort aus. Damit haben wir hier ein Beispiel für die Anwendung des Musters für Diamanten mit sich gegenseitig ausschließenden Sendereignissen.

Im unteren Teil der Bedienungsanleitung sehen wir zwei Diamanten, die jeweils durch die Ereignisse ?invoice und ?deliver entstehen.

Die Beispiele zeigen, dass die Erkennung der Muster und eine entsprechende Ersetzung durch eine Kurzschreibweise eine Reduktion von Bedienungsanleitung zur Folge haben kann. Wir haben aber auch gesehen, dass die Reihenfolge, in der potentielle

4 Reduktion der Bedienungsanleitung

erste Knoten von Diamanten betrachtet werden, Einfluss darauf haben kann, wie stark die Reduktion ausfällt.

5 Fazit

5.1 Zusammenfassung

Diese Arbeit hatte zum Ziel, Bedienungsanleitungen strukturell zu reduzieren. Für einen gegebenen Serviceautomaten ist die Bedienungsanleitungen mitunter exponentiell groß. Als eine der Ursachen dafür haben wir Nebenläufigkeit von Ereignissen identifiziert. In der Modellwelt von Serviceautomaten setzen wir einen asynchronen Nachrichtenaustausch voraus, was die Nebenläufigkeit von Ereignissen fördert.

Um Nebenläufigkeit in Serviceautomaten ausdrücken zu können, haben wir eine Kurzschreibweise eingeführt. In einem Serviceautomaten ist es damit nicht nur möglich, Übergänge, die nur aus einem Ereignis x bestehen, sondern vor allem Übergänge, die mit einer Multimenge T von Ereignissen beschriftet sind, anzugeben. Bei der Beschreibung der Entfaltung solcher Übergänge haben wir festgelegt, dass ein Übergang mit T bedeutet, dass die Ereignisse in T bei dem Übergang in jeder beliebigen Reihenfolge stattfinden können.

Im Hauptteil der Arbeit haben wir den Schritt der Entfaltung umgedreht. Da die Entfaltung eines Übergangs mit einer Multimenge einer Diamantenstruktur entspricht, haben wir Diamantenmuster in Bedienungsanleitungen charakterisiert, die wir durch eine Kurzschreibweise ersetzen können. Die definierte Semantik der Kurzschreibweise stellt dabei sicher, dass der Sinn einer Bedienungsanleitung – die Charakterisierung aller sinnvoll interagierenden Partner eines Serviceautomaten – nicht verloren geht.

Folgende Muster und Ersetzungen haben wir betrachtet:

- **Einfacher Diamant:** Eine Diamantenstruktur, die weitestgehend losgelöst von der restlichen Bedienungsanleitung ist.
- **Diamant mit hinein führenden Kanten:** Eine Erweiterung des einfachen Diamanten, die Kanten erlaubt, die von Knoten außerhalb des Diamanten in den Diamanten führen.
- **Diamant mit heraus führenden Kanten:** Auch eine Erweiterung des einfachen Diamanten, nur das hier Kanten zu Knoten außerhalb des Diamanten heraus führen dürfen.

- **Diamant einer Klausel:** Für Knoten, deren Annotation aus einer nicht trivialen Konjunktion von Klauseln besteht, haben wir jede Klausel einzeln auf die Existenz einer Diamantenstruktur untersucht.
- **Diamant mit sich ausschließenden Sendeereignissen:** Können wir feststellen, dass sich Sendeereignisse in der Annotation eines Knotens ausschließen, können wir für Teilklauseln das Vorhandensein einer Diamantenstruktur nachprüfen.

Für das Muster des einfachen Diamanten haben wir einen Algorithmus angegeben, der die Kriterien, die für einen einfachen Diamanten gelten müssen, überprüft, und im Falle der Existenz eines einfachen Diamanten, eine Menge von Knoten zurück gibt, die diesen Diamanten genau beschreibt.

Im weiteren Verlauf der Arbeit haben wir diesen Algorithmus dahin gehend erweitert, dass wir mit ihm die erweiterten Muster erkennen können.

Für die einzelnen Algorithmen haben wir ihre Korrektheit gezeigt.

5.2 Ausblick

5.2.1 Weitere Ansatzpunkte zur Reduktionen

Diamantenstrukturen stellen ein zentrales Problem für die Größe von Bedienungsanleitungen dar. Sie müssen jedoch nicht zwangsläufig in dieser Form auftreten. So kann es sein, dass eine Menge von Ereignissen in jeder beliebigen Reihenfolge eintreten kann, aber kein gemeinsamer letzter Knoten erreicht wird. Trotzdem kann die Struktur nach jedem *letzten* Knoten einer der Reihenfolgen gleich sein. Dies sollten wir erkennen können, denn die unterschiedlichen Reihenfolgen können wir dann durch die Kurzschreibweise ersetzen. Das Auffinden gleicher Teilstrukturen ist ein graphentheoretisches Problem und sollte unbedingt in zukünftigen Arbeiten verfolgt werden, da sich hier viele Ansätze zur Reduktion bieten.

In Abbildung 5.1 sehen wir eine einem Diamanten ähnelnde Struktur. Die einzelnen Folgen von Ereignissen führen jedoch nicht zu genau einem *letzten* Knoten. Unter Umständen können wir die Struktur trotzdem durch eine Kurzschreibweise ersetzen.

Neben der reinen Struktur können wir auf weitere Informationen der Bedienungsanleitung zurück greifen. Für jeden Knoten steht uns das *Wissen* zur Verfügung, das

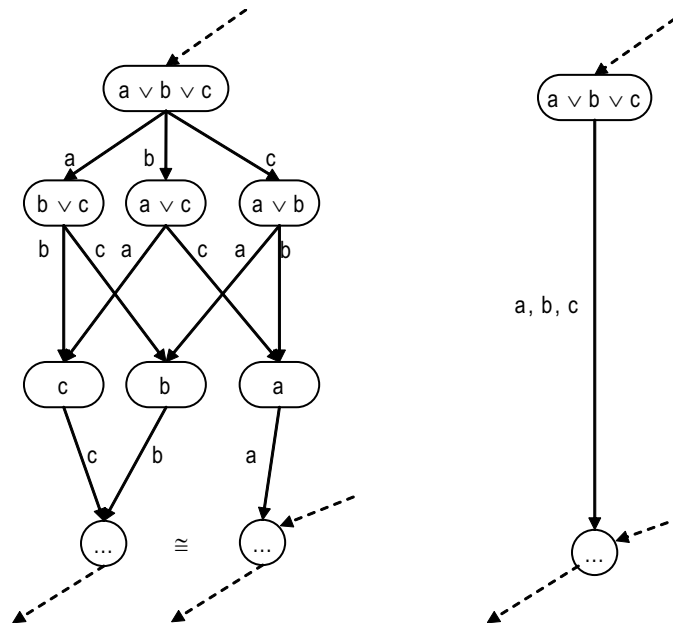


Abbildung 5.1: Links sehen wir den Teil eines Automaten, der in seiner Struktur stark an einen *einfachen Diamanten* erinnert. Jedoch führen nicht alle Folgen zum gleichen Knoten. Wenn wir annehmen, dass die Bedienungsanleitung in den beiden unteren Knoten isomorph in der Struktur und den Annotation ist, dann sind die beiden Knoten nicht unterscheidbar (in keinem Punkt, der uns interessiert). Dann sollte es möglich sein, die Ersetzung rechts vorzunehmen.

wir in dieser Arbeit kaum genutzt haben. Lediglich für den Fall, dass ein einfacher Diamant nur aus Empfangsereignissen besteht, haben wir auf die stabile Situation des ersten Knotens geschaut, und konnten damit einen effizienteren Algorithmus zur Erkennung des Musters angeben. Aus dem Wissen könnten wir jedoch auch für weitere Muster Informationen erhalten, ob sie in einer Bedienungsanleitung vorhanden sind, oder in einem bestimmten Knoten beginnen. Hier lohnte sich die Charakterisierung der Muster mit Hilfe des Wissens, das in den Knoten eines Diamanten vorhanden ist.

Ein weiterer wichtiger Punkt ist die Reduktion einer Bedienungsanleitung zur Berechnungszeit. Damit wir eine Bedienungsanleitung nutzen können, müssen wir sie berechnen können. Es ist aber möglich, dass während der Konstruktion mehr Speicher benötigt wird, als das System, auf dem die Bedienungsanleitung berechnet wird, zur Verfügung stellt. Jedoch wird die Bedienungsanleitung nach und nach aufgebaut. Können wir in einem bereits berechneten Teil ein Diamantenmuster finden, lohnt es

sich, dieses sofort zu ersetzen. Der dadurch frei gewordene Speicherplatz steht dann für die weitere Berechnung der Bedienungsanleitung zur Verfügung.

6 Danksagung

Allen voran möchte ich mich bei Peter Massuthe bedanken, der mich hartnäckig bei meiner Arbeit voran getrieben hat, und mit mir viele hilfreiche Gespräche geführt hat. Insgesamt habe ich die Atmosphäre an Prof. Reisigs Lehrstuhl sehr genossen, und habe mich fachlich als auch menschlich gut aufgehoben gefühlt. Weiterhin weiß ich das Feedback von Karsten Wolf sehr zu schätzen, der von Rostock aus die Arbeit betreut hat.

Ein besonderer Dank gilt meinen Eltern und meiner Freundin Ela, die mir Rückhalt und Unterstützung geboten haben.

Abbildungsverzeichnis

1.1	SOA-Prinzip	6
2.1	Getränkeautomat als oWFN	11
2.2	Getränkeautomat als Serviceautomat	13
2.3	Komponierter Serviceautomat	15
2.4	Wissensfunktion	17
2.5	Bedienungsanleitung des Getränkeautomaten	22
3.1	Serviceautomat mit nebenläufigen Ereignissen	26
3.2	Entfalteter SCE	28
3.3	Diamantenstruktur	32
4.1	Einfacher Diamant	34
4.2	Ersetzung einfacher Diamant	36
4.3	Diamant, in dem das Ereignis b zweimal auftritt	37
4.4	Diamant mit Zyklus	38
4.5	Diamant mit hineinführenden Kanten	46
4.6	Ersetzung Diamant mit hineinführenden Kanten	48
4.7	Diamant mit herausführenden Kanten	51
4.8	Ersetzung Diamant mit herausführenden Kanten	53
4.9	Konjunktion von Ereignissen	56
4.10	Ersetzung Diamant einer Klausel (1)	58
4.11	Ersetzung Diamant einer Klausel (2)	59
4.12	Sich ausschließende Sendeereignisse	61
4.13	Beispiel: Getränkeautomat als oWFN	63
4.14	Beispiel: Bedienungsanleitung des Getränkeautomaten	63
4.15	Beispiel: Parallele Transitionen als oWFN und Bedienungsanleitung	64
4.16	Beispiel: Bedienungsanleitung Parallele Transitionen mit Ersetzung	65
4.17	Beispiel: Onlineshop als oWFN	66
4.18	Beispiel: Bedienungsanleitung Onlineshop	67
5.1	Offener Diamant	71

Literaturverzeichnis

- [CGP01] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 2001.
- [Ker07] Andreas Kerlin. Bedienbarkeit unter Kausalität. Diplomarbeit, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Jan 2007.
- [LMSW06] Niels Lohmann, Peter Massuthe, Christian Stahl, and Daniela Weinberg. Analyzing interacting BPEL processes. In Shahram Dustdar, José Luiz Fiadeiro, and Amit Sheth, editors, *Business Process Management, 4th International Conference, BPM 2006, Vienna, Austria, September 5-7, 2006, Proceedings*, volume 4102 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, September 2006.
- [LMW07] Niels Lohmann, Peter Massuthe, and Karsten Wolf. Operating guidelines for finite-state services. In Jetty Kleijn and Alex Yakovlev, editors, *28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007, Siedlce, Poland, June 25-29, 2007, Proceedings*, volume 4546 of *Lecture Notes in Computer Science*, pages 321–341. Springer-Verlag, 2007.
- [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.
- [MS05] Peter Massuthe and 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)*, pages 452–457, Melbourne, Australia, September 2005. IEEE Computer Society.
- [Rei82] Wolfgang Reisig. *Petrinetze, Eine Einführung*. Springer, 1982.
- [Sch05] Karsten Schmidt. Controllability of open workflow nets. In Jörg Desel and Ulrich Frank, editors, *Enterprise Modelling and Information Systems*

Architectures, volume P-75 of *Lecture Notes in Informatics (LNI)*, pages 236–249, Bonn, 2005. Entwicklungsmethoden für Informationssysteme und deren Anwendung (EMISA, RWTH Aachen), Köllen Druck+Verlag GmbH.

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit „*Strukturelle Reduktion von Bedienungsanleitungen*“ selbstständig und ohne fremde Hilfe verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet zu haben.

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

Berlin, den 25. Januar 2008

