4. Grundlegende Modelle der Informatik

4.1 Einleitung

Die Beschreibung von Phänomen der Informatik und deren Zusammenhängen basiert wie in den klassischen Naturwissenschaften Mathematik und Physik auf präzisen mathematischen Ausdrucksweisen und geeigneten Modellen sowie einer geeigneten Abstraktion dieser Modelle. Die Informatik ist geprägt durch die Entwicklung und Anwendung von Computern, die komplexe Prozesse und Abläufe in Form von Programmen ausführen, dabei mit anderen Prozessen kommunizieren und diese dabei steuern oder regeln. Das wesentliche Element eines Computers ist sein Gedächtnis, in dem entscheidende Ereignisse aus der Vergangenheit, die das Fortschreiten bzw. das Verhalten der Prozesse in der Zukunft beeinflussen, gespeichert sind.

Ein geeignetes Mittel um Verhaltensweisen eines Computers bzw. dessen Umgebung zu modellieren ist das Automatenmodell, das wir in diesem Kapitel als eines der zentralen Beschreibungsmodelle in der Informatik betrachten wollen. Das Automatenmodell ist letzendlich die Basis aller formalen Verhaltensbschreibungen bzw. formalen Methoden in der Informatik und insbesondere im Software-Engineering, auch wenn dies nicht immer offensichtlich ist. Im engen Zusammenhang mit endlichen Automaten stehen Begriffe wie regulärer Ausdruck und reguläre Sprache, die wir anschließend behandeln wollen.

Endliche Automaten sind ein nützliches Beschreibungsmodell für viele Arten von Software und Hardware. Compiler, das sind Übersetzer für Programme in Maschinencode, verwenden endliche Automaten zur lexikalischen Analyse und Aufschlüsselung der Eingaben in logische Einheiten wie Bezeicher, Schlüsselwörter und Satzzeichen. Ebenso werden endliche Automaten zur Modellierung von Software zum Durchsuchen umfangreicher Texte, wie beispielsweise Sammlungen von Web-Seiten, um Vorkommen von Wörtern, Ausdrücken oder anderer Muster zu finden, eingesetzt. Ein sehr komplexes Einsatzgebiet von endlichen Automaten ist die formale Spezifikation und Verifikation (Nachweise der Korrektheit) von Softwaresystemem zur Steuerung von industrieellen Prozessen oder Protokollen bei eingebetteten Systemen oder im Kommunikationsbereich.

Um uns vorsichtig und sehr praktisch orientiert dem Automatenmodell sowie dessen formaler Einführung und Definition zu nähern, beginnen wir im nächsten Abschnitt mit der Betrachtung eines jederman wohlbekannten Getränkeautomaten. Unter dem Automaten verstehen wir dabei ausschließlich die Arbeitsweise des betrachteten Systems oder genauer gesagt das von außen beobachtbare Systemverhalten. Die physikalische bzw. technische Realisierung des Systems spielt in dieser Betrachtungsweise keine Rolle. Ein durch ein Automatenmodell beschriebenes Verhalten kann durch viele völlig unterschiedliche Hardware- und/oder Software-Implementierungen realisiert werden.

Die Beschreibung eines Verhaltens wird im Automatenmodell durch Zustände und Übergänge zwischen diesen Zuständen ausgedrückt. Damit lassen sich Abläufe, wie sie in einem Automaten stattfinden, sehr gut beschreiben. Jeder Zustand des Automatenmodells repräsentiert eine Situation, in der sich das betrachtete System befinden kann. Über den momentan bestehenden Zustand merkt sich der Automat, das bestimmte Ereignisse vorgefallen sind und andere nicht oder noch nicht. Zustandsänderungen finden immer genau dann statt, wenn für das weitere Systemverhalten relevante Ereignisse aufgetreten sind. Dabei handelt es sich in der Regel um Ereignisse, die von der Umgebung des Automaten initiiert bzw. erzeugt werden. Die Beschreibungselemente Zustand und Zustandsübergang (Transition) eines Automaten sind abstrakt schwer zu definieren. Andererseits ist auch deren Bedeutung anschaulich schwer zu erklären. Um leichter ein Gefühl für die Bedeutung der beiden Elemente Zustand und Transition in Bezug auf Automaten zu bekommen, betrachten wir in den ersten Abschnitten dieser KE einige ausführlichere Beispiele von Automatenmodellen.

4.2 Informelle Betrachtung endlicher Automaten

4.2.1 Verhalten eines Automaten und dessen grafische Darstellung

Automatensysteme, die durch elektronische Schaltungen (digitalelektronische Logik oder Mikrorechner und Programme) aufgebaut werden, sind in unserem täglichen Leben heute allgegenwärtig ohne das wir uns dessen ständig bewusst sind. Populäre Beispiele solcher Automaten sind Bankautomaten, Wasch- und Spülmaschinen, Getränke-, Zigaretten-, Fahrkarten und Süßigkeitsautomaten, sowie Computer, Elektronikspielzeuge und Handys. Charakterisiert sind diese Automaten durch die Ausführung einer Aneinanderreihung von Aktionen, die wiederum durch Eingaben und inneren Zustand festgelegt sind und davon abhängige Ausgaben erzeugen. In einem bestimmten Zustand sind definierte Eingaben zulässig, die einen entsprechenden Folgezustand und eine mögliche Ausgabe erzeugen.

Die oben genannten Automatenbeispiele besitzen eine Gemeinsamkeit, sie befinden sich wie alle endlichen Automaten zu jedem gegebenen Zeitpunkt in einem aus einer endlichen Anzahl von Zuständen. Zweck dieses Zustandes ist es, den relevanten Teil der Geschichte eines Systems festzuhalten. Da es nur eine endliche Anzahl von Zuständen gibt, kann natürlich im Allgemeinen nicht die gesamte vergangene Geschichte beschrieben werden. Das Automatenmodell muss daher sehr sorgfältig entworfen werden, so dass das Wichtige im Modell beschrieben und damit gespeichert wird und das Unwichtige weggelassen wird. Ganz wesentlich ist dabei die Vorgabe, mit einer endlichen Zahl von Zuständen zu arbeiten. Ein beliebig lang aktives System muss also früher oder später wieder in bestimmte bereits durchlaufene Zustände zurückkehren, eine unendliche Zustandsmenge wäre mit einer begrenzten Menge an verfügbaren Ressourcen auch nicht implementierbar.

Beispiel 4.1 *Getränkeautomat als Beispiel eines Automaten*
Stellen Sie sich einen einfachen Getränkeautomaten vor, der nur eine Getränkeart A ausgeben kann. Das Getränk A koste einen Betrag S, der in Form des Einwurfs von verschiedenen Geldstückskombinationen beglichen werde. Ein Geldstück G1 entspreche dem Betrag S, eine zweite Geldstücksart G2 entspreche dem halben Betrag S und eine dritte Geldstücksart G3 entspreche einem Viertel des Betrages S. Dabei sei ein Einwurf der Geldstücke in beliebiger Reihenfolge möglich. Nachdem mindestens die Summe S eingeworfen wurde, wird unmittelbar das Getränk A ausgegeben und der Verkaufsvorgang ist abgeschlossen. Der Einfachheit halber seien bei diesem fiktiven Automaten eine Geldrückgabe oder andere Aktionen nicht möglich.

Mit jedem Münzeinwurf bewegt sich der Automat vom Ausgangszustand, in dem sich kein Geld im Einwurf befindet, ausgehend jeweils in einen weiteren Zustand. Ist der Automat in einem Endzustand, in dem der geforderte Betrag S eingeworfen wurde, angekommen, dann wird eine Ausgabe erzeugt. □

Aufgabe 4.1 *Mögliche Eingaben eines Getränkeautomaten*
Geben Sie die Folgen von Eingabeaktionen des Getränkeautomaten aus Beispiel 4.1 an, die zu einem erfolgreich abgeschlossenen Verkaufsvorgang führen. ◇

Die möglichen Eingabefolgen eines Automaten können wir in Form eines gerichteten Graphen, der im Zusammenhang mit Automaten als Zustandsgraph oder Zustandsdiagramm bezeichnet wird, veranschaulichen. Die Knoten des Graphen modellieren die nach einer Eingabe auftretenden Zustände des Automaten und die gerichteten Kanten modellieren die aufgrund der Eingabeaktionen auftretenden Übergänge zwischen diesen Zuständen. Dabei besitzt der Automat in jedem Fall einen Anfangs- oder Startzustand, der durch einen auf dem Anfangsknoten endenden Pfeil hervorgehoben ist. Der Automat besitzt möglicherweise einen oder mehrere Endzustände, die durch einen doppelkreisigen Knoten gekennzeichnet werden.

Beispiel 4.2 *Zustandsdiagramm eines Getränkeautomaten*
Zu Beginn befindet sich der Getränkeautomat in einem Ausgangszustand *Start*. Durch den Einwurf

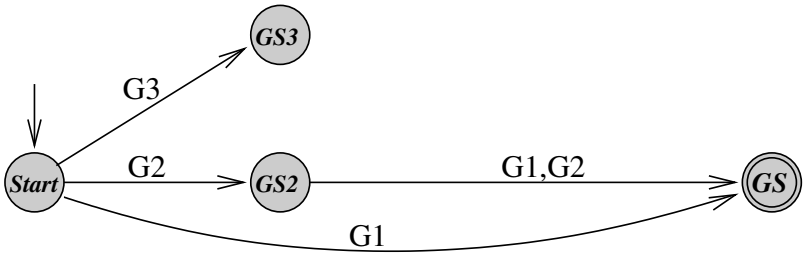


Abbildung 4.1: Vorläufiger Teil des Zustandsdiagramms des Getränkeautomaten

eines Geldstücks G1 geht der Automat in einen Enzustand GS über. Die Kante vom Ausgangszustand *Start* in den Zustand GS haben wir mit dem auslösenden Ereignis, hier der Geldeinwurf, beschriftet.

Durch den Einwurf eines Geldstücks G2 geht der Automat in einen von GS verschiedenen Zustand *GS2* über, um sich damit zu merken, dass ein Teil der Summe, hier die Hälfte, eingegeben wurde. Entsprechend geht der Automat durch den Einwurf eines Geldstücks G3 in einen Zustand *GS3* über. Offensichtlich gibt so viele Möglichkeiten den Ausgangszustand zu verlassen wie es unterschiedliche Einwürfe (Geldstücke) gibt.

Im Zustand GS ist nach Einwurf des Geldstücks G1 und der Ausgabe des Getränks A ein Verkaufsvorgang abgeschlossen. In den Zuständen *GS2* und *GS3* müssen aber noch weitere Geldstücke eingeworfen werden. Von Zustand *GS2* aus kommen wir durch den Einwurf eines weiteren Geldstücks G2 ebenfalls in den Zustand *GS* und der Verkaufsvorgang wird durch die Ausgabe des Getränks A abgeschlossen. Da der betrachtete Automat keine Geldrückgabe durchführt, kommen wir durch den Einwurf eines weiteren Geldstücks G1 ebenfalls in den Zustand *GS* und der Verkaufsvorgang wird abgeschlossen. Durch den Einwurf eines Geldstücks G3 im Zustand *GS2* kommen wir in einen neuen Zustand *GS4*. Es gibt also auch in *GS2* so viele Möglichkeiten den Zustand zu verlassen wie es Geldstücke gibt.

Betrachten wir nun den ebenfalls noch nicht abgeschlossenen Zustand *GS3*. Von Zustand *GS3* aus kommen wir durch den Einwurf eines weiteren Geldstücks G1 in den Zustand *GS* und der Verkaufsvorgang wird abgeschlossen. Durch den Einwurf eines Geldstücks G2 kommen wir von Zustand *GS3* aus in den bereits vorhandenen Zustand *GS4* und durch den Einwurf eines Geldstücks G3 kommen wir in einen neuen Zustand *GS5*.

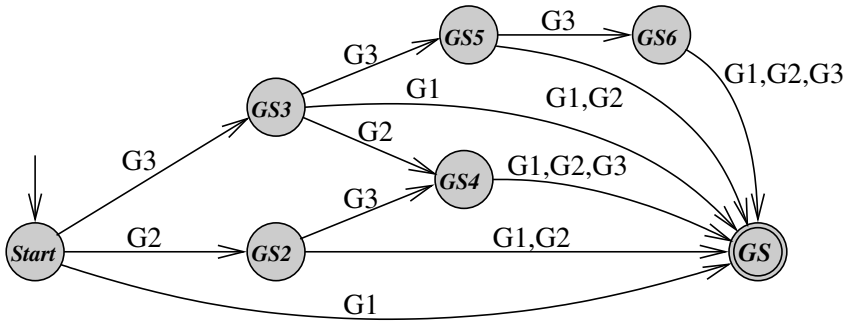


Abbildung 4.2: Vervollständigtes Zustandsdiagramm des Getränkeautomaten

Betrachten wir nun den Zustand *GS4*. Von Zustand *GS4* aus kommen wir durch den Einwurf des nächsten Geldstücks, egal ob G1, G2 oder G3 in den Zustand GS und der Verkaufsvorgang wird abgeschlossen. Von Zustand *GS5* aus kommen wir durch den Einwurf der Geldstücke G1 oder G2 in den Zustand *GS*. Durch den Einwurf von G3 kommen wir in einen weiteren Zustand *GS6*, von dem aus wir aber mit dem nächsten Einwurf in jedem Fall den Zustand *GS* erreichen. □

Im obigen Beispiel der Modellierung eines Getränkeautomaten besitzt der Zustand *GS* den Charakter eines Endzustandes. Mit einem solchen Endzustand ist der Automat in dieser Form natürlich unbrauchbar, da dieser nur genau einen Verkaufsvorgang ausführen könnte und danach im Endzustand

blockiert ist. Ein Automat, der mehrere Verkaufsvorgänge hintereinander ausführen soll, muss sich selbsständig aus diesem Zustand befreien und in den Ausgangszustand zurückkehren um einen neuen Verkaufsvorgang beginnen zu können. Wir wollen das Problem an dieser Stelle nicht weiter verfolgen und auf sehr einfache Weise lösen, indem wir den Abschluss des Verkaufsvorgangs durch den bedingungslosen Übergang zurück in den Ausgangszustand *Start* modellieren. Dabei werde angenommen, dass mit dem Zustandsübergang das Getränk A ausgegeben wird. Wir haben damit aber schon eine wichtige Eigenschaft eines praktischen Automaten identifiziert, ein solcher Verkaufsautomat muss einen Zyklus besitzen, bei dem er nach endlichen vielen Zustandsübergängen in den Ausgangszustand zurückkehrt.

4.2.2 Eigenschaften des Automatenmodells

Wir wollen nun versuchen für ein Automatenmodell wichtige allgemeingültige Elemente, die für die Verhaltensbeschreibung von möglichst grundlegender Bedeutung sind, anhand des bisherigen Betrachtungen zu identifizieren. Solche Merkmale oder Eigenschaften des Automatenmodells müssen natürlich unabhängig sein vom konkret beschriebenen Vorgang, der jeweiligen Anwendung oder der beschriebenen Funktion. Ferner sollen diese Merkmale unabhängig sein von der gegebenen Aufgabenstellung sowie der Komplexität.

1. Ein Automat wird von außen mit diskreten Eingaben versorgt, auf die er reagieren kann.
Im vorliegenden Beispiel sind die Eingaben durch die Einwürfe der Geldstücke gegeben, wobei in jedem Zustand die Eingabe jedes der drei Geldstücke erfolgen kann.
Eingaben können allgemein durch umgangssprachliche Formulierungen wie `Startknopf drücken`, `Geld einwerfen`, `Eingabezeichen (Wert einer Variablen) lesen`, `Lichtschranke unterbrechen`, usw. beschrieben werden. In der Regel gibt es eine begrenzte Menge von Eingaben, die in allen oder nur jeweils in einer Teilmenge der Zustände akzeptiert werden. Dabei kann auch der Fall auftreten, dass eine Eingabe zwar akzeptiert wird, jedoch keine beobachtbare Reaktion zur Folge hat. Dies wird im Zustandsdiagramm durch eine Kante augedrückt, die in den Zustand zurückführt.

Um von diesen verbalen Beschreibungen der die Zustandsübergänge auslösenden Ereignisse zu abstrahieren, werden wir dafür zur formalen Beschreibung in der Automatendarstellung einzelne Symbole oder Zeichen wie im vorherigen Beispiel die Zeichen G1, G2 oder G3 verwenden. Genauer betrachtet können auch Werte bestimmter Variablen (beispielsweise $A = 5$) abgefragt bzw. als Eingabe akzeptiert werden. In der formalen Darstellung eines Automaten wollen wir diese endliche Menge $E = \{e_1, e_2, ..., e_k)$ von Eingabezeichen als **Eingabealphabet** E bezeichnen.

2. Ein Automat befindet sich zu jedem Zeitpunkt in einem bestimmten Zustand.
Dieser momentan bestehende Zustand umfasst alle Informationen, die sich aus bisherigen Eingaben und Zustandsfolgen ergeben und für das weitere Verhalten von Bedeutung sind. Das weitere Verhalten des Automaten entwickelt sich in der Zukunft ausschließlich in Abhängigkeit vom momentan bestehenden Zustand und den in diesem Zustand wirkenden Eingaben. Durch diese Tatsache wird der Automat quasi gedächtnisfrei in Bezug auf Verhalten in der Vergangenheit, er entwickelt sich in der Zukunft quasi nur in Abhängigkeit von der Gegenwart. Es ist nicht mehr von Bedeutung über welchen Pfad der Automat in den aktuellen Zustand gekommen ist.

Beispiel 4.3 *Automatenverhalten*
Im Beispiel des Getränkeautomaten 4.2 kann dieser Automat vom Zustand *Start* aus über den Zustand *GS2* als auch über den Zustand *GS3* in den Zustand *GS4* kommen. Von da aus kommt der Automat völlig unabhängig von den vorher eingenommenen Zuständen mit dem nächsten Einwurf, egal welcher Art, in den Zustand *GS*. □

Unter der Einwirkung der Eingabe kann der Automat eine Menge von Zuständen durchlaufen. Diese Zustände können, abhängig von der konkreten Anwendung, unterschiedlichster Natur sein. Angefangen von den zwei trivialsten Zuständen `System eingeschaltet` und `System aus` oder `Zeichen A`

`gelesen`, `Geldstücke G1 und G2 eingeworfen`, können Zustände auch durch zusammengesetzte Aussagen wie `Steuerwerk verarbeitet Befehl, Rechenwerk frei` festgelegt sein. In der formalen Darstellung werden die Zustandsbeschreibungen in der Regel durch Zeichen abgekürzt, denen dann eine bestimmte Bedeutung gegeben wird. Häufig werden die Zustände einfach mit $s_1, s_2, ...$ bezeichnet.
Wir werden also für das Automatenmodell die Existenz einer endlichen Menge von Zuständen $S = \{s_0, s_1, ..., s_n\}$ annehmen, die wir als **Zustandsmenge** oder auch als Zustandsraum bezeichnen.

3. Ein Automat besitzt eine Übergangsfunktion.
Die Übergangsfunktion eines Automaten legt fest, welche Übergänge zwischen den einzelnen Zuständen möglich sind und durch welche Ereignisse diese ausgelöst werden. Genauer gesagt gibt die Übergangsfunktion zu jedem Zustand die aus diesem Zustand heraus führenden Zustandsübergänge, die im Zusammenhang mit Automaten auch als Transitionen bezeichnet werden, sowie die dazu nötigen Ereignisse und die entsprechenden Folgezustände an. Im Zustandsdiagramm wird jede Kante mit dem entsprechenden Ereignis markiert, dabei müssen alle aus einem Zustand herausführenden Kanten einen vollständigen Ereignisraum bilden, d.h. jedes Ereignis, das eintreten kann, muss durch eine Kante repräsentiert werden.

4. Ein Automat produziert zu bestimmten Zeitpunkten Ausgaben.
In der Regel generiert ein Automat als Folge der Ausführung von Zustandsübergängen und der Einnahme von Zuständen für die Umgebung relevante Informationen, die explizit ausgegeben werden können. Dabei handelt es sich um Ergebnisse des ausgeführten Prozesses, um Statusinformationen über den Zustand des ausgeführten Prozesses oder um Bestätigungen über einzelne ausgeführte Aktionen. Praktische Beispiele solcher Ausgaben sind `Ware ausgegeben`, `Vorgang beendet` oder das zahlenmäßige Ergebnis einer Rechenoperation. Analog zum Eingabealphabet nehmen wir an, dass es ein **Ausgabealphabet** $A = \{a_1, a_2, ..., a_k\}$ gibt. Ansonsten wollen wir die Ausgaben von Automaten in einem späteren Abschnitt genauer betrachten.

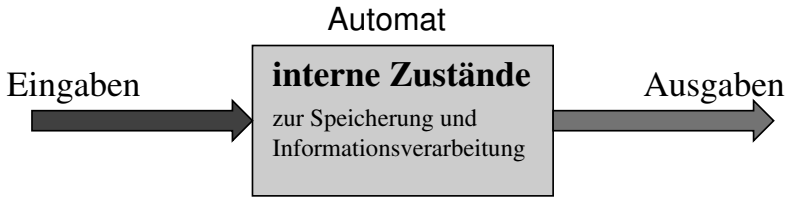
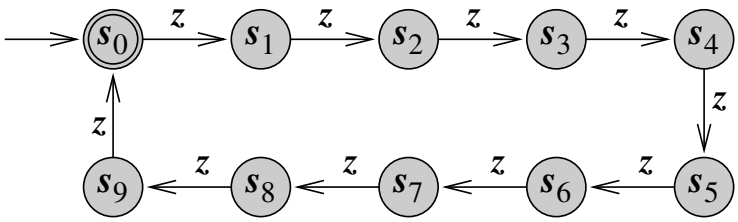


Abbildung 4.3: Allgemeines Prinzip eines Automaten

Zusammenfassend kann man sagen, dass ein Automat eine endliche Folge von Eingaben erhält, von denen er nacheinander jede einzelne Eingabe verarbeitet. Die Verarbeitung einer solchen Eingabe kann völlig unterschiedliche Reaktionen zur Folge haben, im Hinblick auf die abstrahierende formelle Betrachtung des Automaten kann dies jedoch nur der Wechsel des Zustandes und/oder die Ausgabe eines Zeichen sein, der durch die Übergangsfunktion festgelegt ist. Dabei kann die Reaktion vom momentan bestehenden Zustand und von der Eingabe abhängig sein. Das zentrale Element dieses Automaten ist der interne Speicher, der aus nur einer Speicherzelle besteht, dem sog. Zustandsspeicher. Der Inhalt dieses Zustandsspeichers repräsentiert den momentan bestehenden Zustand des Automaten. Die zentrale Erkenntnis aus dem Beispiel des Getränkeautomaten ist jedoch, dass es offensichtlich möglich ist, Automatenfunktionen bzw. dessen Abläufe ausschließlich durch Zustände und Zustandsübergänge (Transitionen) zwischen diesen Zuständen zu beschreiben.

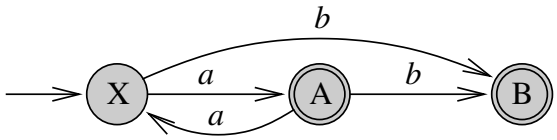
Beispiel 4.4 *Beschreibung eines Modulo-10-Zählers im Automatendiagramm*
Ein Beispiel eines populären Automaten ist der Modulo-10-Zähler, der die an einem Eingang auftretenden Eingaben zählt. Die Eingaben bestehen in diesem Fall aus Zählsignalen, dies können beispielsweise Impulse sein.

Der Modulo-10-Zähler besitzt 10 Zustände $S = \{s_0, s_1, ..., S_9\}$, die eingenommenen Zählerstände, die im internen Zustandsspeicher abgelegt sind. Er besitzt nur eine Art Eingabe, die Zählsignale z, die in



jedem Zustand auftreten können. Jedes Zählsignal in einem beliebigen Zustand löst einen Zustandsübergang in den Zustand mit dem jeweils nächsthöheren Index aus. Ein Zählsignal im Zustand s_9 führt zu einem Übergang in den Zustand s_0 zurück, der Zähler beginnt seinen Zählzyklus von vorne. □

Beispiel 4.5 *Beschreibung eines Verwaltungsvorganges im Automatenmodell*
Wie technische Vorgänge können auch organisatorische Abläufe durch ein Automatenmodell beschrieben werden. Wir betrachten eine Verwaltung, in der eine Abteilung X Bearbeitungsaufträge entgegen nimmt. Nach der Bearbeitung eines Auftrages und Bestimmung der Auftragsart (*a* oder *b*) in Abteilung X wird dieser entsprechend der Auftragsart an Abteilung A (bearbeitet *a* weiter) oder an Abteilung B (bearbeitet *b* weiter) weitergeleitet.



Gibt es in Abteilung A ein Problem mit dem Auftrag *a*, so wird dieser an Abteilung X zurück verwiesen. Stellt die Abteilung A andererseits fest, dass der Auftrag *a* auch von Abteilung B bearbeitet werden muss, so leitet sie diesen unter *b* an Abteilung B weiter. □

Die Beschreibung des Zusammenspiels der Zustände und der Zustandsübergänge, die das Verhalten des Automaten spezifizieren, erfolgte in den bisherigen Beispielen in Form des Zustandsdiagramms. Zustandsdiagramme bilden eine sehr eingängige Möglichkeit zur Visualisierung der Arbeitsweise bzw. des Verhaltens eines Automaten. Bei einer größer werdenden Zahl von Zuständen sind jedoch andere Darstellungen des Verhaltens angebracht. Ein zweite Möglichkeit zur Verhaltensbeschreibung ist die Zustandsübergangsmatrix oder die Zustandstabelle. Unabhängige Variablen in der Zustandstabelle sind der momentane Zustand und die Eingabe, abhängige Variablen sind der Folgezustand sowie eine mögliche Ausgabe. Die Zustandstabelle enthält also eine Spalte für den bestehenden Zustand und für jede Eingabe je eine Spalte für den Folgezustand. Die mögliche Ausgabe kann mit beim Folgezustand vermerkt werden. Die Zustandstabelle enthält für jeden Zustand eine Zeile, wobei ein Zeileneintrag aus einem geordneten Paar aus Folgezustand und möglicher Ausgabe besteht.

Beispiel 4.6 *Zustandstabellen der beiden Automatenmodelle aus den Beispielen 4.4 und 4.5*

S^n	S^{n+1}
s_0	s_1
s_1	s_2
s_2	s_3
s_4	s_5
s_5	s_6
s_6	s_7
s_7	s_8
s_8	s_9
s_9	s_0

S^n	S^{n+1}	
	a	b
X	A	B
A	X	B
B	–	–

z^n bestehender Zustand
z^{n+1} Folgezustand

□

4.2.3 Einführung der Begriffe Alphabet, Wort und formale Sprache

Wir wollen an dieser Stelle einige wichtige Begriffe vorstellen, die in der Theorie der endlichen Automaten von zentraler Bedeutung sind.

Definition 4.1 *Alphabet und Zeichenreihe*
Unter einem Alphabet *versteht man allgemein eine endliche, nicht leere Menge von atomaren (atomar im Sinne von unzertrennbar) Elementen, den Symbolen bzw. Zeichen.*
Unter einer Zeichenreihe *(auch als Wort, Zeichenkette oder Zeichenfolge bezeichnet) versteht man eine endliche Folge von Symbolen eines bestimmten Alphabets.*

Jedem bekannte Alphabete sind das *binäre* Alphabet $E_b = \{0,1\}$ oder die Menge der Kleinbuchstaben $E_k = \{a,b,...,z\}$. Beispielsweise ist *automat* eine Zeichenreihe über dem Alphabet der Kleinbuchstaben und 100110 eine Zeichenreihe über dem binären Alphabet $E_b = \{0,1\}$.

Zeichenreihen (*Wörter*) über E sind also endlich lange Zeichenfolgen, die über dem Alphabet $E = \{e_1,e_2,...,e_k\}$ gebildet werden können. Weitere Zeichenreihen entstehen, in dem Symbole oder bereits existierende Zeichenreihen aneinander gereiht, d.h. miteinander verkettet oder konkateniert werden. Sind u und v Wörter, dann steht uv für die Konkatenation von u und v. Insbesondere enthält die leere Zeichenreihe keine Symbole, sie wird durch das Zeichen ε dargestellt und kann aus jedem Alphabet kommen. Wir können damit E^*, die Menge aller Wörter, die über dem Alphabet E gebildet werden können, mathematisch beschreiben:

1.) $a \in E \Longrightarrow a \in E^*$
Jedes Symbol aus dem Alphabet E ist eine Zeichenreihe (Wort) über E^*.

2.) $v,w \in E^* \Longrightarrow vw \in E^*$
Werden bereits existierende Zeichenreihen (Wörter) v und w miteinander verkettet, dann entsteht eine neue Zeichenreihe vw.

3.) $\varepsilon \in E$, die leere Zeichenreihe ist ein Wort über jedem E, das leere Wort ist ein definiertes Wort ohne Ausdehnung mit der Eigenschaft: $\varepsilon w = w\varepsilon = w$ für alle $w \in E^*$.

Häufig werden Zeichenreihen nach ihrer Länge klassifiziert, d.h. der Anzahl der für Symbole verfügbaren Positionen. Die Länge des Wortes wa wird berechnet, in dem zur Länge $l(w)$ des Wortes w eine 1 addiert wird: $l(wa) = l(w)+1$ für $w \in E^*$, $a \in E$. Weiter wird die Menge aller Zeichenreihen einer bestimmten Länge über dem Alphabet E durch E^k bezeichnet. Dabei gilt $E^0 = \{\varepsilon\}$, ungeachtet dessen welches Alphabet E bezeichnet. Für E^* folgt somit:

$$E^* = E^0 \cup E^1 \cup E^2 \cup E^3 \cup \ldots = E^+ \cup \{\varepsilon\}$$

Beispiel 4.7 *Zeichenreihen*
Nehmen wir beispielsweise das binäre Alphabet $E_b = \{0,1\}$, dann ist $E^1 = \{0,1\}$,
$E^2 = \{00,01,10,11\}$ und $E^3 = \{000,001,010,011,100,101,110,111\}$.
Weiter gilt $E^+ = \{0,1,00,01,10,11,000,001,010,011,100,101,...\}$ und
$E^* = \{\varepsilon,0,1,00,01,10,11,000,001,010,011,100,101,...\}$.
Sind u=1001 und v=011001 zwei Worte aus E_b, dann ist uv=1001011001.
Die Eingabe eines Automaten besteht im allgemeinen aus Zeichenreihen, dem *Eingabealphabet* $E = \{e_1,e_2,...,e_k\}$. Das Eingabealphabet des Getränkeautomaten beispielsweise ist gegeben durch die Menge der Geldstücke $E = \{G_1,G_2,G_3\}$, das des Modulo-Zählers durch den Zählimpuls $E = \{z\}$. □

Aufgabe 4.2 *Zeichenreihen*
Bestimmen Sie E^3 für $E = \{a,b,x\}$. ◇

Formale Sprache
Ein Menge von Zeichenreihen aus E^*, wobei E ein bestimmtes Alphabet darstellt, wird als *Sprache* L bezeichnet. Anders ausgedrückt, wenn E ein Alphabet ist und $L \subseteq E^*$, dann ist L eine Sprache

über dem Alphabet E. Wesentlich dabei ist, dass in den Zeichenreihen einer Sprache über E nicht alle Symbole aus E vorkommen müssen. Oder anders herum ausgedrückt, wenn L eine Sprache über E ist, dann ist L gleichzeitig auch eine Sprache über jedem Alphabet, das eine Obermenge von E darstellt. Damit definieren wir den Begriff der (formalen) Sprache:

Definition 4.2 *(Formale) Sprache*
Sei E ein Alphabet, dann heisst jede Menge $L \subseteq E^*$ eine (formale) Sprache über E.

Eine formale Sprache L ist also eine Menge von Zeichenreihen (Wörtern) $\subseteq E^*$, die aus den Symbolen eines beliebigen Alphabets aufgebaut ist. E^* ist eine Sprache für jedes beliebige Alphabet E. Die leere Menge $\emptyset$ ist eine Sprache für jedes beliebige Alphabet E. Die davon der leeren Menge $\emptyset$ verschiedene Menge $\{\varepsilon\}$ ($\emptyset \neq \{\varepsilon\}$), die nur die leere Zeichenreihe enthält, ist ebenfalls eine Sprache. Die leere Sprache $\emptyset$ enthält keine Zeichenreihen, $\{\varepsilon\}$ hingegen enthält die leere Zeichenreihe.

Beispiel 4.8 *Zeichenreihen und Sprache*
Besteht das Alphabet aus einem Zeichen $E = \{a\}$, dann ist $L(E) = E^* = \{\varepsilon, a, aa, aaa, ...\}$, die Menge aller Wörter über $E = \{a\}$, eine Sprache über E.
Ist $E = \{0,1\}$, dann ist $L(E) = \{1, 11, 111\}$ eine Sprache über $E = \{0,1\}$.
Ebenso ist $L(E) = E^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$ eine Sprache über E. □

Sprachen können eine unendliche Anzahl von Zeichenreihen enthalten, sie sind aber immer auf Zeichenreihen über einem festen, endlichen Alphabet beschränkt. Es mag etwas seltsam erscheinen, den Begriff Sprache für Zeichenreihen in der Informatik zu verwenden. Letztendlich ist aber das, was wir schlechthin als natürliche Sprache verstehen, in der wir uns ausdrücken, genau so aufgebaut wie die Sprache L, die wir hier definiert haben. Andere Beispiele für Sprachen sind alle Programmiersprachen wie Java oder C, deren Alphabet eine Teilmenge des ASCII-Zeichensatzes ist.

Beispiel 4.9 *Häufige Informatik-Sprachen*
Im Zusammenhang mit endlichen Automaten treten bestimmte Sprachen häufig auf. Abstrakte Beispiele für Sprachen über $E_b = \{0,1\}$ sind:
Die Sprache aller Zeichenreihen, die aus n Nullen gefolgt von n Einsen bestehen,
$\{\varepsilon, 01, 0011, 000111, ...\}$.
Die Sprache aller Zeichenreihen aus Nullen und Einsen, die jeweils aus der gleichen Anzahl von Nullen und Einsen bestehen, $\{\varepsilon, 01, 10, 0011, 1100, 0101, 1010, ...\}$.
Die Sprache aller Zeichenreihen aus Nullen und Einsen, deren Binärwert eine Primzahl ist,
$\{10, 11, 101, 111, 1011, ...\}$. □

Zur Beschreibung einer Sprache wird in dieser Kurseinheit wie auch sonst in der Mathematik üblich, meistens eine Mengendefinition verwendet:

$$\{w \mid \text{Aussage über } w\}$$

Damit wird die Menge aller Wörter beschrieben, für die die angegebene Aussage gilt. Häufig wird w dabei durch einen Ausdruck mit Parametern ersetzt und die Zeichenreihe der Sprache durch für die Parameter geltende Bedingungen beschrieben.

Beispiel 4.10 *Definition von Sprachen*
Die Zeichenreihe $\{\varepsilon, ab, aabb, aaabbb, ...\}$ wird auch durch
$\{w \mid w \text{ enthält die gleiche Anzahl von } a \text{ und } b\}$ oder durch $\{a^n b^n \mid n \geq 0\}$ beschrieben.
Die Sprache $\{0^i 1^j \mid 0 \leq i \leq j\}$ besteht aus Zeichenreihen, die einige oder keine Nullen gefolgt von mindestens ebenso vielen Einsen enthalten. □

4.2.4 Spezifikation eines Automaten im Automatenmodell

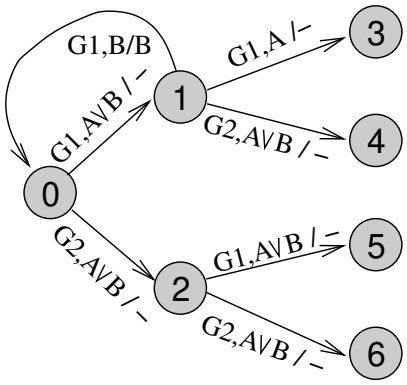
Wir wollen nun ein größeres und umfangreicheres Beispiel eines praktischen Automaten betrachten, bei dem wir auch explizit die entsprechenden Ausgaben spezifizieren wollen. Um nicht eine weitere Anwendung einzuführen und zu erklären, werden wir den vorher betrachteten Getränkeautomaten entsprechend modifizieren. Wir werden dem Getränkeautomaten durch mehr Eingabemöglichkeiten eine größere Komplexität geben.

Der Automat akzeptiere nun nur noch zwei Geldstücks- bzw. Münzarten G1 und G2, wobei G2 dem halben Betrag von G1 entspricht (G1=2G, G2=G). Er kann aber mit jedem Verkaufsvorgang wahlweise eines von zwei Getränken A oder B ausgeben. Das Getränk A koste einen Betrag S1=6G, das Getränk B einen Betrag S2=4G. Der jeweilige Betrag kann durch den Einwurf beliebiger Kombinationen der beiden Geldstücksarten in beliebiger Reihenfolge beglichen werden. Aus dieser Vereinbarung resultiert unmittelbar die Möglichkeit einer entsprechenden Geldrückgabe.

Die Auswahl der Getränkeart erfolge über einen Wahlschalter für das Getränk A oder B. Die Stellung des Wahlschalters kann jederzeit verändert werden, und zwar bis zu dem Zeitpunkt, zu dem für das zuletzt gewählte Getränk ein ausreichender Betrag eingeworfen wurde. Die Ausgabe des gewählten Getränkes erfolgt unmittelbar mit dem letzten notwendigen Münzeinwurf. Den Abbruch eines Verkaufsvorganges durch einen zusätzlichen Geldrückgabekopf wollen wir der Einfachheit halber nicht modellieren, da dadurch in unserer Spezifikation keine neuen für uns interessanten Aspekte untersucht werden, sondern im wesentlichen nur die Zahl der Transitionen erhöht und der Zustandsraum vergrößert wird.

Um den Zustandgraphen zu diesem Getränkeautomaten zu entwickeln, müssen wir alle möglichen Folgen von Eingabekombinationen betrachten, die zu einem erfolgreichen Abschluss eines Verkaufsvorganges führen. Zu Beginn befindet sich der Automat in einem Ausgangszustand 0 (*Start*). In diesem Ausgangszustand 0 wie auch in jedem weiteren Zustand sind die folgenden möglichen Eingaben von Bedeutung: Der Wahlschalter für die Getränkeart kann sich in der Stellung A oder in der Stellung B befinden oder dorthin gestellt werden. Es kann ein Geldstück G1 oder G2 eingeworfen werden. Aus der Kombination entstehen vier verschieden mögliche Eingaben: Einwurf von G1 bei Wahlschalter in Stellung A, Einwurf von G1 bei Wahlschalter in Stellung B, G2 in Kombination mit A und G2 kombiniert mit B. Die entsprechenden Kennzeichnungen der Transitionen sind G1,A/–, G1,B/–, G2,A/– und G2,B/–. Die Kennzeichnung vor dem Schrägstrich entspricht der Eingabe, die nach dem Schrägstrich entspricht der Ausgabe.

Beginnen wir unsere Betrachtung mit dem Ausgangszustand 0. Aufgrund der vier unterschiedlichen Eingaben kann der Zustand 0 auf vier Wegen verlassen werden, wobei die vier Zustandsübergänge im allgemeinen in vier unterschiedliche Folgezustände führen. Theoretisch könnten durch die vier Zustandsübergänge aber auch die gleichen oder teilweise die gleichen Folgezustände erreicht werden. Dies ist immer dann der Fall, wenn das durch das Zustandsdiagramm beschriebene Verhalten von diesem Zeitpunkt an nicht mehr unterscheidbar ist.



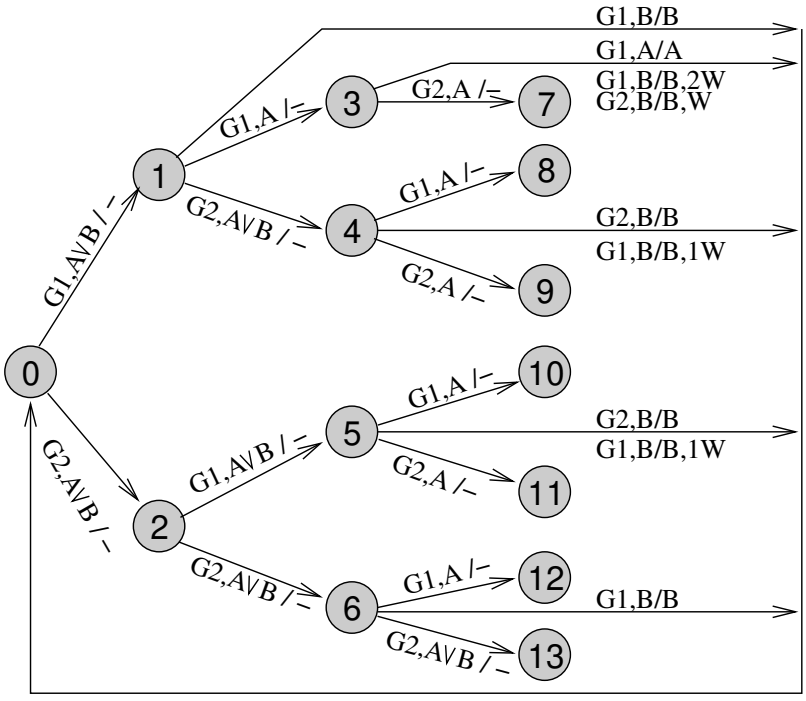
Betrachten wir zuerst den Einwurf eines Geldstücks G1. Damit befindet sich ein Betrag 2G im Automaten, was für keines der beiden Getränke ausreichend ist. Es ist also bedeutungslos in welcher

Stellung sich der Wahlschalter befindet, wir kommen unabhängig davon in einen neuen Zustand 1, der den Einwurf von G1 abbildet. Analog gilt dies für den Einwurf eines Geldstücks G2, wir kommen unabhängig von der Stellung des Wahlschalters in einen neuen Zustand 2, der den Einwurf von G2 abbildet. Mit diesen beiden Transitionen ist das mögliche Verhalten im Zustand 0 vollständig beschrieben.

Zur weiteren Entwicklung betrachten wir nun die beiden Folgezustände 1 und 2. Wir stellen uns in den Zustand 1 und überlegen, welche Ereignisse eintreten können und wie diese mit den vier möglichen Transitionen zu modellieren sind. Durch den weiteren Einwurf eines Geldstücks G1 würde sich ein Geldbetrag 4G im Automaten befinden, was für ein Getränk B ausreichend ist. Befindet sich der Wahlschalter in Stellung B, dann muss der Automat das Getränk B ausgeben. Da mit der Ausgabe von Getränk B der Verkaufsvorgang abgeschlossen ist, muss der Automat durch dieses Ereignis zurück in den Ausgangszustand 0 kommen um für den nächsten Verkaufsvorgang bereit zu sein. Damit haben wir eine der vier Transition abgehandelt, wir kennzeichnen diese Transition mit G1,B/B. Die restlichen drei Transition können leicht gemeinsam abgehandelt werden. Befindet sich der Wahlschalter beim Einwurf von G1 in Stellung A, dann kommen wir mit G1,A/- in einen neuen Zustand 3. Wird ein Geldstück G2 eingeworfen, dann kommen wir mit G2,A/- und G2,B/-, ausgedrückt durch G2,A∨B/-, in einen neuen Zustand 4.

In Zustand 2 spielt die Stellung des Wahlschalters keine Rolle, wir können weder mit dem Einwurf von G2 oder G1 einen Verkaufsvorgang abschließen. Wird ein Geldstück G1 eingeworfen, dann kommen wir mit G1,A∨B/- in einen neuen Zustand 5. Wird ein Geldstück G2 eingeworfen, dann kommen wir mit G2,A∨B/- in einen neuen Zustand 6. An dieser Stelle könnte man nun argumentieren, dass die Zustände 4 und 5 identisch sind, da in beiden der gleiche Geldbetrag 3G eingeworfen wurde. Wir wollen diese Diskussion äquivalenter Zustände aber auf einen späteren Zeitpunkt verschieben. Der weitere Entwurf läuft analog zur Entwicklung der Zuständen 1 und 2 ab. Wir begeben uns gedanklich nacheinander in die Zuständen 3, 4, 5 und 6 und generieren entsprechend den möglichen Ereignissen jeweils die vier Transitionen.

Aufgabe 4.3 *Weitere Entwicklung des Zustandsdiagramms*
Überlegen Sie wie die Transitionen aus den Zuständen 3 und 4 heraus aussehen bzw. wo diese hinführen. ◇



Bei der Betrachtung des Zustandes 3 fällt auf, dass das gewählte Getränk bereits bezahlt ist, wenn der Wahlschalter auf B steht. Man kann sich hier mehrere Alternativen vorstellen, wie der Automat in diesem Fall reagieren soll. Die erste Strategie wäre, dass der Automat durch den Wechsel des Wahlschalters von A nach B das Getränk B unmittelbar ausgibt und den Verkaufsvorgang abschließt. In einer praktischen Realisierung wird man sicherlich diese Strategie wählen. Aus Gründen der größeren Komplexität bei der Modellierung, die uns aber keinen neuen Lerneffekt bringt, wählen wir diese Alternative hier nicht. Diese Alternative hätte zur Folge, dass wir eine dritte Art Eingabe, nämliche *keine* Eingabe, modellieren müssten und dadurch aus jedem Zustand heraus sechs statt vier Transitionen bekämen.

Alternativ dazu können wir den Automaten so ausführen, dass er erst auf den nächsten Einwurf reagiert. Damit wird aber auch die Rückgabe von Wechselgeld impliziert, was ohnehin erforderlich ist und andererseits jedoch unser Automatenmodell nicht komplexer werden lässt. Der Grund ist, dass es sich bei der Berücksichtigung des Wechselgeldes um eine Ausgabe handelt, welche die Zahl der Transitionen nicht erhöht.

Kommen wir auf die Eingabe G1,B im Zustand 3 zurück. Damit befinden sich 6G im Verkaufsprozess, das Getränk B wird also ausgegeben und gleichzeitig auch 2G Wechselgeld wodurch der Verkaufsvorgang mit dem Übergang nach Zustand 0 abgeschlossen ist. Wir kennzeichnen die Transition mit G1,B/B,2W, wobei W den Geldwert G beschreibt. Entsprechend dazu kommen wir durch die Eingabe G2,B mit der Transition G2,B/B,1W nach Zustand 0 zurück. So wie wir für die Zustände 0, 1, 2 und 3 das Automatenmodell schrittweise entwickelt haben, müssen wir dies auch für die Zustände 4, 5, 6, 7, 8 und 9 und für alle weiteren dabei generierten Zustände tun. Dabei ergibt sich schließlich das folgende Zustandsdiagramm mit insgesamt 20 Zuständen.

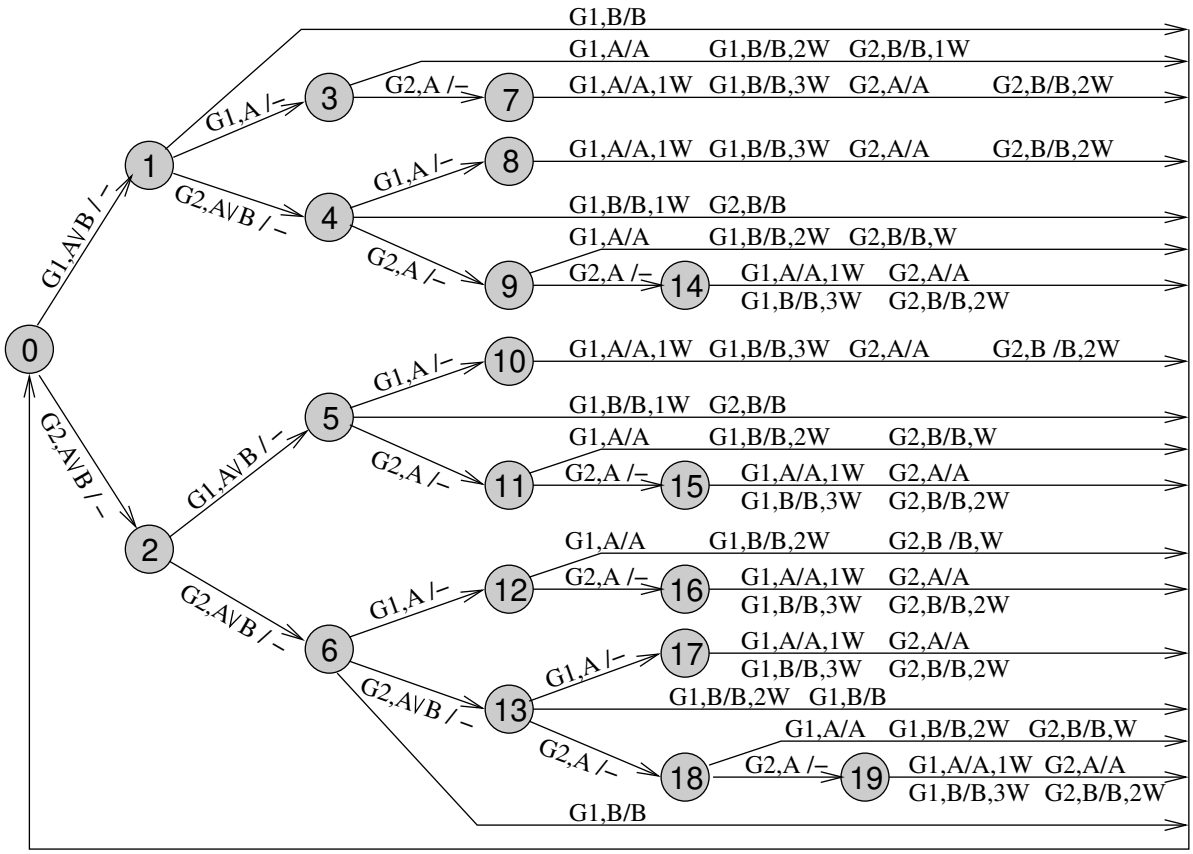


Abbildung 4.4: Vollständiges Zustandsdiagramm des Getränkeautomaten

Da es in jedem Zustand 4 Eingabemöglichkeiten gibt, gehen vom jedem Zustand 4 Transitionen aus. Bei 20 Zuständen besitzt das Diagramm demnach 80 Transitionen. Da oft mehrere Transitionen von demselben Zustand ausgehend in denselben Zielzustand (insbesondere bei Zustand 0) führen, haben

Zustand	bisherige Eingabe	Zustandsdefinition	
0	-	Ausgangszustand	0 G gespeichert
1	G1	$1 \times$ G1 eingeworfen	2 G gespeichert
2	G2	$1 \times$ G2 eingeworfen	1 G gespeichert
3	G1,G1	$2 \times$ G1 eingeworfen	4 G gespeichert
4	G1,G2	$1 \times$ G1, $1 \times$ G2 eingeworfen	3 G gespeichert
5	G2,G1	$1 \times$ G2, $1 \times$ G1 eingeworfen	3 G gespeichert
6	G2,G2	$2 \times$ G2 eingeworfen	2 G gespeichert
7	G1,G1,G2	$2 \times$ G1, $1 \times$ G2 eingeworfen	5 G gespeichert
8	G1,G2,G1	$2 \times$ G1, $1 \times$ G2 eingeworfen	5 G gespeichert
9	G1,G2,G2	$1 \times$ G1, $2 \times$ G2 eingeworfen	4 G gespeichert
10	G2,G1,G1	$2 \times$ G1, $1 \times$ G2 eingeworfen	5 G gespeichert
11	G2,G1,G2	$1 \times$ G1, $2 \times$ G2 eingeworfen	4 G gespeichert
12	G2,G2,G1	$1 \times$ G1, $2 \times$ G2 eingeworfen	4 G gespeichert
13	G2,G2,G2	$3 \times$ G2 eingeworfen	3 G gespeichert
14	G1,G2,G2,G2	$1 \times$ G1, $3 \times$ G2 eingeworfen	5 G gespeichert
15	G2,G1,G2,G2	$1 \times$ G1, $3 \times$ G2 eingeworfen	5 G gespeichert
16	G2,G2,G1,G2	$1 \times$ G1, $3 \times$ G2 eingeworfen	5 G gespeichert
17	G2,G2,G2,G1	$1 \times$ G1, $3 \times$ G2 eingeworfen	5 G gespeichert
18	G2,G2,G2,G2	$4 \times$ G2 eingeworfen	4 G gespeichert
19	G2,G2,G2,G2,G2	$5 \times$ G2 eingeworfen	5 G gespeichert

Tabelle 4.1: Definition der Zustände des Getränkeautomaten

wir die entsprechenden Kanten zusammengefasst. Diese Zusammenfassung dient ausschließlich der besseren Übersichtlichkeit des Zustandsdiagramms. Die zusammengefassten Kanten wurden dann mit allen durch die Kante beschriebenen Transitionen gekennzeichnet. Der Getränkeautomat besitzt also die Zustandsmenge

$$S = \{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19\}$$

Das Eingabealphabet ist definiert durch

$$E = \{(G1;A),(G1;B),(G2;A),(G2;B)\}$$

und ein Ausgabealphabet ist gegeben durch

$$A = \{(A;-),(A;1W),(A;2W),(A;3W),(B;-),(B;1W),(B;2W),(B;3W)\}$$

mit den folgenden Bedeutungen

G1 = Geldstück G1 eingeworfen	A = Ausgabe Getränk A
G2 = Geldstück G2 eingeworfen	B = Ausgabe Getränk B
A = Getränk A gewählt	1W = Ausgabe Wechselgeld 1G
B = Getränk B gewählt	2W = Ausgabe Wechselgeld 2G
	3W = Ausgabe Wechselgeld 3G
	- = keine Ausgabe

In der Tabelle 4.1 haben wir zusammenfassend die Bedeutung der Zustände des Getränkeautomaten aufgelistet. Genau genommen müssten wir noch eine weitere Eingabe $0 \hat{=}$ `kein Geldstück` definieren und alle Transitionen mit dieser Eingabe in den jeweiligen Zustand zurückgehen lassen. Der Einfachheit halber haben wir dies hier vernachlässigt und die Annahme gemacht, dass eine Transition nur in Verbindung mit einem Geldeinwurf ausgeführt werden kann, die Transition wird sozusagen durch den Einwurf ausgelöst. Mit dieser Annahme erübrigt sich dann auch die Schalterstellung `kein Getränk gewählt`.

Natürlich lässt sich dieser Getränkeautomat auch in Form einer Zustandstabelle beschreiben (Tabelle 4.2). Zu jeder möglichen Eingabe gibt es eine Spalte und zu jedem möglichen Zustand eine Zeile.

Zustand	**Folgezustand**			
S^n	S^{n+1}			
Eingabe	G1,A	G1,B	G2,A	G2,B
0	1/-	1/-	2/-	2/-
1	3/-	0/B	4/-	4/-
2	5/-	5/-	6/-	6/-
3	0/A	0/B,2W	7/-	0/B,1W
4	8/-	0/B,1W	9/-	0/B
5	10/-	0/B,1W	11/-	0/B
6	12/-	0/B	13/-	13/-
7	0/A,1W	0/B,3W	0/A	0/B,2W
8	0/A,1W	0/B,3W	0/A	0/B,2W
9	0/A	0/B,2W	14/-	0/B,1W
10	0/A,1W	0/B,3W	0/A	0/B,2W
11	0/A	0/B,2W	15/-	0/B,1W
12	0/A	0/B,2W	16/-	0/B,1W
13	17/-	0/B,1W	18/-	0/B
14	0/A,1W	0/B,3W	0/A	0/B,2W
15	0/A,1W	0/B,3W	0/A	0/B,2W
16	0/A,1W	0/B,3W	0/A	0/B,2W
17	0/A,1W	0/B,3W	0/A	0/B,2W
18	0/A	0/B,2W	19/-	0/B,1W
19	0/A,1W	0/B,3W	0/A	0/B,2W

Tabelle 4.2: Zustandstabelle des Getränkeautomaten

Wie man sieht ist die Zustandstabelle für Modelle mit mehr Zuständen wesentlich kompakter und damit geeigneter als das Zustandsdiagramm. Um beispielsweise das Verhalten des Automaten zu ermitteln, wenn man im Ausgangszustand sechs Geldstücke G2 einwirft und den auf A stehenden Wahlschalter nach dem Einwurf der fünften Münze auf B stellt, dann kann man dies in der Tabelle durch zeilenweises Auffinden der jeweiligen Folgezustände herausfinden. Wesentlich einfacher ist dies jedoch im Zustandsdiagramm. Man stellt sich in Zustand 0 und fährt entsprechend den Eingaben entlang der Kanten über die Zustände 2,6,13,18,19 und erkennt schließlich an der von Zustand 19 nach Zustand 0 verlaufenden Kante, dass das Getränk B und Wechselgeld im Wert von 2G ausgegeben wird.

Gegenüber dem Zustandsdiagramm aus Beispiel 4.1 auf Seite 2 ist dieses Diagramm wesentlich komplexer geworden. Die größere Komplexität wird insbesondere durch die größere Zahl an möglichen Eingaben verursacht. Die Spezifikation von praktischen Problemen in der Softwaretechnik erzeugt i.A. Zustandsräume mit tausenden von Zuständen, wobei die große Anzahl an Zuständen aus der Komposition einzelner Systeme resultiert. Im Gegensatz zum Beispiel 4.1 haben wir hier auch den Rückweg in den Ausgangszustand genutzt um die Ware auszugeben, sodass der Automat mit der Ausgabe den Verkaufsvorgang abschließt und sich wieder im Ausgangszustand befindet.

Wir haben hier einen iterativen Weg gewählt, um das Zustandsdiagramm des Getränkeautomaten zu entwickeln. In vielen Fällen ist auch ein anderer Weg geeigneter. Dabei wird zuerst die Menge der Zustände bestimmt. Im zweiten Schritt werden alle Transitionen zwischen diesen Zuständen festgelegt. Dieser Weg ermöglich auch die Überprüfung der vorher bestimmten Zustandsmenge, denn sollte sich bei der Festlegung der Transitionen eine Transition ergeben, die in keinem der vorab bestimmten Zustände endet, dann hat man bei der Bestimmung der Zustandsmenge einen Fehler gemacht. Weiter fällt auf, dass wir den gleichzeitigen Einwurf zweier Geldstücke nicht betrachtet haben. Abgesehen davon, dass dies hier aus technischen Gründen wahrscheinlich gar nicht möglich ist, kann ein solcher Vorgang durch ein Automatenmodell nicht beschrieben werden. Im Automatenmodell werden alle auftretenden Ereignisse, die wir auch als Aktionen bezeichnen, sequentialisiert. Ereignisse, die

tatsächlich gleichzeitig (d.h. in der differentiell kleinen Zeit $t \to 0$) ablaufen, werden als eine Aktion bzw. eine Transition modelliert. Man bezeichnet diese Ereignisse auch als atomar. Alle anderen Ereignisse werden im Modell durch zwei sequentielle Transitionen beschrieben.

Wenn wir die betrachteten Beispiele vergleichen, dann sind die Zustandsdiagramme um so größer je komplexer der beschriebene Ablauf ist und um so detaillierter die Beschreibung. Je mehr Eingaben, um so mehr Transitionen ergeben sich und damit erhöht sich auch in der Regel die Zahl der Zustände. Umgekehrt heisst das, dass wir mit wachsender Zahl an Eingabe- und Ausgabesymbolen sowie vor allem der Zustände mehr Sachverhalte modellieren können, wobei zwischen den einzelnen Modellen ausschließlich ein quantitativer, aber kein qualitativer Unterschied besteht. Wir haben die Abläufe mit einem Automatenmodell beschrieben, d.h. Zustände und Transitionen sind als Beschreibungselemente völlig ausreichend, um die Abläufe in einer geeigneten Abstraktion zu beschreiben. Dies sollen auch die wesentlichen Erkenntnise aus diesem Abschnitt sein, denn das Spezifizieren von komplexen Systeme geht weit über das Kursziel hinaus.

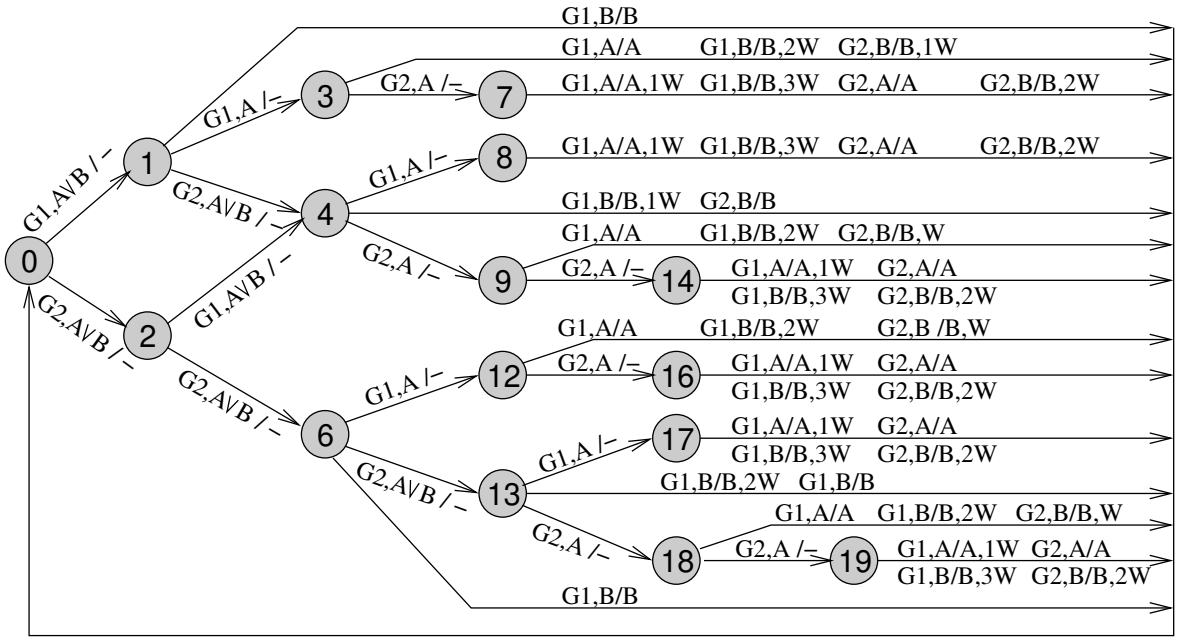
Eine der wichtigen Fragen bei der Entwicklung einer Spezifikation in Form eines Zustandsdiagramms ist natürlich, wann ist das Zustandsdiagramm vollständig und kommt man immer nach endlich vielen Transitionen in den Ausgangszustand zurück? Vollständig ist es dann wenn alle Zustände des realen Systems berücksichtigt wurden. Ein Indiz für die Vollständigkeit ist, dass aus jedem vorhandenen Zustand heraus die den Eingaben entsprechende Zahl an Transitionen herauslaufen muss. Für all diese Transitionen muss ein entsprechender Zustand vorhanden sein, in dem diese enden. Bei allen Automaten, die selbständig in Abhängigkeit von Eingaben einen bestimmten Vorgang wiederholt ausführen, muss man nach endlich vielen Transitionen in den Ausgangszustand zurückkommen.

Im Falle des Getränkeautomaten muss ein Verkaufsvorgang sicherlich nach endlich vielen Transitionen abgeschlossen sein. Man kann sich leicht vorstellen, dass alles andere unbrauchbar wäre. Selbst wenn man davon ausgeht, dass der Verkaufsautomat durch eine externe Störung oder einen Fehler in einen Zustand gerät, der im entwickelten Zustandsdiagramm nicht enthalten ist und der Automat sich daraus von selbst nicht mehr befreien kann, dann kommt der Automat letztendlich durch ein Aus-/Einschalten oder eine eventuelle Reparatur wieder in den Ausgangszustand zurück.

4.2.5 Reduzierung des Zustandsraumes

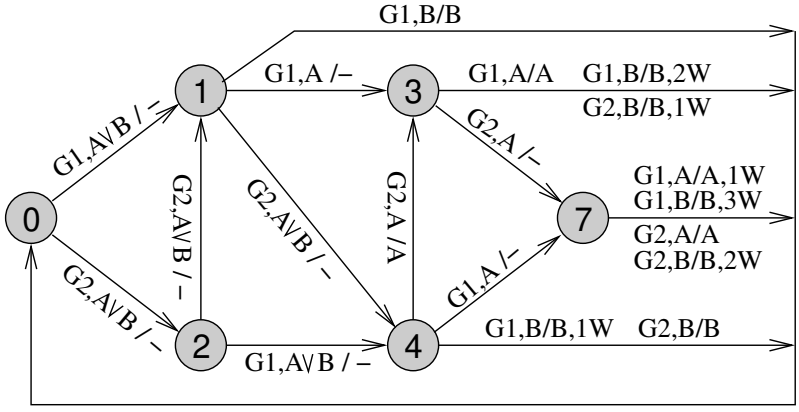
Im Folgenden soll noch eine weitere Betrachtung an dem Modell des Getränkeautomaten durchgeführt werden. Wir wollen uns überlegen, ob auch ein anderes Zustandsdiagramm existiert, welches das gleiche Verhalten äquivalent beschreibt. Dieses kann mehr oder aber weniger Zustände aufweisen, es muss aber die gleichen Transitionssequenzen, durch welche die Verkaufsvorgänge modelliert werden, ausführen können.

Schaut man sich das entwickelte Zustandsdiagramm genauer an, dann erkennt man, dass dort auch Vereinfachungen in Form des Zusammenlegens von Zuständen vorgenommen werden können, ohne die durch das Zustandsdiagramm beschriebene Funktion zu verändern. In Zustand 4 beispielsweise wurde zuerst ein Geldstück G1 und dann ein Geldstück G2 eingeworfen, in Zustand 5 erfolgte der gleiche Einwurf in umgekehrter Reihenfolge. Offensichtlich können die Zustände 4 und 5 als gleichwertig angesehen werden, wenn die Reihenfolge des Münzeinwurfs für die Funktion bzw. das weitere Verhalten des Automaten nicht von Bedeutung ist. Vergleicht man die Transitionen und die Folgezustände, die auf die Zustände 4 und 5 folgen, dann erkennt man, dass beide Transitionssequenzen identisch sind. Die Folgezustände sind bis auf die Bezeichnung identisch. Mit dem Einwurf von G1 folgt auf Zustand 4 der Zustand 8, entsprechend folgt auf Zustand 5 der Zustand 10. Mit dem Einwurf von G2 folgt auf Zustand 4 der Zustand 9, entsprechend folgt auf Zustand 5 der Zustand 11. Bis auf die Reihenfolge des Münzeinwurfs sind die beiden Zustände nicht unterscheidbar. Die Äquivalenz ist auch aus der Tabelle 4.1 zu erkennen. In den äquivalenten Zuständen wurde der gleiche Geldbetrag gespeichert und dies ist genau die Information, die für das weitere Verhalten des Automaten von Bedeutung ist.



Durch die Zusammenlegung der Zustände 4 und 5 wird das Zustandsdiagramm also wesentlich vereinfacht ohne die beschriebene Funktion verändert zu haben. Nach diesem Prinzip können wir aber noch weitere Vereinfachungen vornehmen. Ohne die beschriebene Funktion zu verändern können wir die Zustände 9 und 12 zusammenlegen. Ferner stellen wir fest, dass auch der Zustand 3 das gleiche Verhalten aufweist wie der Zustand 9. Gleiches gilt für die Zustände 12 und 18. Diese können im Zustand 3 zusammengelegt werden. Nach allen weiteren möglichen Zusammenlegungen ergibt sich schließlich das dargestellte Zustandsdiagramm mit nur noch 6 Zuständen.

Über schwierigere und komplexere Überlegungen hätten wir auch direkt dieses Diagramm entwerfen können. Das wäre uns jedoch schwer gefallen. Wir erkennen unmittelbar, dass der eingeschlagene Lösungsweg der gedanklich einfachere und sichere Weg war, das Diagramm zu entwickeln. Und wir erkennen auch, wie uns die Automatentheorie letztlich zu einer optimalen Lösung geführt hat.



Wir wollen diese Reduktion des Zustandsraumes als intuitive Vorwegnahme ansehen. Wir werden nach der formalen Definition des Automaten auch formal definierte Regeln kennenlernen, mit denen die Äquivalenz von Zuständen eindeutig berechenbar ist.

In den betrachteten Beispielen haben wir das Zustandsdiagramm, das auch als Automatenmodell bezeichnet wird, dazu verwendet, Abläufe in Automaten zu modellieren. Natürlich findet das Automatenmodell auch bei vielen anderen Phänomenen in der Informatik Anwendung. Ein Beispiel für eine weitere Anwendung ist der Entwurf von Compilern (Übersetzern) oder Parsern, die aus einer Sequenz von Eingabezeichen bestimmte Sequenzen identifizieren. Daneben wollen wir später noch eine weitere Anwendung kennenlernen, die es ermöglicht, anhand des Automatenmodells Aussagen

über die Komplexität und die Berechenbarkeit einer Aufgabe oder die Ausführung eines Algorithmus zu machen.

In den folgenden Abschnitten werden wir an die hier eingeführten Begriffe anknüpfen, diese weiter präzisieren und formalisieren. Wir werden damit den endlichen Automaten formal einführen, definieren und anschließend damit im Zusammenhang mit Automaten stehende Konzepte und Methoden behandeln.

4.3 Formale Einführung endlicher Automaten

Ein endlicher Automat ist ein mathematisches Modell eines Systems (oder allgemeiner eines Prozesses), das diskrete Eingaben entgegennimmt, verarbeitet und diskrete Ausgaben generiert. Dieses System befindet sich zu jedem Zeitpunkt in genau einem aus einer endlichen Menge von Zuständen. Ein bestehender Zustand umfasst alle Informationen, die sich aus den bisherigen Eingaben sowie den bisher eingenommenen Zuständen ergeben haben und die benötigt werden, um die Reaktion des System auf zukünftige Eingaben zu bestimmen.

Ein *endlicher Automat (EA)* besteht aus einer endlichen Menge von Zuständen, der *Zustandsmenge* $S = \{s_1, s_2, ..., s_n\}$ und einer endlichen Menge von Eingabezeichen, dem **Eingabealphabet** $E = \{e_1, e_2, ..., e_k\}$. Ein *endlicher Automat (EA)* geht in Abhängigkeit des aktuellen bestehenden Zustandes s_i sowie des nächsten zu verarbeitenden Zeichens e_j des Eingabealphabets in einen neuen Zustand über, den Folgezustand s_k. Diese Verhaltensweise wird durch δ, die Zustandsübergangsfunktion oder kurz Übergangsfunktion, festgelegt:

$$\delta : S \times E \longrightarrow S$$

δ wird auf geordnete Paare (s_i, e_j), bestehend aus einem Zustand s_i und einer Eingabe e_j, angewendet und liefert einen Folgezustand s_k. Genauer gesagt bedeutet $\delta(s_i, a) = s_k$, dass der Automat, wenn er sich im Zustand s_i befindet und das Zeichen a liest, in den Zustand s_k übergeht. Dabei existiert für jedes Eingabezeichen genau ein Zustandsübergang, der möglicherweise auch in den gleichen Zustand zurückführt. Die Zustandsübergangsfunktion δ, welche die Arbeitsweise des Automaten im Wesentlichen bestimmt, haben wir in den vergangenen Beispielen in Form des Zustandsdiagramms oder der Zustandstabelle dargestellt.

4.3.1 Endliche Automaten ohne Ausgabe

Die einfachste Klasse endlicher Automaten sind die Automaten ohne Ausgabe. In dem im vorherigen Abschnitt betrachteten Beispiel des Getränkeautomaten bewegte sich dieser aufgrund des bestehenden Zustandes und der momentanen Eingabe stets in einen Folgezustand und generierte dabei eine Ausgabe. Bei einem endlichen Automaten ohne Ausgabe ist die Situation einfacher. Nach der Verarbeitung eines Eingabewortes, einer Zeichenreihe, befindet sich der Automat in einem sog. *Endzustand* oder *nicht*. Im Endzustand ist das Eingabewort akzeptiert worden, in allen anderen Zuständen nicht. Dabei handelt es sich um eine einfache binäre Entscheidung.

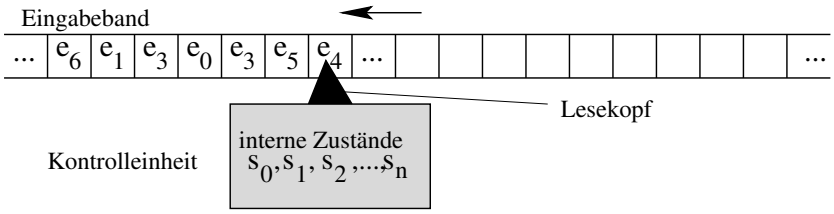


Abbildung 4.5: Endlicher Automat ohne Ausgabe

Um uns eine abstrakte, aber anschaulische Vorstellung von einem endlichen Automaten ohne Ausgabe zu schaffen, betrachten wir ein Band, das fortlaufend in Felder unterteilt ist. In jedem Feld steht jeweils genau ein Zeichen aus dem Eingabealphabet. Der endliche Automat stellt eine Kontrolleinheit dar, die sich in einem bestimmten Zustand s_i befindet und eine Folge von Zeichen von dem Eingabeband liest und verarbeitet. In jedem Schritt wird genau ein Zeichen a bearbeitet, die Kontrolleinheit wechselt dabei von Zustand s_i nach Zustand s_k und positioniert den Lesekopf über dem nächsten Symbol.

Mit dieser informellen Beschreibung des Automaten wollen wir nun einen endlichen Automaten ohne Ausgabe formal definieren:

Definition 4.3 *Endlicher Automat ohne Ausgabe*
Ein endlicher Automat EA ohne Ausgabe, im folgenden kurz endlicher Automat (EA) genannt, ist durch ein Quintupel

$$EA = (S, E, \delta, s_0, F)$$

definiert, wobei

$$\begin{array}{rcll} S & = & \{s_0, s_1, \ldots, s_n\} & \textit{eine nichtleere Menge, die Zustandsmenge}, \\ E & = & \{e_0, e_1, \ldots, e_r\} & \textit{eine nichtleere Menge, das Eingangsalphabet}, \\ \delta & : & S \times E \longrightarrow S & \textit{die Zustandsübergangsfunktion}, \\ s_0 & \in & S & \textit{der Anfangszustand und} \\ F & \subseteq & S & \textit{die nichtleere Menge der Endzustände ist}. \end{array}$$

Ein EA besteht aus den angegebenen fünf Komponenten und einem Ablaufmechanismus, dem Zusammenwirken der Komponenten. Neben der Zustandsübergangsfunktion δ sowie den Mengen S und E, treten in der obigen Definition des EA ohne Ausgabe noch der Anfangszustand $s_0 \in S$ und die Menge der Endzustände $F \subseteq S$ auf. Mit dem Anfangszustand beginnt die Verarbeitung jedes Eingabewortes. Die Menge der Endzustände ermöglicht es dem Automaten zwischen den Eingabewörter zu unterscheiden, die er *akzeptiert* und denen, die er *nicht akzeptiert*. Befindet sich der Automat nach dem Verarbeiten eines Wortes in einem der Endzustände aus F, dann wurde das Wort akzeptiert, andernfalls nicht. Natürlich kann ein Endzustand auch wieder verlassen werden (siehe Beispiel 4.11).

Das Zustandsdiagramm, das wir bereits ausführlich betrachtet haben, ist ein gerichteter Graph zur Beschreibung der Zustandsübergangsfunktion $\delta : S \times E \longrightarrow S$. Jeder Knoten entspricht einem Zustand s_i, jede gerichtete Kante entspricht einer Transition von einem Zustand s_i in einen anderen Zustand s_k, wobei $s_k = s_i$ sein kann. Die Kante ist gekennzeichnet durch das Eingabezeichen e_i, welches diese Transition entsprechend der Übergangsfunktion $\delta(s_i, e_j) = s_k$ bewirkt. Von jedem Knoten gehen so viele Kanten aus wie es Eingabemöglichkeiten gibt. Häufig werden der Anfangs- und die Endzustände im Diagramm besonders gekennzeichnet. Der Anfangszustand s_0 durch einen frei beginnenden Pfeil und die Endzustände durch einen Doppelkreis. Dabei können natürlich mehrere Beschriftungen zur Bezeichnung einzelner Transitionen an einer Kante angebracht werden.

4.3.2 Verhalten des endlichen Automaten ohne Ausgabe

Bei einem endlichen Automaten (EA) wird nicht jede Eingabe einen Zustandswechsel zur Folge haben. Der EA akzeptiert nur bestimmte Sequenzen von Eingaben, die ihn sukzessive vom Anfangszustand in einen Endzustand überführen. Wir wollen diese Verhaltensweise eines EA, d.h. die Zustandsfolgen, die sich ergeben, wenn er beginnend mit dem Anfangszustand eine Eingabefolge in Form einer Zeichenreihe verarbeitet, formal beschreiben. Dazu müssen wir unsere Zustandsübergangsfunktion δ erweitern, denn wir wollen δ nicht nur auf einen einzelnen Zustand und einzelne Zeichen, sondern direkt auf einen Zustand und ganze Zeichenreihen anwenden. Wir definieren also eine Funktion δ^* von $S \times E^*$ nach S, wobei E^* die Menge aller Zeichenreihen (Wörter) w darstellt, die über dem Eingangsalphabet E gebildet werden können (siehe Abschnitt 4.2.3). Die Funktion $\delta^*(s_i, w)$ liefert den Zustand s_k, in dem sich der EA befindet, wenn der Ausgangszustand s_i ist und die Zeichenreihe bzw. das Wort w gelesen wurde.

Definition 4.4 *Natürliche Fortsetzung δ^* von δ*
Es sei $\delta : S \times E \longrightarrow S$ die Zustandsübergangsfunktion eines EA.
Dann ist die natürliche Fortsetzung $\delta^* : S \times E^* \longrightarrow S$ der Funktion δ gegeben durch:

$$\begin{array}{lrcl} (1) & \delta^*(s, \varepsilon) & = & s \\ (2) & \delta^*(s, we) & = & \delta(\delta^*(s, w), e) \quad \textit{für alle} \quad s \in S,\ e \in E \textit{ und } w \in E^*. \end{array}$$

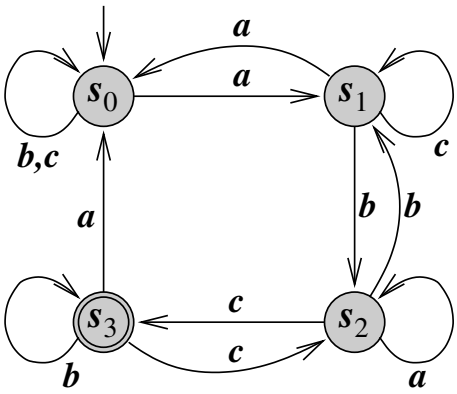
Anders ausgedrückt liefert $\delta^*(s_i, w)$ genau den Zustand s_k, für den es im Zustandsdiagramm einen Pfad von s_i nach s_k gibt, der mit w markiert ist. Teil (1) der Definition besagt, dass der EA seinen Zustand nicht ändern kann ohne ein Eingabezeichen zu lesen. Teil (2) gibt an wie der EA den Folgezustand nach dem Lesen von we bestimmt, indem er zunächst den Zustand $s_w = \delta^*(s, w)$ nach dem

Lesen von w berechnet, und daraus dann weiter den Zustand $s_{we} = \delta(s_w, e)$ bestimmt. Setzt man im Teil (2) der Definition $w = \varepsilon$, also die leere Zeichenreihe, dann folgt daraus mit $\delta^*(s,\varepsilon) = s$:

$$\delta^*(s, we) = \delta^*(s, \varepsilon e) = \delta^*(s, e) = \delta(\delta^*(s,\varepsilon), e) = \delta(s, e)$$

Es gilt also $\delta^*(s,e) = \delta(s,e)$, d.h. dass δ und δ^* für alle Argumente der Mengen S und E, für die beide definiert sind, identisch sind. Wir werden daher die Anwendung der Funktion δ auf ein einzelnes Zeichen oder eine Zeichenreihe nicht mehr durch δ und δ^* unterscheiden und in beiden Fällen nur noch δ schreiben.

Beispiel 4.11 *Zustandsübergänge eines EA*
Gegeben sei der endliche Automat EA$= (S, E, \delta, s_0, F)$ durch $S = \{s_0, s_1, s_2, s_3\}$, $E = \{a, b, c\}$, $F = \{s_3\}$ und der Übergangsfunktion δ, die durch das folgende Zustandsdiagramm bzw. die Tabelle beschrieben ist.



Übergangsfunktion δ in Tabellenform

δ	a	b	c
s_0	s_1	s_0	s_0
s_1	s_0	s_2	s_1
s_2	s_2	s_1	s_3
s_3	s_0	s_3	s_2

Wir wollen uns anhand einiger Eingabewörter klarmachen, wie der Automat arbeitet. Die Zeichenreihe *abc* wird durch folgende Zustandsübergänge verarbeitet:

$$\delta(s_0, a) = s_1, \quad \delta(s_1, b) = s_2, \quad \delta(s_2, c) = s_3, \quad \text{also gilt} \quad \delta(s_0, abc) = s_3$$

Die folgende Tabelle zeigt die Verarbeitung weiterer Wörter.

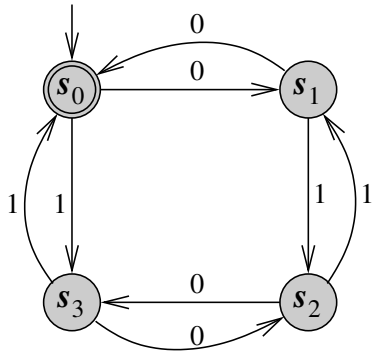
	Eingabewort	erreichter Zustand
(1)	abc	$\delta(s_0, abc) = s_3$
(2)	acb	$\delta(s_0, acb) = s_2$
(3)	bac	$\delta(s_0, bac) = s_1$
(4)	cab	$\delta(s_0, cab) = s_2$
(5)	$aabb$	$\delta(s_0, aabb) = s_0$
(6)	$abab$	$\delta(s_0, abab) = s_1$
(7)	$caabc$	$\delta(s_0, caabc) = s_0$
(8)	$cabab$	$\delta(s_0, cabab) = s_1$
(9)	$abccc$	$\delta(s_0, abccc) = s_3$
(10)	$abbbca$	$\delta(s_0, abbbca) = s_0$

Da s_3 der Endzustand ist, führen nur die Eingabeworte (1) und (9) in einen Endzustand. □

Beispiel 4.12 *Zustandsübergänge eines EA*
Gegeben sei der endliche Automat EA$= (S, E, \delta, s_0, F)$ durch $S = \{s_0, s_1, s_2, s_3\}$, $E = \{0, 1\}$, $F = \{s_0\}$ und der nachfolgend dargestellten Übergangsfunktion δ.

Die rechts stehende Tabelle zeigt die Verarbeitung verschiedener Wörter. Da s_0 der Endzustand ist, führen die Eingabeworte (4), (5), (6), (7), (8) und (9) in einen Endzustand. Der EA startet im Zustand s_0 und bewegt sich mit der Eingabe der ersten 0 nach rechts in Zustand s_1, mit der Eingabe der ersten 1 bewegt er sich nach unten in Zustand s_3. Mit der Eingabe der zweiten 0 bewegt er sich von Zustand

s_1 nach links zurück in Zustand s_0. Mit der Eingabe der zweiten 1 bewegt er sich von Zustand s_3 nach oben zurück in Zustand s_0. Befindet sich der EA im Zustand s_2, dann kommt er mit der Eingabe der nächsten 0 nach Zustand s_3, mit der Eingabe der nächsten 1 nach Zustand s_1.

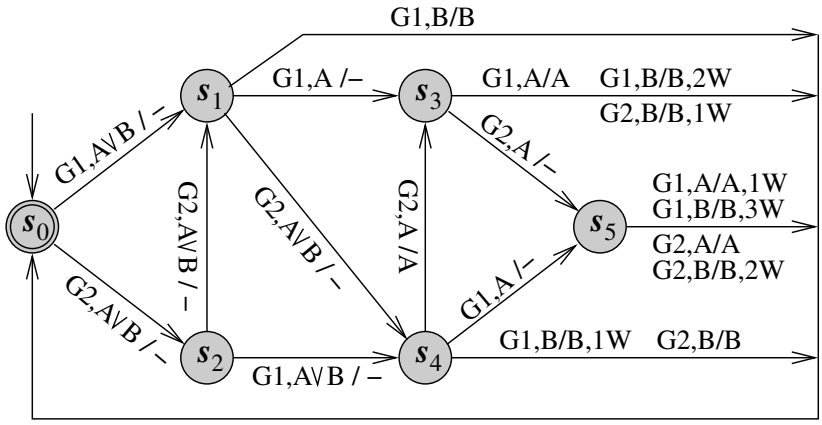


δ	0	1
s_0	s_1	s_3
s_1	s_0	s_2
s_2	s_3	s_1
s_3	s_2	s_0

	Eingabewort	erreichter Zustand
(1)	101	$\delta(s_0,101) = s_1$
(2)	010	$\delta(s_0,010) = s_3$
(3)	100	$\delta(s_0,100) = s_3$
(4)	1010	$\delta(s_0,1010) = s_0$
(5)	0101	$\delta(s_0,0101) = s_0$
(6)	1100	$\delta(s_0,1100) = s_0$
(7)	0011	$\delta(s_0,0011) = s_0$
(8)	0110	$\delta(s_0,0110) = s_0$
(9)	1001	$\delta(s_0,1001) = s_0$
(10)	01100	$\delta(s_0,01100) = s_1$
(11)	11001	$\delta(s_0,11001) = s_3$

Offensichtlich bewegt sich der EA aufgrund der ersten eingegebenen 0 jeweils nach rechts und mit der zweiten eingegebenen 0 jeweils nach links. Aufgrund der ersten eingegebenen 1 bewegt sich der EA jeweils nach unten und mit der zweiten eingegebenen 1 jeweils nach oben.
Der Automat befindet sich also nach einem Eingabewort genau dann im Endzustand s_0 wenn das Eingabewort sowohl eine gerade Anzahl von Nullen als auch eine gerade Anzahl von Einsen enthält. Der EA benutzt die Zustände um die Geradheit bzw. die Ungeradheit der Anzahl der Nullen und der Anzahl der Einsen zu speichern, die genaue Zahl wird jedoch nicht gespeichert. Dazu wären unendlich viele Zustände erforderlich. □

Beispiel 4.13 *Zustandsübergänge eines EA*
Gegeben sei der endliche Automat EA$= (S,E,\delta,s_0,F)$ durch $S = \{s_0,s_1,s_2,s_3,s_4,s_5\}$,
$E = \{(G1;A),(G1;B),(G2;A),(G2;B)\}$, $F = \{s_0\}$ und der folgenden Übergangsfunktion δ.



δ	$G1;A$	$G1;B$	$G2;A$	$G2;B$
s_0	s_1	s_1	s_2	s_2
s_1	s_3	s_0	s_4	s_4
s_2	s_4	s_4	s_1	s_1
s_3	s_0	s_0	s_5	s_0
s_4	s_5	s_0	s_3	s_0
s_5	s_0	s_0	s_0	s_0

Die folgende Tabelle zeigt die Verarbeitung verschiedener Wörter.

	Eingabewort	erreichter Zustand
(1)	*G1;A G1;A G1;A*	$\delta(s_0, G1;A\ G1;A\ G1;A) = s_0$
(2)	*G1;B G1;B G1;B*	$\delta(s_0, G1;B\ G1;B\ G1;B) = s_1$
(3)	*G2;B G2;B G1;B*	$\delta(s_0, G2;B\ G2;B\ G1;B) = s_0$
(4)	*G2;B G2;A G2;A G1;B*	$\delta(s_0, G2;B\ G2;A\ G2;A\ G1;B) = s_0$
(5)	*G1;B G1;A G2;A G2;A*	$\delta(s_0, G1;B\ G1;A\ G2;A\ G2;A) = s_0$
(6)	*G2;A G2;B G2;B G1;A G1;A*	$\delta(s_0, G2;A\ G2;B\ G2;B\ G1;A\ G1;A) = s_0$
(7)	*G2;A G2;B G2;A G2;A G2;A*	$\delta(s_0, G2;A\ G2;B\ G2;A\ G2;A\ G2;A) = s_5$
(8)	*G2;B G2;B G1;A G2;B G2;B*	$\delta(s_0, G2;B\ G2;B\ G1;A\ G2;B\ G2;B) = s_2$
(9)	*G1;A G2;A G2;A G2;A G2;A*	$\delta(s_0, G1;A\ G2;A\ G2;A\ G2;A\ G2;A) = s_0$

Da s_0 der Endzustand ist, führen die Eingabeworte (1), (3), (4), (5), (6) und (9) in einen Endzustand. Da es sich hier um den früher betrachteten Getränkeautomaten handelt, kann man den Eingaben auch eine Bedeutung zuordnen und die Korrektheit überprüfen. Es zeigt sich, dass immer dann wenn ein ausreichender Geldbetrag eingegeben wurde, so dass das gewählte Getränk plus Wechselgeld ausgegeben werden kann, der EA den Zustand s_0 erreicht. Wird kein ausreichender Geldbetrag für das gewählte Getränk eingegeben, dann erreicht der EA den Endzustand s_0 nicht. Wird ein Geldbetrag eingegeben, der über die Wechselgeldzahlung hinausgeht, so dass das Getränk zwischenzeitlich schon ausgegeben werden konnte wie im Fall (8), dann hat der Automat den nächsten Vorgang bereits begonnen und erwartet weitere Eingaben. □

Aufgabe 4.4 *Zustandsübergänge eines EA*
Gegeben sei der endliche Automat EA$= (S, E, \delta, s_0, F)$ durch $S = \{s_0, s_1, ..., s_{19}\}$,
$E = \{(G1;A), (G1;B), (G2;A), (G2;B)\}$ und $F = \{s_0\}$. Dabei handelt es sich um den Getränkeautomaten mit der in Abb. 4.4 dargestellten Übergangsfunktion δ.

Stellen Sie fest, welche Zustände vom Ausgangszustand s_0 aus durch die folgenden Eingabewörter erreicht werden:

	Eingabewort	erreichter Zustand
(1)	*G1;A G1;A G1;A*	
(2)	*G1;B G1;B G1;B*	
(3)	*G2;B G2;B G1;B*	
(4)	*G2;B G2;A G2;A G1;B*	
(5)	*G1;B G1;A G2;A G2;A*	
(6)	*G2;A G2;B G2;B G1;A G1;A*	
(7)	*G2;A G2;B G2;A G2;A G2;A*	
(8)	*G2;B G2;B G1;A G2;B G2;B*	
(9)	*G1;A G2;A G2;A G2;A G2;A*	

◇

An diesen Beispielen haben wir gesehen, dass ein endlicher Automat in Abhängigkeit von der eingegebenen Zeichenreihe ein bestimmtes Verhalten zeigt. Durch das Lesen der einzelnen Eingaben führt der Automat einzelne Transitionen aus und befindet sich nach der Eingabe eines bestimmten Wortes *w* in einem bestimmten Zustand *s*, der ein sog. *Endzustand* sein kann oder *nicht*. Befindet er sich in einem Endzustand, dann sagen wir, das Eingabewort *w* ist akzeptiert worden. Im nächsten Abschnitt wollen wir allgemeiner der Frage nachgehen, welche Eingabeworte von einem endlichen Automaten akzeptiert werden und welche nicht.

4.3.3 Konfiguration und Übergangsrelation eines endlichen Automaten

Als Sprache eines endlichen Automaten bezeichnet man die von ihm akzeptierte Menge von Zeichenreihen oder Wörtern, deren Verarbeitung in einen Endzustand führt. Um die durch diese Sprache beschriebenen Ausdrücke allgemein formal zu definieren, benötigen wir die Begriffe *Konfiguration* und *Übergangsrelation*. Unter einer Konfiguration eines technischen Systems versteht i.a. eine Momentaufnahme seines Gesamtzustandes. Zu einer solchen Konfiguration gehören alle charakteristischen Größen, die das weitere Verhalten in der Zukunft vollständig bestimmen. Wenden wir diesen Begriff auf unseren endlichen Automaten EA an, dann folgt daraus:

Definition 4.5 *Konfiguration eines endlichen Automaten (EA)*
Es sei $EA = (S, E, \delta, s_0, F)$ *ein endlicher Automat ohne Ausgabe. Unter einer Konfiguration des EA versteht man ein Paar* $(s,w) \in S \times E^*$*, wobei s als der zu diesem Zeitpunkt bestehende Zustand und w als die noch zu verarbeitende Zeichenreihe (Wort) verstanden wird.*

Das Paar $k_{EA} = (s,w)$ mit $s \in S$ und $w \in E^*$ beschreibt also den aktuellen Stand der Verarbeitung einer Eingabefolge w. Die Menge aller prinzipiell möglichen Konfigurationen K eines EA ist die Menge aller Paare von Zuständen und Zeichenreihen, d.h. $K = S \times E^*$.

Ein Übergang von einer Konfiguration $k_1 = (s_1, ew)$ zu einer Konfiguration $k_2 = (s_2, w)$ findet statt, falls in der Zustandsübergangsfunktion der Übergang $\delta(s_1, e) = s_2$ mit $s_1, s_2 \in S$, $e \in E$ und $w \in E^*$ enthalten ist. Einen Konfigurationsübergang von $k_1 = (s_1, w_1)$ nach $k_2 = (s_2, w_2)$ wollen wir allgemein durch den Ausdruck

$$(s_1, w_1) \models_{EA} (s_2, w_2)$$

darstellen. Das Symbol $\models_{EA}$ beschreibt den Konfigurationsübergang und wird gesprochen als `geht über in`. Mit Hilfe des Symbols $\models_{EA}$ können wir die Abarbeitung einer gelesenen Zeichenreihe $w = e_1 e_2 ... e_n$ durch den folgenden Ausdruck beschreiben:

$$(s_0, e_1, e_2 ... e_n) \models_{EA} (s_1, e_2 ... e_n) \models_{EA} \ldots \models_{EA} (s_n, \varepsilon) \quad \text{mit} \quad \delta(s_i, e_{i+1}) = s_{i+1},\ 0 \leq i \leq n.$$

Ist s_n ein Endzustand, ($s_n \in F$), dann wird das Wort w vom EA akzeptiert, sonst nicht.

Eine binäre Relation über zwei Mengen A und B ist allgemein eine Teilmenge R der Menge $A \times B$, also $R \subseteq A \times B$. Eine solche binäre Relation kann man auch über einer einzelner Menge A bilden, wobei jeweils zwei Elemente aus A zueinander in Beziehung stehen, also $R \subseteq A \times A$. Das Symbol $\models_{EA}$ stellt eine solche Relation über der Menge der Konfigurationen $K = S \times E^*$ dar. Wir können also auch schreiben:

$$\begin{aligned} R_{\models} &\subseteq K \times K \\ &\subseteq ((S \times E^*) \times (S \times E^*)) \end{aligned}$$

Die Relation $R_{\models}$ setzt jeweils zwei Konfiguration $k_1 = (s_1, ew)$ und $k_2 = (s_2, w)$ zueinander in Beziehung, für die der EA durch die Eingabe $e \in E$ von der Konfiguration k_1 in die Konfiguration k_2 übergeht. Wir wollen diese Beschreibung der Übergangsrelation in einer Definition zusammenfassen.

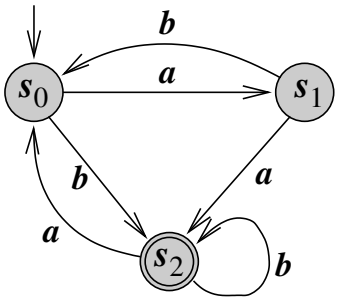
Definition 4.6 *Übergangsrelation eines EA*
Es sei $EA = (S, E, \delta, s_0, F)$ *ein endlicher Automat ohne Ausgabe. Die Übergangsrelation* $\models_{EA}$ *eines EA, gegeben durch*

$$R_{\models} \subseteq ((S \times E^*) \times (S \times E^*))$$

beschreibt alle möglichen Übergänge von einer Konfiguration (s, ew) *zu einer Folgekonfiguration* (s, w)*, die durch die Übergangsfunktion* δ *des EA hervorgerufen werden können:*

$$(s, ew) \models_{EA} (\delta(s, e), w) \quad \textit{für alle } s \in S,\ e \in E,\ \textit{und } w \in E^*$$

Beispiel 4.14 *Übergangsrelation* $\models_{EA}$
Gegeben sei der endliche Automat EA$= (S, E, \delta, s_0, F)$, der durch $S = \{s_0, s_1, s_2\}$, $E = \{a, b\}$, $F = \{s_2\}$ und die folgende Übergangsfunktion δ beschrieben ist.



Übergangsfunktion δ in Tabellenform

δ	a	b
s_0	s_1	s_2
s_1	s_2	s_0
s_2	s_0	s_2

Um die Übergangsrelation $\models_{EA}$ zu bestimmen, müssten wir zunächst alle möglichen Konfigurationen $K = S \times E^*$ auflisten. Dazu müssten wir vorher alle möglichen Zeichenreihen E^* des Eingabealphabets $E = \{a,b\}$ bilden. Wenn wir beliebige Längen der Zeichenreihen E^* zulassen, dann gibt es natürlich auch beliebig viele Worte, nämlich alle möglichen Kombinationen, in denen a und/oder b vorkommt:

$$E^* = \{\varepsilon, a, b, aa, bb, ab, ba, aaa, bbb, aab, bba, abb, baa, aba, bab, \dots\}$$

Um ein anschaulisches Beispiel zu erzeugen, begrenzen wir der Einfachheit halber die Menge der zu betrachtenden Eingabeworte w auf Worte mit nur zwei Zeichen, also $|w| \leq 2$. Damit sind die folgenden Worte E^2 für die Betrachtung relevant:

$$E^2 = \{\varepsilon, a, b, aa, bb, ab, ba\}$$

Daraus ergeben sich die folgenden Konfigurationen $K_2 = S \times E^2$:

$$\begin{array}{lllllll} (s_0,\varepsilon) & (s_0,a) & (s_0,aa) & (s_0,ab) & (s_0,b) & (s_0,bb) & (s_0,ba) \\ (s_1,\varepsilon) & (s_1,a) & (s_1,aa) & (s_1,ab) & (s_1,b) & (s_1,bb) & (s_1,ba) \\ (s_2,\varepsilon) & (s_2,a) & (s_2,aa) & (s_2,ab) & (s_2,b) & (s_2,bb) & (s_2,ba) \end{array}$$

Die Übergangsrelation $\models_{EA}$ enthält alle möglichen Übergänge von einer Konfiguration (s,ew) zu einer Folgekonfiguration (s,w), die aufgrund der Übergangsfunktion δ möglich sind:

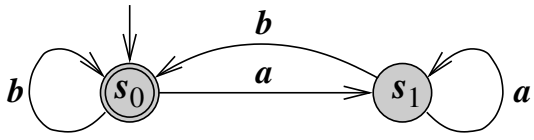
$$\begin{array}{lll} (s_0,a) \models_{EA} (s_1,\varepsilon) & (s_0,aa) \models_{EA} (s_1,a) & (s_0,ab) \models_{EA} (s_1,b) \\ (s_0,b) \models_{EA} (s_2,\varepsilon) & (s_0,bb) \models_{EA} (s_2,b) & (s_0,ba) \models_{EA} (s_2,a) \\ (s_1,a) \models_{EA} (s_2,\varepsilon) & (s_1,aa) \models_{EA} (s_2,a) & (s_1,ab) \models_{EA} (s_2,b) \\ (s_1,b) \models_{EA} (s_0,\varepsilon) & (s_1,bb) \models_{EA} (s_0,b) & (s_1,ba) \models_{EA} (s_0,a) \\ (s_2,a) \models_{EA} (s_0,\varepsilon) & (s_2,aa) \models_{EA} (s_0,a) & (s_2,ab) \models_{EA} (s_0,b) \\ (s_2,b) \models_{EA} (s_2,\varepsilon) & (s_2,bb) \models_{EA} (s_2,b) & (s_2,ba) \models_{EA} (s_2,a) \end{array}$$

Dargestellt als Relation $R_{\models} \subseteq ((S \times E^*) \times (S \times E^*))$ erhalten wir somit:

$$\begin{array}{rcl} R_{\models} & = & \{((s_0,a),(s_1,\varepsilon)), ((s_0,aa),(s_1,a)), ((s_0,ab),(s_1,b)), \\ & & ((s_0,b),(s_2,\varepsilon)), ((s_0,bb),(s_2,b)), ((s_0,ba),(s_2,a)), \\ & & ((s_1,a),(s_2,\varepsilon)), ((s_1,aa),(s_2,a)), ((s_1,ab),(s_2,b)), \\ & & ((s_1,b),(s_0,\varepsilon)), ((s_1,bb),(s_0,b)), ((s_1,ba),(s_0,a)), \\ & & ((s_2,a),(s_0,\varepsilon)), ((s_2,aa),(s_0,a)), ((s_2,ab),(s_0,b)), \\ & & ((s_2,b),(s_2,\varepsilon)), ((s_2,bb),(s_2,b)), ((s_2,ba),(s_2,a))\} \end{array}$$

□

Aufgabe 4.5 *Übergangsrelation* $\models_{EA}$
Gegeben sei der endliche Automat EA$= (S,E,\delta,s_0,F)$, der durch $S = \{s_0,s_1\}$, $E = \{a,b\}$, $F = \{s_0\}$ und die folgende Übergangsfunktion δ beschrieben ist.



Geben Sie die Übergangsrelation $\models_{EA}$ an, wobei die Menge der zu betrachtenden Eingabeworte w auf Worte mit zwei Zeichen, also $|w| \leq 2$, begrenzt werden sollen. ◇

4.3.4 Sprache eines endlichen Automaten

Unter der Sprache $L(EA)$ eines endlichen Automaten (EA) wollen wir die Menge an Zeichenreihen oder Worten $w \in E^*$ verstehen, die der EA akzeptiert. Ein EA akzeptiert genau die Menge an Zeichenreihen $w \in E^*$, durch dessen Eingabe der EA vom Anfangszustand s_0 in einen Endzustand $s \in F$ überführt wird. Solche Zeichenreihen werden durch Konfigurationsübergänge, deren Startkonfiguration k_0 den Anfangszustand s_0 ($k_0 = (s_0, w)$) und deren Endkonfiguration einen Endzustand $s \in F$ enthalten ($k_E = (s \in F, \varepsilon)$), also

$$(s_0, w) \models_{EA} (s \in F, \varepsilon)$$

oder mit Hilfe der Übergangsfunktion ausgedrückt:

$$\delta(s_0, w) = s \in F$$

Wir fassen diese Aussage in einer Definition zusammen:

Definition 4.7 *Sprache eines EA*
Es sei $EA = (S, E, \delta, s_0, F)$ ein endlicher Automat ohne Ausgabe. EA akzeptiert eine Zeichenreihe $w \in E^*$, falls $\delta(s_0, w) \in F$. Unter der Sprache $L(EA)$ des Automaten EA verstehen wir die Menge aller Worte w, die EA akzeptiert:

$$L(EA) \equiv \{w \in E^* \mid \delta(s_0, w) \in F\}$$

$L(EA)$ heisst die von EA akzeptierte Sprache.

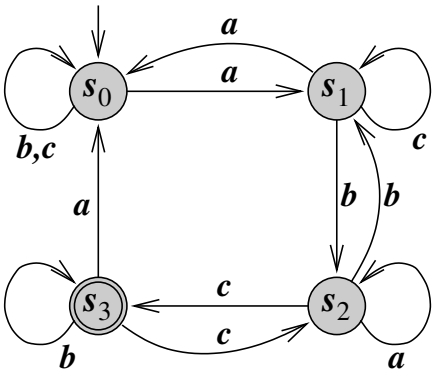
EAs ohne Ausgabe werden deshalb auch oft als Akzeptoren bezeichnet. Wir wollen an dieser Stelle bereits den Begriff der regulären Sprache im Zusammenhang mit EAs einführen, ohne genauer zu definieren, was eine reguläre Sprache ist.

Eine Sprache $L \subseteq E^*$ wird als *regulär* bezeichnet, falls es einen Automaten gibt, der diese Sprache L akzeptiert, d.h. für den $L = L(EA)$ gilt. Auf die Bedeutung und die Eigenschaften regulärer Ausdrücke und Sprachen werden wir im nächsten Abschnitt eingehen.

Beispiel 4.15 *Sprache eines endlichen Automaten*
Wir betrachten den endlichen Automaten EA$= (S, E, \delta, s_0, F)$ aus Beispiel 4.11 mit
$S = \{s_0, s_1, s_2, s_3\}$, $E = \{a, b, c\}$, $F = \{s_3\}$ und der nachfolgenden Übergangsfunktion δ.

Welche Konfigurationen treten beim Verarbeiten der folgenden Worte auf?
Welches der beiden Wort wird akzeptiert?

a) *abbcbacb* b) *caabc*



Übergangsfunktion δ in Tabellenform

δ	a	b	c
s_0	s_1	s_0	s_0
s_1	s_0	s_2	s_1
s_2	s_2	s_1	s_3
s_3	s_0	s_3	s_2

a) Wort *abbcbacb*

$$(s_0, abbcbacb) \models_{EA} (s_1, bbcbacb) \models_{EA} (s_2, bcbacb) \models_{EA} (s_1, cbacb) \models_{EA} (s_1, bacb)$$
$$\models_{EA} (s_2, acb) \models_{EA} (s_2, cb) \models_{EA} (s_3, b) \models_{EA} (s_3, \varepsilon)$$

Das Wort *abbcbacb* wird vom EA akzeptiert, da der Zustand s_3 in der letzten Konfiguration ein Endzustand ist, das Wort gehört damit zur Sprache des Automaten.

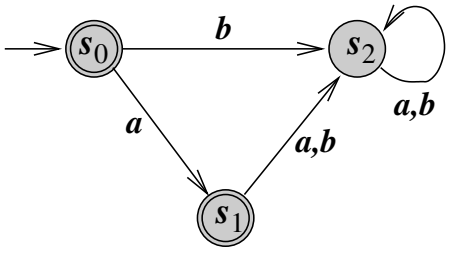
b) Wort *caabc*

$$(s_0, caabc) \models_{EA} (s_0, aabc) \models_{EA} (s_1, abc) \models_{EA} (s_0, bc) \models_{EA} (s_0, c) \models_{EA} (s_0, \varepsilon)$$

Das Wort *caabc* gehört nicht zur Sprache des Automaten, da der Zustand s_0 in der letzten Konfiguration kein Endzustand ist. □

Bisher haben wir nur Automaten betrachtet, deren Sprachen $L(EA)$ aus einer unendlichen Menge von Wörtern w bestanden. Es ist aber auch möglich, einen EA anzugeben, dessen Sprache eine endliche Menge ist.

Beispiel 4.16 *Endlicher Automaten mit endlicher Wortmenge*
Wir betrachten den endlichen Automaten EA$=(S,E,\delta,s_0,F)$ mit der Zustandsmenge $S=\{s_0,s_1,s_2\}$, dem Eingabealphabet $E=\{a,b\}$, den Endzuständen $F=\{s_0,s_1\}$ und der Übergangsfunktion δ.



δ	a	b
s_0	s_1	s_2
s_1	s_2	s_2
s_2	s_2	s_2

Dieser Automat besitzt die Sprache $L(EA)=\{\varepsilon,a\}$. Mit dem Wort $w=a$ kommt der Automat in den Endzustand s_1 und mit $w=\varepsilon$ kommt der Automat in den Endzustand s_0, der gleichzeitig auch Anfangszustand ist. □

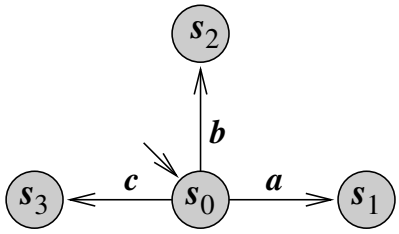
Aufgabe 4.6 *Sprache eines endlichen Automaten*
Gegeben sei der endliche Automat EA$=(S,E,\delta,s_0,F)$ durch $S=\{s_0,s_1,s_2\}$, $E=\{a,b\}$, $F=\{s_2\}$ und der Übergangsfunktion δ aus Beispiel 4.14.

Bestimmen Sie die Sprache des endlichen Automaten EA. ◇

In den bisherigen Beispielen sind wir von einem gegebenen Automaten ausgegangen und haben überprüft ob bestimmte Worte von diesem EA akzeptiert wurden, um so die Sprache $L(EA)$ des gegebenen EAs zu erkennen. Wir wollen nun die schwierigere umgekehrte Aufgabe betrachten, einen Automaten mit einer bestimmten Sprache $L(EA)$ zu entwerfen. Wir stellen uns also genauer die Frage, wieviele und welche Zustände, wieviele und welche Endzustände, und welche Übergangsfunktion δ muss der Automat aufweisen, der eine fest vorgegebene Sprache besitzt.

Beispiel 4.17 *Entwurf eines endlichen Automaten*
Wir suchen einen EA mit dem Eingabealphabet $E=\{a,b,c\}$, der die Menge aller Worte akzeptiert, in denen je zwei aufeinanderfolgende a, b oder c vorkommen, d.h.
$L(EA)=\{w \mid w$ enthält aa oder bb oder $cc\}$.

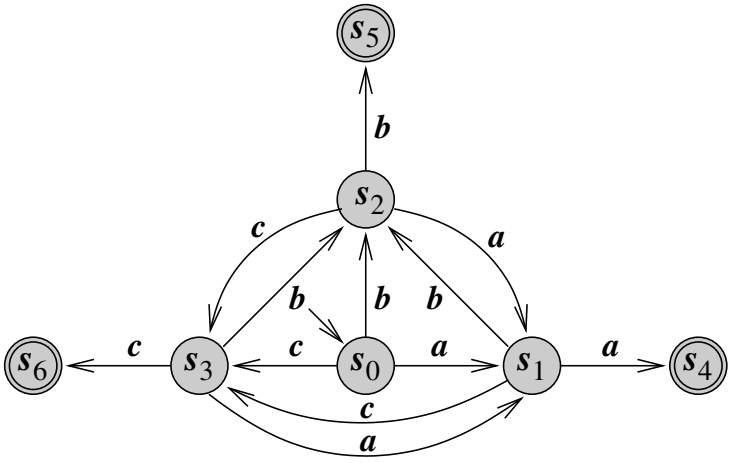
Gesucht sind also die Zustandsmenge S und die Übergangsfunktion δ. Für einen systematischen Entwurf gehen wir so vor, wie wir es zu Beginn dieser KE beim Entwurf der verschiedenen Variationen des Getränkeautomaten kennen gelernt haben.
Im vorliegenden Fall könnten wir vorab die Zustände festlegen und anschließend die Übergänge (Transitionen) zwischen diesen Zuständen bestimmen. Für den einfachsten Entwurf würden wir neben einem Anfangszustand 3 Endzustände, in denen jeweils eines der drei Worte aa, bb, oder cc verarbeitet wurde, und je einen Zwischenzustand, in dem je eines der Zeichen verarbeitet wurde, vorsehen. Wir wollen uns den Entwurf durch ein iteratives Vorgehen aber noch einfacher machen, in dem wir den Anfangszustand s_0 betrachten, und feststellen, was bei den drei möglichen Eingaben a, b und



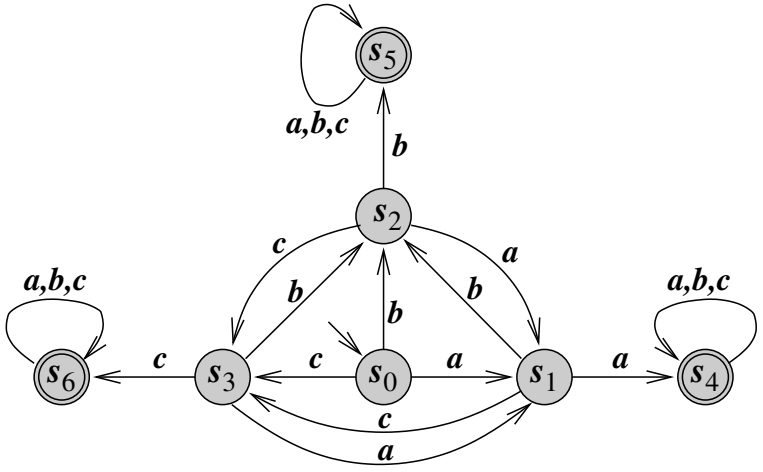
c passieren muss. Offensichtlich muss jedes der drei Zeichen den Automaten in einen neuen Zustand überführen. Daraus resultieren die 3 Folgezustände s_1, s_2 und s_3.

Als nächstes betrachten wir den Zustand s_1 mit den drei möglichen Eingaben a, b und c. Mit der Eingabe a haben wir bereits einen Endzustand, den wir s_4 nennen wollen, erreicht, in dem das erste Wort aa akzeptiert wurde. Mit der Eingabe b aber müssen wir natürlich in einen Zustand kommen, in dem ein b verarbeitet wurde. Dieser Zustand existiert bereits, es handelt sich um den Zustand s_2. Analog gilt dies für die Eingabe c; mit c kommen wir in den bereits existierenden Zustand s_3.

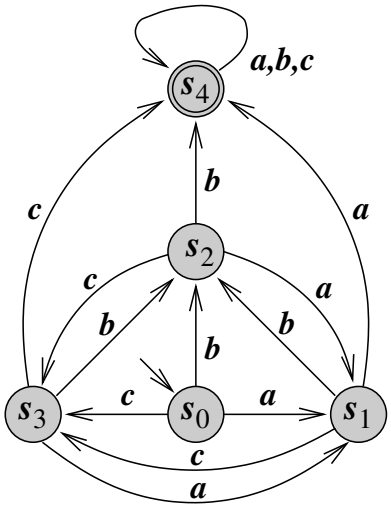
Anschließend nehmen wir den Zustand s_2 und betrachten diesen mit den drei möglichen Eingaben a, b und c. Mit der Eingabe a kommen wir in den existierenden Zustand s_1, in dem bereits ein a verarbeitet wurde. Mit der Eingabe c kommen wir in den Zustand s_3, in dem bereits ein c verarbeitet wurde. Mit der Eingabe b kommen wir in einen weiteren Endzustand s_5, in dem das zweite Wort bb akzeptiert wurde. Danach nehmen wir den Zustand s_3, mit der Eingabe a kommen wir in den Zustand s_1, mit der Eingabe b kommen wir in den Zustand s_2 und mit der Eingabe c kommen wir in einen weiteren Endzustand s_6, in dem das dritte Wort cc akzeptiert wurde.



Schließlich schauen wir uns noch die drei Endzustände s_4, s_5 und s_6 an; in allen drei Zuständen sind die drei Eingaben a, b und c möglich. Da jedoch keine der Eingaben eine Veränderung bewirken soll, führen die drei Transitionen a, b und c in den jeweiligen Zustand zurück. Damit ist der Automat vollständig.



Man erkennt auch unmittelbar, dass damit die drei Endzustände äquivalent sind und in einem Zustand s_4 zusammengefasst werden können. Der vollständige Automat lautet also: $\mathrm{EA} = (S, E, \delta, s_0, F)$ mit $S = \{s_0, s_1, s_2, s_3, s_4\}$, $E = \{a, b, c\}$, $F = \{s_4\}$ und der Übergangsfunktion δ:



δ	a	b	c
s_0	s_1	s_2	s_3
s_1	s_4	s_2	s_3
s_2	s_1	s_4	s_3
s_3	s_1	s_2	s_4
s_4	s_4	s_4	s_4

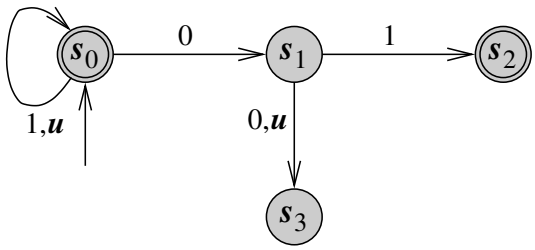
□

Beispiel 4.18 *Entwurf eines endlichen Automaten*
Wir suchen einen EA, der alle Eingaben akzeptiert, für die auf jede 0 genau eine 1 folgt. Ferner soll dieser Automat neben den beiden interessierenden Zeichen 0 und 1 auch noch das zusätzliche Zeichen u im Eingabealphabet besitzen. Es gilt also $E = \{0,1,u\}$. Da wir hier *jede* Eingabe, für die auf eine 0 eine 1 folgt, akzeptieren wollen folgt für die genaue Definition der Sprache $L(EA)$ dieses Automaten: $L(EA) = \{w \in E^* \mid w \text{ enthält direkt nach jeder 0 genau eine 1}\}$.
Beispiele für Worte $w \in L(EA)$ sind: $w = 01uuu01$, $w = 1u1u11$ und $w = 1u101$.
Beispiele für Worte $w \notin L(EA)$ sind hingegen: $w = 011u$, $w = 10u00$, $w = 1u10$ und $w = 0011$.

Gesucht sind die Zustandsmenge S und die Übergangsfunktion δ. Für einen intuitiv einfachen, aber systematischen Entwurf wählen wir auch hier, wie im vorherigen Beispiel, die iterative Methode.

Wir stellen uns in den Anfangszustand s_0 und fragen uns, was bei den drei möglichen Eingaben 0, 1 und u passieren soll/muss. Ist das verarbeitete Zeichen eine 0, dann kommen wir in einen neuen Zustand s_1. Ist das verarbeitete Zeichen eine 1 oder ein u, dann ist das für die weitere Funktion des Automaten bedeutungslos, da über führende 1 und u in $L(EA)$ nichts ausgesagt ist. Wir müssen uns also nicht merken, dass im Ausgangszustand ein solches Zeichen verarbeitet wurde und können einfach im Ausgangszustand s_0 verbleiben.

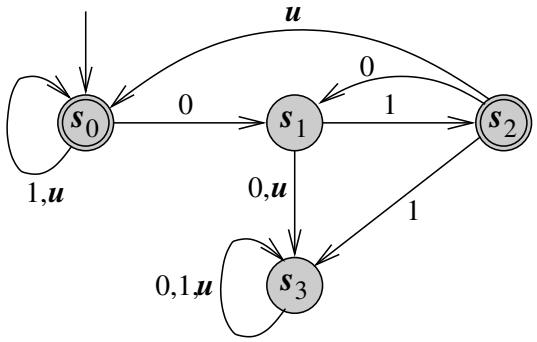
Als nächstes betrachten wir den Zustand s_1 mit den drei möglichen Eingaben 0, 1 und u. Mit der Eingabe 1 erreichen wir den Endzustand s_2, da die Folge 01 zur Sprache $L(EA)$ des Automaten EA gehört. Mit der Eingabe einer 0 oder eines u müssen wir in einen weiteren Zustand s_3 verzweigen, der aber kein Endzustand sein darf. Es kann sich bei dieser Eingabe nicht mehr um eine Folge aus 0 und 1 handeln, das Eingabewort kann somit nicht mehr zur Sprache $L(EA)$ des Automaten EA gehören.



Der Einfachheit halber betrachten wir den Zustand s_3, in dem entweder 00 oder $0u$ verarbeitet wurde, zuerst. Nach der Verarbeitung von 00 oder $0u$ kann kein Wort aus der Sprache $L(EA)$ mehr entstehen. Wir müssen in der Zukunft des Automaten mit beliebigen Eingaben (0, 1 oder u) unbedingt stets in dem Zustand s_3 verbleiben.

Betrachten wir nun die drei Eingaben 0, 1 und u im Zustand s_2. Im Zustand s_2 waren alle bisher gelesenen Zeichen, ausgenommen führender Zeichen vor der ersten 0, Folgen aus 0 und 1 und damit Teil von $L(EA)$.

Mit einer weiteren 0 wird das nächste Zeichen eines Wortes der Sprache $L(EA)$ gelesen. Der Automat kommt in den Zustand s_1 zurück, wo dann die nächste 1 erwartet wird, so dass der Automat das eingegebene Wort akzeptieren kann.
Mit einem u wird eine 0/1-Folge nicht unterbrochen, so dass damit ebenso das nächste Zeichen eines Wortes der Sprache $L(EA)$ gelesen wird. Der Automat kommt aber in den Zustand s_0 zurück, wo dann die nächste 0 erwartet wird.
Mit einer 1 wird die Folge von 0 und 1 jedoch durch eine zweite 1 unterbrochen, sodass es sich bei dieser Eingabe nicht mehr um eine Folge aus 0 und 1 handeln kann, das Eingabewort gehört nicht mehr zur Sprache $L(EA)$ des Automaten EA. Der Automat kommt in den Zustand s_3.
Damit sind alle Zustände sowie alle Transitionen bearbeitet, der vollständige Automat lautet also: EA$=(S,E,\delta,s_0,F)$ mit $S=\{s_0,s_1,s_2,s_3\}$, $E=\{0,1,u\}$, $F=\{s_0,s_2\}$ und der Übergangsfunktion δ.



δ	0	1	u
s_0	s_1	s_0	s_0
s_1	s_3	s_2	s_3
s_2	s_1	s_3	s_0
s_3	s_3	s_3	s_3

□

Wir haben bereits erwähnt, dass die von einem Automaten akzeptierte Sprache $L(EA)$ auch als reguläre Sprache bezeichnet wird. Damit kann man natürlich die Frage verbinden, ob es auch Sprachen gibt, die nicht Sprache eines endlichen Automaten sind. Uns interessiert insbesondere, woran wir diese Sprachen erkennnen können. Oder genauer gesagt, wo unterscheiden sich Sprachen, die von endlichen Automaten verstanden werden strukturell von denen, die diese Eigenschaft nicht aufweisen. Die Antwort auf diese Frage ist im Zusammenhang mit den regulären Ausdrücken bzw. regulären Sprachen zu klären. Dieser Thematik wollen wir in den nächsten beiden Abschnitten nachgehen.

4.4 Endliche Automaten und reguläre Sprachen

Das Ziel ist es, eine formale mathematische Beschreibungsmöglichkeit anzugeben, mit der wir die Sprache eines endlichen Automaten, die wir bisher in intuitiver Weise in Form der Eigenschaften der Wortmenge charakterisiert haben, elegant und präzise spezifizieren können. Reguläre Ausdrücke erlauben die kompakte Beschreibung von regulären Sprachen. Da es sich bei der von einem endlichen Automaten akzeptierten Sprache um eine reguläre Sprache handelt, ist es naheliegend, die Sprache eines endlichen Automaten durch reguläre Ausdrücke zu beschreiben. Zu diesem Zweck wollen wir den Zusammenhang zwischen einem regulären Ausdruck und der dazugehörigen regulären Sprache definieren.

4.4.1 Konstruktionselemente regulärer Ausdrücke

Die Beschreibung einer formalen Sprache basiert auf einfachen Operationen der Mengenlehre wie der Vereinigung, dem Produkt und der Iteration, die ausreichend sind, um die Sprachen endlicher Automaten vollständig zu beschreiben. Bevor wir die Definition einer regulären Sprache aufschreiben, geben wir die drei benötigten Mengenoperationen noch einmal an. Sind A und B Mengen, dann bezeichnen

$$\begin{aligned} A \cup B &= \{x \mid x \in A \text{ oder } x \in B\} && \text{die } \textit{Vereinigung} \text{ von } A \text{ und } B, \\ A.B &= \{xy \mid x \in A \text{ und } y \in B\} && \text{das } \textit{Produkt} \text{ von } A \text{ und } B, \\ A^* &= A^0 \cup A^1 \cup A^2 \cup A^3 \cup \ldots && \text{die } \textit{Iteration} \text{ von } A. \end{aligned}$$

Während bei Vereinigung und Produktbildung aus zwei endlichen Mengen eine neue endliche Menge gebildet wird, entsteht bei der Iteration aus einer endlichen Menge eine neue unendliche Menge.

Beispiel 4.19 *Regulärer Ausdruck*
Der reguläre Ausdruck $\boldsymbol{ab}^* + \boldsymbol{ba}^*$ beschreibt die Sprache, die aus allen Zeichenreihen besteht, die sich aus einem einzelnen führenden a und einer beliebigen Anzahl von nachfolgenden b oder aus einem einzelnen führenden b und einer beliebigen Anzahl von nachfolgenden a zusammensetzen.

Der reguläre Ausdruck $(\boldsymbol{ab})^* + (\boldsymbol{ba})^*$ beschreibt hingegen die Sprache, die aus allen Zeichenreihen besteht, die sich aus einer beliebigen Anzahl von ab-Folgen oder aus einer beliebigen Anzahl von ba-Folgen zusammensetzen. □

Bevor wir die Bildung regulärer Ausdrücke definieren, wollen wir uns mit den dabei verwendeten drei Operatoren noch etwas besser vertraut machen:

1. Die *Vereinigung* zweier Sprachen L_1 und L_2, beschrieben durch $L_1 \cup L_2$, ist die Menge aller Zeichenreihen, die entweder in L_1 oder L_2 oder in beiden Sprachen enthalten sind. Ist beispielsweise $L_1 = \{001, 10, 111\}$ und $L_2 = \{\varepsilon, 001\}$, dann gilt
   $L_1 \cup L_2 = \{\varepsilon, 001, 10, 111\}$.
2. Das *Produkt*, auch als *Verkettung* oder Konkatenation bezeichnet, zweier Sprachen L_1 und L_2 ist die Menge aller Zeichenreihen, die gebildet werden, indem eine Zeichenreihe aus L_1 mit einer beliebigen Zeichenreihe aus L_2 verkettet wird. Die Verkettung wird entweder durch einen Punkt als $L_1.L_2$ beschrieben oder abkürzend einfach ohne Operator als L_1L_2 dargestellt. Ist beispielsweise $L_1 = \{001, 10, 111\}$ und $L_2 = \{\varepsilon, 001\}$, dann gilt $L_1.L_2$ oder einfach $L_1L_2 = \{001, 10, 111, 001001, 10001, 111001\}$. Die ersten drei Zeichenreihen von L_1L_2 sind die mit ε verketteten aus L_1. Da ε die Identität der Verkettung ist ($\varepsilon w = w\varepsilon = w$), entsprechen die resultierenden Zeichenreihen den Zeichenreihen aus L_1. Die weiteren drei Zeichenreihen werden gebildet, indem die einzelnen Zeichenreihen von L_1 mit der zweiten Zeichenreihe (001) verkettet werden.
3. Die *Iteration*, auch als *Sternoperator* oder als *(Kleenesche) Hülle* bezeichnet, einer Sprache L wird durch L^* beschrieben und ist die Menge aller Zeichenreihen, die durch die Verkettung

einer beliebigen Anzahl von Zeichenreihen aus L gebildet werden. In dieser Menge sind auch alle Wiederholungen enthalten, d.h. diesselbe Zeichenreihe kann mehrmals verwendet werden.
Ist beispielsweise $L = \{0,1\}$, dann gilt für
$L^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, ...\}$.
L^* enthält alle möglichen Zeihenreihen aus 0 und 1.
Ist beispielsweise $L = \{0,11\}$, dann gilt für
$L^* = \{\varepsilon, 0, 11, 00, 011, 110, 1111, 000, 0011, 0110, 01111, 1100, ...\}$.
L^* enthält alle möglichen Zeihenreihen aus 0 und 11, die Einsen müssen immer paarweise auftreten. L^* enthält also z.B. 011, 11110 und ε, nicht jedoch 01011 oder 101. Mathematisch ausgedrückt ist L^* die unendliche Vereinigung

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^+ = L^0 \cup L^1 \cup L^2 \cup \ldots$$

mit $L^0 = \{\varepsilon\}$, $L^1 = L$, $L^2 = LL$, $L^i = L^{i-1}L$.

Beispiel 4.20 *Berechnung der Iteration*
Gegeben sei die Sprache $L = \{000, 1\}$. Man bestimme die Sprache L^*.

Um die oben angegebene Formel anzuwenden, müssen wir zunächst die einzelnen L^i für alle i berechnen. Unabhängig davon, um was für eine Sprache es sich bei L handelt, gilt stets $L^0 = \{\varepsilon\}$. L^0 repräsentiert die Auswahl von null Zeichenreihen aus L. $L^1 = L$ steht für die Auswahl einer einzelnen Zeichenreihe aus L, also gilt $L^1 = \{000, 1\}$.
Betrachten wir als nächstes L^2. Da L aus zwei Zeichenreihen besteht, gibt es $2^2 = 4$ Kombinationsmöglichkeiten, die erste verkettet mit sich selbst, die erste verkettet mit der zweiten, die zweite verkettet mit der ersten und die zweite verkettet mit sich selbst, also
$L^2 = \{000000, 0001, 1000, 11\}$.
Entsprechend gibt es bei L^3 $2^3 = 8$ Kombinationsmöglichkeiten, also
$L^3 = \{000000000, 0000001, 0001000, 00011, 1000000, 10001, 11000, 111\}$.
Zur Berechnung von L^* müssen wir alle L^i berechnen und anschließend alle diese Sprachen vereinigen. Obwohl jede Sprache L^i endlich ist, ergibt die Vereinigung der unendlichen Anzahl von Mengen L^i eine unendliche Sprache. □

Beispiel 4.21 *Berechnung der Iteration*
Gegeben sei die Sprache L, die aus der Menge aller aus 0 aufgebauten Zeichenreichen besteht, $L = \{0, 00, 000, 0000, 0000, ...\}$.

Man erkennt sofort, dass die Sprache L selbst bereits unendlich ist. Trotzdem kann die Sprache L^* leicht ermittelt werden. Es gilt natürlich $L^0 = \{\varepsilon\}$ und $L^1 = L = \{0, 00, 000, 0000, 0000, ...\}$.
L^2 ist die Menge der Zeichenreihen, die dadurch gebildet werden, indem man eine Zeichenreihe aus Nullen mit einer anderen Zeichenreihe aus Nullen verkettet, das Ergbnis ist ebenfalls eine Zeichenreihe aus Nullen. Jede Zeichenreihe aus Nullen kann als Verkettung zweier Zeichenreihen aus Nullen dargestellt werden. Folglich gilt $L^2 = L$, $L^3 = L$ usw. In diesem speziellen Fall, in dem die Sprache L die Menge aller aus Nullen bestehenden Zeichenreihen umfasst, gilt für die unendliche Vereinigung $L^0 \cup L^1 \cup L^2 \cup \ldots = L \cup \{\varepsilon\}$. □

Beispiel 4.22 *Berechnung der Iteration*
Gegeben sei die Sprache $L = \emptyset$. Naürlich gilt weiterhin $L^0 = \{\varepsilon\}$. Da aber die Menge L leer ist und wir aus einer leeren Menge keine Zeichenreihe auswählen können, sind alle anderen Menge L^i für alle i auch leer. Damit gilt $L^* = \{\varepsilon\}$, $L = \emptyset$ stellt als Ausnahme eine Sprache dar, deren Hülle nicht unendlich ist. □

Aufgabe 4.7 *Berechnung der Iteration*
Gegeben sei die Sprache $L = \{0, 11\}$. Bestimmen Sie die Sprache L^*. ◇

4.4.2 Reguläre Ausdrücke und reguläre Sprachen

In der nun folgenden Definition wird zunächst rein syntaktisch die Menge der sog. regulären Ausdrücke in den Punkten (1) und (2) festgelegt. Diese gehen von einem Alphabet E aus, aus dem durch Anwendung vorgegebener Regeln Zeichenketten $w \in E^*$ gebildet werden können.
Dieses Alphabet E wird in den Punkten (3), (4) und (5) um Mengenoperationen zur Vereinigung, Produktbildung (Verkettung oder Konkatenation) und Iteration (Hülle) erweitert, deren Anwendung auf reguläre Sprachen als Ergebnis wieder reguläre Sprachen sind.

Definition 4.8 *Regulärer Ausdruck und reguläre Sprache*
Es sei E ein Alphabet. Ein regulärer Ausdruck RA ist ein Ausdruck in einer bestimmten Form über der Zeichenmenge $U = E \cup \{\varepsilon, \emptyset, +, (.), *\}$. *Jeder reguläre Ausdruck RA legt eindeutig eine ihm zugeordnete Sprache* $L(RA) \subseteq E^*$ *fest. Jede solche Sprache* $L(RA)$ *wird als reguläre Sprache oder reguläre Menge bezeichnet. Reguläre Ausdrücke werden wie folgt gebildet:*

(1) Die Zeichen $\emptyset$ *und* ε *sind reguläre Ausdrücke,* $\emptyset, \varepsilon \in RA$.
Die dadurch beschriebenen Sprachen $L(RA)$ *sind* $L(\emptyset) = \{\ \}$ *bzw.* $L(\varepsilon) = \{\varepsilon\}$.

(2) Für jede Zeichenreihe $w \in E^*$ *ist w ein regulärer Ausdruck,* $w \in RA$.
Die durch diesen Ausdruck beschriebene Sprache $L(RA)$ *ist* $L(w) = \{w\}$.

(3) Sind A_1 *und* A_2 *reguläre Ausdrücke, dann ist auch die Vereinigung* $A_1 + A_2$ *ein regulärer Ausdruck.*
Die durch diesen Ausdruck beschriebene Sprache $L(RA) = L(A_1 + A_2)$ *ist die Vereinigung der Sprachen von* A_1 *und* A_2, $L(A_1 + A_2) = L(A_1) \cup L(A_2)$.

(4) Sind A_1 *und* A_2 *reguläre Ausdrücke, dann ist auch das Produkt* $A_1.A_2$ *ein regulärer Ausdruck.*
Die dadurch beschriebene Sprache $L(RA) = L(A_1.A_2)$ *ist die Verkettung der Sprachen von* A_1 *und* A_2, $L(A_1.A_2) = \{w_1 w_2 \mid w_1 \in L(A_1) \wedge w_2 \in L(A_2)\}$.

(5) Ist A ein regulärer Ausdruck, dann ist auch die Iteration A^* *ein regulärer Ausdruck.*
Die dadurch beschriebene Sprache $L(RA) = L(A^*)$ *ist die beliebige Hintereinanderausführung der Sprache von A,* $L(A^*) = (L(A))^*$. *Anders ausgedrückt,* $L(RA) = L(A^*)$ *ist die Sprache, die entsteht, wenn man die Wörter einer regulären Sprache beliebig miteinander konkateniert.*

Durch diese Definition haben wir nicht nur festgelegt, was zulässige reguläre Ausdrücke sind, sondern beschreiben für jeden regulären Ausdruck *RA* auch die von diesem Ausdruck repräsentierte Sprache *L(RA)*. Jedem regulären Ausdruck *RA* wird in eindeutiger Weise eine reguläre Sprache (oder auch reguläre Menge) *L(RA)* zugeordnet, die man durch geeignete Interpretation des regulären Ausdrucks erhält. Reguläre Sprachen sind also Mengen von Zeichenreihen (Wortmengen), die durch Anwendung von Mengenoperationen aus den Grundmengen $\emptyset$, $\{\varepsilon\}$ und $\{a\}$ für jedes $a \in E$ gebildet werden können. Dabei handelt es sich um eine rekursive Definition, bei denen aus Basisausdrücken mit den unter Punkt (1) bis (5) festgelegten Regeln neue Ausdrücke gebildet werden.

Die Definition unterscheidet zwischen dem syntaktischen Konstrukt, dem *regulären Ausdruck* und der Bedeutung dieses syntaktischen Konstruktes, der *regulären Sprache*. Das Zeichen $+$ beispielsweise ist in einem regulären Ausdruck der Form $A + B$ zunächst von rein syntaktischer Bedeutung. Erst durch die Interpretation der Ausdrücke A und B durch die Sprachen $L(A)$ und $L(B)$ und der anschließenden Vereinigung beider Sprachen erhält $+$ die Bedeutung der mengentheoretischen Vereinigung $\cup$. Oft wird eine Sprache einfach durch ihren regulären Ausdruck *RA* bezeichnet, obwohl *L(RA)* gemeint ist. Wir werden davon auch Gebrauch machen, wenn aus dem Zusammenhang klar ist was gemeint ist.

Beispiel 4.23 *Beschreibung einer Zeichenreihe durch einen regulären Ausdruck*
Gegeben sei die Menge der Zeichenreihen, die aus in abwechselnder Reihenfolge auftretenden a und b besteht, $L = \{\varepsilon, ab, ba, aba, bab, abab, baba, ababa, babab, ababab, ...\}$. Gesucht ist ein regulärer Ausdruck zur Beschreibung dieser Sprache.

Da die gegebene Sprache aus *ab*-Elementen aufgebaut ist, entwickeln wir zunächst einen regulären

Ausdruck für die Sprache, die lediglich die Zeichenreihe *ab* beinhaltet. Danach versuchen wir mit Hilfe der Iteration einen Ausdruck zu bilden, der für alle Zeichenreihen der Form *abab...ab* steht. Aus der Regel (2) der obigen Definition folgt, dass $\boldsymbol{a}$ und $\boldsymbol{b}$ zwei reguläre Ausdrücke A_1 und A_2 sind, welche die Sprachen $L(A_1) = \{a\}$ und $L(A_2) = \{b\}$ repräsentieren. Durch eine Verkettung $L(A_1.A_2)$ der beiden Ausdrücke erhalten wir einen regulären Ausdruck für die Sprache $\{ab\}$, $L(A_1.A_2) = \boldsymbol{ab}$.
Um alle Zeichenreihen zu beschreiben, die aus null oder mehr *ab*-Elementen bestehen, verwenden wir den regulären Ausdruck $(\boldsymbol{ab})^*$. Dabei haben wir $\boldsymbol{ab}$ in Klammern gesetzt, um diesen Ausdruck von $\boldsymbol{ab}^*$ zu unterscheiden. Dieser beschreibt hingegen die Menge aller Zeichenreihen, die sich aus einem führenden *a* gefolgt von einer beliebigen Anzahl *b* zusammensetzen.
Aber auch $L[(\boldsymbol{ab})^*]$ beschreibt noch nicht die Sprache $L = \{\varepsilon, ab, ba, aba, bab, abab, baba, ababa, ...\}$. $L[(\boldsymbol{ab})^*]$ umfasst nur Zeichenreihen aus *a* und *b*, die mit *a* beginnen und mit *b* enden. Diese Einschränkung wurde aber nicht gemacht, wir müssen also in unserer Beschreibung auch alle anderen Fälle mit abdecken. Die Zeichenreihe kann auch mit *b* beginnen und mit *a* enden. Der Ausdruck $(\boldsymbol{ab})^*$ beschreibt alle Zeichenreihen, die aus null oder mehr *ab*-Elementen bestehen. Wir fügen dieser Sprache alle Zeichenreihen hinzu, die aus null oder mehr *ba*-Elementen bestehen, $L[(\boldsymbol{ba})^*]$.
Weiter umfasst $L = \{\varepsilon, ab, ba, aba, bab, abab, baba, ababa, ...\}$ auch noch *ab*-Zeichenreihen, die mit einem *a* beginnen und mit *a* enden bzw. mit *b* beginnen und mit *b* enden. Diese werden durch $\boldsymbol{a}(\boldsymbol{ba})^*$ bzw. $\boldsymbol{b}(\boldsymbol{ab})^*$ beschrieben. Der gesamte reguläre Ausdruck lautet demnach:

$$(\boldsymbol{ab})^* + (\boldsymbol{ba})^* + \boldsymbol{a}(\boldsymbol{ba})^* + \boldsymbol{b}(\boldsymbol{ab})^*$$

Eine andere Lösung zur Beschreibung der Sprache $L = \{\varepsilon, ab, ba, aba, bab, abab, baba, ababa, ...\}$ funktioniert mit Hilfe des regulären Ausdrucks ε. Wir verwenden weiterhin den Ausdruck $(\boldsymbol{ab})^*$ zur Beschreibung aller *ab*-Zeichenreihen. Um am Anfang und am Ende der *ab*-Zeichenreihen ein optionales *a* oder *b* hinzuzufügen, versuchen wir dafür einen Ausdruck mit Hilfe der leeren Zeichereihe ε zu erzeugen um diesen mit den *ab*-Zeichenreihen zu verketten.
Vereingt man ein *a* oder ein *b* mit der leeren Zeichereihe ε zu $\varepsilon + \boldsymbol{a}$ bzw. $\varepsilon + \boldsymbol{b}$, dann folgt für die Sprache $L[\varepsilon + \boldsymbol{a}] = L[\varepsilon] \cup L[\boldsymbol{a}] = \{\varepsilon, a\}$ bzw. für $L[\varepsilon + \boldsymbol{b}] = \{\varepsilon, b\}$. Wenn wir nun die Sprache $L[\varepsilon + \boldsymbol{b}]$ mit $L[(\boldsymbol{ab})^*]$ zu $L[\varepsilon + \boldsymbol{b}].L[(\boldsymbol{ab})^*]$ verketten, dann wird dabei durch die Verkettung von $L[\varepsilon]$ mit $L[(\boldsymbol{ab})^*]$ die Menge aller *ab*-Zeichenreihen ohne führendes *a* und durch die Verkettung von $L[\boldsymbol{b}]$ mit $L[(\boldsymbol{ab})^*]$ die Menge aller *ab*-Zeichenreihen mit führendem *b* beschrieben. Dasselbe müssen wir entsprechend mit *a* am Ende der *ab*-Zeichenreihen tun. Wir erhalten somit alternativ folgenden Ausdruck für die Menge der Zeichenreihen, die aus abwechselnden *a* und *b* bestehen:

$$(\varepsilon + \boldsymbol{b})(\boldsymbol{ab})^*(\varepsilon + \boldsymbol{a})$$

□

Im obigen Beispiel haben wir die Sprache $\{ab\}$ durch den regulären Ausdruck $\boldsymbol{ab}$ beschrieben. Die Beschreibung einer Sprache, die nur aus einer Zeichenreihe *w* besteht, erfolgt in der Regel durch die Zeichenreihe selbst als regulärem Ausdruck. Um den regulären Ausdruck von der Zeichenreihe zu unterscheiden, haben wir diesen fett gedruckt.

Wie in jeder operationalen Algebra müssen auch hier für die Operatoren $+, (.), *$ der regulären Ausdrücke bestimmte Vorrangregeln für die Ausführungsreihenfolge der Operationen festgelegt werden. In der Arithmetik beispielsweise wird im Ausdruck $xy + z$ das Produkt xy vor der Addition berechnet, die Reihenfolge kann durch eine Klammerung $x(y + z)$ umgekehrt werden. Klammern binden grundsätzlich immer stärker als alle Operatoren, d.h. Operationen innerhalb von Klammern sind vor denen außerhalb von Klammern auszuführen.

Die Iteration * bindet stärker als die Produktbildung . und die Produktbildung wiederum bindet stärker als die Vereinigung $+$.

Der Sternoperator wird also immer auf die kleinstmögliche Sequenz von Symbolen auf seiner linken Seite angewandt, die einen wohlgeformten regulären Ausdruck darstellen. Nach den Sternoperationen werden die Verkettungen ihren Operanden zugeordnet. Da die Verkettung assoziativ ist, ist die Reihenfolge dabei gleichgültig. Zuletzt werden alle Vereinigungsoperatoren ihren Operanden zugeordnet, wobei die Vereinigung ebenfalls assoziativ ist. Ferner ist der Vereinigungsoperator distributiv, d.h. es gilt $(a + b).c = a.c + b.c$.

In den folgenden Beispielen wollen wir die Bestimmung der regulären Sprachen eines regulären Ausdrucks mit Hilfe der Regeln, die sich aus der Definition ergeben, praktisch üben.

Beispiel 4.24 *Sprache regulärer Ausdrücke*
Gegeben seien die folgenden regulären Ausdrücke RA über dem Alphabet $E = \{a,b,c,d\}$, die Produkt und Vereinigung enthalten.
Bestimmen Sie zu diesen Ausdrücken die reguläre Sprache $L(RA)$.

1)	A_1	$=$	$(\boldsymbol{a}+(\boldsymbol{b}.\boldsymbol{c}))$	4)	$A_4 = (\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{bc})$
2)	A_2	$=$	$\boldsymbol{a}+\boldsymbol{bc}$	5)	$A_5 = (\boldsymbol{abc})(\boldsymbol{abc})$
3)	A_3	$=$	$(\boldsymbol{a}+\boldsymbol{b})\boldsymbol{c}$	6)	$A_6 = (\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{d})$

1) Ausdruck $A_1 = (\boldsymbol{a}+(\boldsymbol{b}.\boldsymbol{c}))$

$$\begin{aligned} L[A_1] &= L[\boldsymbol{a}+(\boldsymbol{b}.\boldsymbol{c})] = L[\boldsymbol{a}] \cup L[(\boldsymbol{b}.\boldsymbol{c})] = L[\boldsymbol{a}] \cup L[\boldsymbol{b}]L[\boldsymbol{c}] \\ &= \{a\} \cup \{b\}\{c\} = \{a\} \cup \{bc\} = \{a, bc\} \end{aligned}$$

2) Ausdruck $A_2 = \boldsymbol{a}+\boldsymbol{bc}$

$$L[A_2] = L[\boldsymbol{a}+\boldsymbol{bc}] = L[\boldsymbol{a}] \cup L[(\boldsymbol{b}.\boldsymbol{c})] = L[\boldsymbol{a}] \cup L[\boldsymbol{b}]L[\boldsymbol{c}] = \{a, bc\}$$

3) Ausdruck $A_3 = (\boldsymbol{a}+\boldsymbol{b})\boldsymbol{c}$

$$\begin{aligned} L[A_3] &= L[(\boldsymbol{a}+\boldsymbol{b})\boldsymbol{c}] = L[(\boldsymbol{ac}+\boldsymbol{bc})] = L[(\boldsymbol{ac})] \cup L[(\boldsymbol{bc})] = L[\boldsymbol{a}]L[\boldsymbol{c}] \cup L[\boldsymbol{b}]L[\boldsymbol{c}] \\ &= \{a\}\{c\} \cup \{b\}\{c\} = \{ac\} \cup \{bc\} = \{ac, bc\} \end{aligned}$$

4) Ausdruck $A_4 = (\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{bc})$

$$\begin{aligned} L[A_4] &= L[(\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{bc})] = L[(\boldsymbol{abc}+\boldsymbol{bbc})] = L[(\boldsymbol{abc})] \cup L[(\boldsymbol{bbc})] \\ &= L[\boldsymbol{a}]L[\boldsymbol{b}]L[\boldsymbol{c}] \cup L[\boldsymbol{b}]L[\boldsymbol{b}]L[\boldsymbol{c}] \\ &= \{a\}\{b\}\{c\} \cup \{b\}\{b\}\{c\} = \{abc\} \cup \{bbc\} = \{abc, bbc\} \end{aligned}$$

alternative Auflösung

$$\begin{aligned} L[A_4] &= L[(\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{bc})] = L[(\boldsymbol{a}+\boldsymbol{b})]L[(\boldsymbol{bc})] = (L[\boldsymbol{a}] \cup L[\boldsymbol{b}])L[\boldsymbol{b}]L[\boldsymbol{c}] \\ &= (\{a\} \cup \{b\})\{b\}\{c\} = \{a\}\{b\}\{c\} \cup \{b\}\{b\}\{c\} = \{abc\} \cup \{bbc\} = \{abc, bbc\} \end{aligned}$$

5) Ausdruck $A_5 = (\boldsymbol{abc})(\boldsymbol{abc})$

$$\begin{aligned} L[A_5] &= L[(\boldsymbol{abc})(\boldsymbol{abc})] = L[\boldsymbol{abc}]L[\boldsymbol{abc}] = (L[\boldsymbol{a}]L[\boldsymbol{b}]L[\boldsymbol{c}])(L[\boldsymbol{a}]L[\boldsymbol{b}]L[\boldsymbol{c}]) \\ &= (\{a\}\{b\}\{c\})\{a\}\{b\}\{c\} = \{abc\}\{abc\} = \{abcabc\} \end{aligned}$$

6) Ausdruck $A_6 = (\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{d})$

$$\begin{aligned} L[A_6] &= L[(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{d})] = L[(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})]L[(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{d})] \\ &= (L[\boldsymbol{a}] \cup L[\boldsymbol{b}] \cup L[\boldsymbol{c}])(L[\boldsymbol{a}] \cup L[\boldsymbol{b}] \cup L[\boldsymbol{d}]) \\ &= (\{a\} \cup \{b\} \cup \{c\})(\{a\} \cup \{b\} \cup \{d\}) = \{a,b,c\}\{a,b,d\} \\ &= \{aa, ab, ad, ba, bb, bd, ca, cb, cd\} \end{aligned}$$

□

Die eckigen Klammern dienen nur der besseren Lesbarkeit, haben also keine besondere Bedeutung. Ausdrücke, die die Operationen Vereinigung und Produkt enthalten, sind relativ leicht auszuwerten. Zu beachten ist hierbei lediglich die Priorität der Operationen sowie die Klammerung. Etwas schwieriger sind hingegen Ausdrücke auszuwerten, welche die Iteration enthalten, da diese eine unendliche Sprachmenge erzeugen.

Beispiel 4.25 *Sprache regulärer Ausdrücke*
Gegeben seien die folgenden regulären Ausdrücke RA über dem Alphabet $E = \{a,b,c,u,v,w\}$, die Produkt, Vereinigung und Iteration enthalten.
Bestimmen Sie zu diesen Ausdrücken die reguläre Sprache $L(RA)$.

1) $A_1 = ((\boldsymbol{a})^*)\boldsymbol{b}$
2) $A_2 = (\boldsymbol{ab})^*$
3) $A_3 = ((\boldsymbol{a})^*)+\boldsymbol{b}$
4) $A_4 = (\boldsymbol{a}+\boldsymbol{b})^*$
5) $A_5 = (\boldsymbol{abc})^*$
6) $A_6 = \boldsymbol{a}^*+\boldsymbol{b}^*+\boldsymbol{c}^*$
7) $A_7 = (\boldsymbol{a}+\boldsymbol{b})^*+(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})^*$

1) Ausdruck $A_1 = ((\boldsymbol{a})^*)\boldsymbol{b}$

$$\begin{aligned} L[A_1] &= L[((\boldsymbol{a})^*)\boldsymbol{b}] = L[((\boldsymbol{a})^*).\boldsymbol{b}] = L[((\boldsymbol{a})^*)]\,L[\boldsymbol{b}] = (L[(\boldsymbol{a})])^*\,L[\boldsymbol{b}] \\ &= \{a\}^*\{b\} = \{\varepsilon, a, aa, aaa, aaaa, ...\}\{b\} \\ &= \{b, ab, aab, aaab, aaaab, ...\} = \{a^n b \mid n \in \mathbb{N}_0\} \end{aligned}$$

2) Ausdruck $A_2 = (\boldsymbol{ab})^*$

$$\begin{aligned} L[A_2] &= L[(\boldsymbol{ab})^*] = (L[(\boldsymbol{ab})])^* = (L[\boldsymbol{a}]\,L[\boldsymbol{b}])^* \\ &= (\{a\}\{b\})^* = (\{ab\})^* = \{ab\}^* \\ &= \{\varepsilon, ab, abab, ababab, abababab, ...\} = \{(ab)^n \mid n \in \mathbb{N}_0\} \end{aligned}$$

3) Ausdruck $A_3 = ((\boldsymbol{a})^*)+\boldsymbol{b}$

$$\begin{aligned} L[A_3] &= L[((\boldsymbol{a})^*)+\boldsymbol{b}] = L[((\boldsymbol{a})^*)] \cup L[\boldsymbol{b}] = (L[(\boldsymbol{a})])^* \cup L[\boldsymbol{b}] \\ &= \{a\}^* \cup \{b\} = \{\varepsilon, a, aa, aaa, aaaa, ...\} \cup \{b\} \\ &= \{\varepsilon, a, aa, aaa, aaaa, ..., b\} \end{aligned}$$

4) Ausdruck $A_4 = (\boldsymbol{a}+\boldsymbol{b})^*$

$$\begin{aligned} L[A_4] &= L[(\boldsymbol{a}+\boldsymbol{b})^*] = (L[(\boldsymbol{a}+\boldsymbol{b})])^* = ([L[\boldsymbol{a}] \cup L[\boldsymbol{b}])^* \\ &= (\{a\} \cup \{b\})^* = \{a, b\}^* \\ &= \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, aaaa, ...\} \end{aligned}$$

5) Ausdruck $A_5 = (\boldsymbol{abc})^*$

$$\begin{aligned} L[A_5] &= L[(\boldsymbol{abc})^*] = (L[(\boldsymbol{abc})])^* = (L[\boldsymbol{a}]\,L[\boldsymbol{b}]\,L[\boldsymbol{c}])^* \\ &= (\{a\}\{b\}\{c\})^* = (\{abc\})^* \\ &= \{abc\}^* = \{\varepsilon, abc, abcabc, abcabcabc, abcabcabcabc, ...\} = \{(abc)^n \mid n \in \mathbb{N}_0\} \end{aligned}$$

6) Ausdruck $A_6 = \boldsymbol{a}^*+\boldsymbol{b}^*+\boldsymbol{c}^*$

$$\begin{aligned} L[A_6] &= L[\boldsymbol{a}^*+\boldsymbol{b}^*+\boldsymbol{c}^*] = L[\boldsymbol{a}^*] \cup L[\boldsymbol{b}^*] \cup L[\boldsymbol{c}^*] = (L[\boldsymbol{a}])^* \cup (L[\boldsymbol{b}])^* \cup (L[\boldsymbol{c}])^* \\ &= \{a\}^* \cup \{b\}^* \cup \{c\}^* = \{\varepsilon, a, aa, aaa, ...\} \cup \{\varepsilon, b, bb, bbb, ...\} \cup \{\varepsilon, c, cc, ccc, ...\} \\ &= \{\varepsilon, a, b, c, aa, bb, cc, aaa, bbb, ccc, aaaa, bbbb, cccc, ...\} \end{aligned}$$

7) Ausdruck $A_7 = (\boldsymbol{a}+\boldsymbol{b})^*+(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})^*$

$$\begin{aligned} L[A_7] &= L[(\boldsymbol{a}+\boldsymbol{b})^*+(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})^*] \\ &= L[(\boldsymbol{a}+\boldsymbol{b})^*] \cup L[(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})^*] \\ &= L[(\boldsymbol{a}+\boldsymbol{b})^*] \cup (L[(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})]\,L[(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})^*]) \\ &= (L[(\boldsymbol{a}+\boldsymbol{b})])^* \cup (L[(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})]\,(L[(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})])^*) \\ &= (\{a\} \cup \{b\})^* \cup ((\{a\} \cup \{b\} \cup \{c\})\,(\{u\} \cup \{v\} \cup \{w\})^*) \\ &= (\{a, b\})^* \cup ((\{a, b, c\})\,(\{u, v, w\})^*]) = \{a, b\}^* \cup (\{a, b, c\}\,\{u, v, w\}^*) \\ &= \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, aaaa, ...\} \\ &\quad \cup (\{a, b, c\}\,\{\varepsilon, u, v, w, uu, uv, uw, vu, vv, vw, wu, wv, ww, \\ &\qquad uuu, uuv, uuw, uvu, uvv, uvw, uwu, uwv, uww, \\ &\qquad vuu, vuv, vuw, vvu, vvv, vvw, vwu, vuw, vww, \\ &\qquad wuu, wuv, wuw, wvu, wvv, wvw, wwu, wwv, www, uuuu, ...\}) \\ &= \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, aaaa, ...\} \\ &\quad \cup (\{a\}\,\{\varepsilon, u, v, w, uu, uv, uw, vu, vv, vw, wu, wv, ww, \\ &\qquad uuu, uuv, uuw, uvu, uvv, uvw, uwu, uwv, uww, ...\} \\ &\quad \cup \{b\}\,\{\varepsilon, u, v, w, uu, uv, uw, vu, vv, vw, wu, wv, ww, \\ &\qquad uuu, uuv, uuw, uvu, uvv, uvw, uwu, uwv, uww, ...\} \\ &\quad \cup \{c\}\,\{\varepsilon, u, v, w, uu, uv, uw, vu, vv, vw, wu, wv, ww, \\ &\qquad uuu, uuv, uuw, uvu, uvv, uvw, uwu, uwv, uww, ...\}) \end{aligned}$$

□

Aufgabe 4.8 *Sprache regulärer Ausdrücke*
Gegeben seien die folgenden regulären Ausdrücke RA über dem Alphabet $E = \{a,b\}$, die Produkt, Vereinigung und Iteration enthalten.
Bestimmen Sie zu diesen Ausdrücken die reguläre Sprache $L(RA)$.

1) $A_1 = (\boldsymbol{ab})^* + (\boldsymbol{ab})^*$ 2) $A_2 = (\boldsymbol{a}+\varepsilon)(\boldsymbol{ab})^*$ ◇

4.4.3 Endliche Automaten und reguläre Ausdrücke

Obwohl der in regulären Ausdrücken (RA) verwendete Ansatz zur Beschreibung von Sprachen sich grundlegend vom dem endlicher Automaten (EA) unterscheidet, repräsentieren die beiden Notationen die gleiche Menge von Sprachen, die als reguläre Sprachen bezeichnet werden. Um diese Aussage zu beweisen, müssten wir zeigen, dass jede Sprache, die durch einen EA definiert wird, auch durch einen bestimmten RA definiert wird und umgekehrt. Dieser formale Beweis ist jedoch sehr komplex und aufwändig, bringt aber wenig für unsere eher informellen Betrachtungen. Aufgrund dessen wollen wir darauf verzichten und anstatt dessen mehr den Zusammenhang zwischen einem endlichen Automaten und der von ihm erkannten Sprache genauer kennenlernen, indem wir uns auf die wesentlichen Aussagen und das Konzept beschränken und dieses durch intuitive Erklärungen sowie an mehreren Beispielen vertiefen.

Gehen wir mal davon aus, dass wir die Sprache $L(EA)$ eines endlichen Automaten EA durch einen regulären Ausdruck RA kompakt beschreiben können. Dann stellt sich unmittelbar die Frage, wie wir von einem endlichen Automaten zu einem regulären Ausdruck kommen, der genau die von diesem Automaten akzeptierte Sprache beschreibt. Wir müssten wir dazu Ausdrücke formulieren, die Mengen von Zeichenreihen beschreiben, die als Kantenbeschriftung beliebiger Pfade im Zustandsdiagramm möglich sind. Die Sprache eines EA in Form eines regulären Ausdrucks zu beschreiben ist eines der Ziele dieses Abschnitts.

Wir wollen aber zuerst den umgekehrten Weg betrachten, die Bestimmung eines endlichen Automaten EA zu einem gegebenen regulären Ausdruck RA. Diese Umwandlung basiert auf einer strukturellen Analyse des regulären Ausdrucks RA. Wir zeigen zuerst wie der EA (Grundautomat) für die Anfangsausdrücke ε, $\emptyset$ und a konstruiert wird. Anschließend zeigen wir, wie diese Grundautomaten zu größeren EAs zusammengesetzt werden, welche die Iteration, Vereinigung und Verkettung der von den Grundautomaten akzeptierten Sprachen akzeptieren.

Beginn der Konstruktion

Die Konstruktion beginnt mit den Grundautomaten, die nur die Ausdrücke ε, $\emptyset$ und a akzeptieren. Die entsprechenden Automaten besitzen genau einen akzeptierenden Zustand, von dem aus keine weiteren Transitionen ausgehen und weisen keine Transitionen zum Startzustand auf.

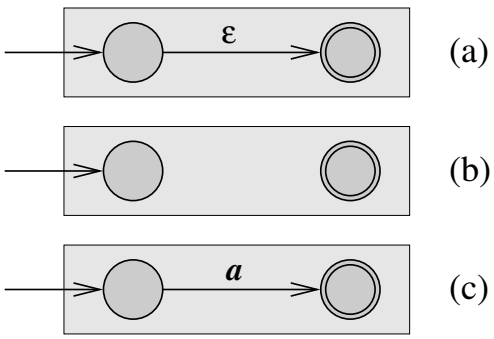


Abbildung 4.6: Konstruktionsbeginn

Der Automat, der die Sprache ε akzeptiert, besitzt einen einzigen Zustandsübergang von einem Startzustand in einen Endzustand mit der Markierung ε (Abb. 4.6(a)).

Der Automat, der die Sprache $\emptyset$ akzeptiert, besitzt keinen Zustandsübergang (Abb. 4.6(b)).
Der Automat, der die Sprache a akzeptiert, besitzt einen einzigen Zustandsübergang von einem Startzustand in einen Endzustand mit der Markierung a (Abb. 4.6(c)).

Umsetzung der Operatoren

(a) Vereinigung $A = A_1 + A_2$

Der Ausdruck $A_1 + A_2$ für die Vereinigung bedeutet für einen EA, dass dieser ausgehend von einem Anfangszustand s_0, in den es keine Transitionen gibt, in einen Startzustand für den Ausdruck A_1 **oder** in einen Startzustand für den Ausdruck A_2 kommen kann. Dabei handelt es sich um einen wechselseitigen Ausschluss, also eine echte Aufspaltung in zwei Pfade. Sobald durch eine ε-Transition die Entscheidung für einen der Pfad gefallen ist, ist der jeweils andere Pfad bedeutungslos, er kann auch später nicht mehr genommen werden.

Beide Pfade sind mit der jeweiligen Zeichenreihe markiert, es gibt einen Automaten $\mathrm{EA}_1 = (S_1, E_1, \delta_1, s_1, \{u_1\})$ und einen Automaten $\mathrm{EA}_2 = (S_2, E_2, \delta_2, s_2, \{u_2\})$ mit $L[\mathrm{EA}_1] = L[A_1]$ und $L[\mathrm{EA}_2] = L[A_2]$.
Dabei gehen von den beiden Endzuständen u_1 und u_2 keine Transitionen aus (Abb. 4.7).

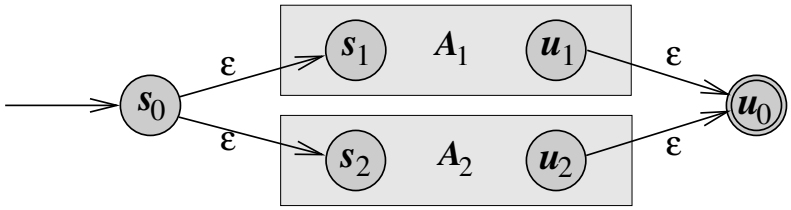


Abbildung 4.7: Konstruktion des Vereinigungsoperators

Nach einer Umbennung der Zustände können wir davon ausgehen, dass die beiden Zustandsmengen S_1 und S_2 disjunkt sind. Dann ist der Automat für die Vereinigung wie folgt definiert:

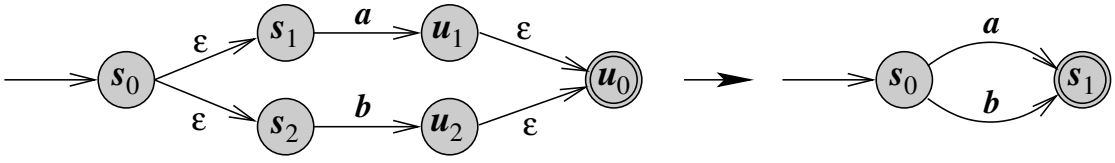
$$\mathrm{EA} = (S, E, \delta, s_0, \{u_0\})$$

wobei $S = S_1 \cup S_2 \cup \{s_0, u_0\}$ und $E = E_1 \cup E_2$. Die Übergangsfunktion δ ist gegeben durch:

$$\begin{aligned}
\delta(s_0, \varepsilon) &= \{s_1, s_2\} \\
\delta(s, a) &= \delta_1(s, a) \text{ für } s \in \{S_1 - \{u_1\}\} \text{ und } a \in \{E_1 \cup \varepsilon\} \\
\delta(s, a) &= \delta_1(s, a) \text{ für } s \in \{S_2 - \{u_2\}\} \text{ und } a \in \{E_2 \cup \varepsilon\} \\
\delta(u_1, \varepsilon) &= \delta(u_2, \varepsilon) = \{u_0\}
\end{aligned}$$

Da es keine Transitionen aus den beiden Endzuständen u_1 und u_2 gibt, sind alle Transitionssequenzen aus EA_1 und EA_2 in EA repräsentiert. Jeder Pfad von Zustand s_0 nach Zustand u_0 im Zustandsdiagramm von EA muss mit einer ε-Transition von s_0 nach s_1 oder s_2 beginnen. Wird die ε-Transition von s_0 nach s_1 genommen, dann folgt darauf eine beliebige Transitionssequenz in EA_1 zu Zustand u_1 und dann mit ε nach Zustand u_0. Analog folgt auf die ε-Transition nach s_2 eine beliebige Transitionssequenz in EA_2 zu Zustand u_2 und u_0. Es folgt unmittelbar, dass es genau dann in EA einen mit der Zeichenkette w markierten Pfad von s_0 nach u_0 gibt, wenn in EA_1 ein mit w markierter Pfad von s_1 nach u_1 oder in EA_2 ein ebensolcher von s_2 nach u_2 existiert. Demnach gilt für die Sprache dieses Automaten $L[A] = L[\mathrm{EA}] = L[A_1 + A_2] = L[A_1] \cup L[A_2]$.

Beispiel 4.26 *Beispiel zur Vereinigung*
Für den Ausdruck $A = \boldsymbol{a} + \boldsymbol{b}$ mit der Sprachmenge $L[A] = \{a, b\}$ folgt der EA mit folgender Übergangsfunktion:



Durch Elimination der verketteten ε-Transitionen mit der Regel $w.\varepsilon = \varepsilon.w = w$ und einer Umbenennung der Zustände erhält man den rechts daneben gezeigten EA. □

(b) Verkettung $A = A_1.A_2$
Der Ausdruck $A_1 + A_2$ für die Vereinigung bedeutet für einen EA, dass dieser ausgehend von einem Anfangszustand s_1 in EA_1 in einen Endzustand u_1 kommt und von da aus weiter in einem Anfangszustand s_2 in EA_2 und schließlich in einen Endzustand u_2. Dabei wird der Startzustand von A_1 zum Startzustand des EA und der oder die Endzustände von A_2 zu Endzuständen des EA. Alle Pfade die vom Startzustand des EA in einen akzeptierenden Endzustand führen, müssen zuerst einen Pfad im Automaten für A_1 passieren um dann über einen Pfad im Automaten für A_2 in einen Endzustand zu kommen (Abb. 4.8).

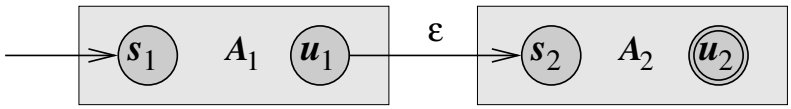


Abbildung 4.8: Konstruktion des Verkettungsoperators

Mit den beiden Automaten $\mathrm{EA}_1 = (S_1, E_1, \delta_1, s_1, \{u_1\})$ und $\mathrm{EA}_2 = (S_2, E_2, \delta_2, s_2, \{u_2\})$
mit $L[\mathrm{EA}_1] = L[A_1]$ und $L[\mathrm{EA}_2] = L[A_2]$ sowie disjunkten Zustandsmengen S_1 und S_2 ist der Automat für die Verkettung wie folgt definiert:

$$\mathrm{EA} = (S, E, \delta, s_0, \{u_0\})$$

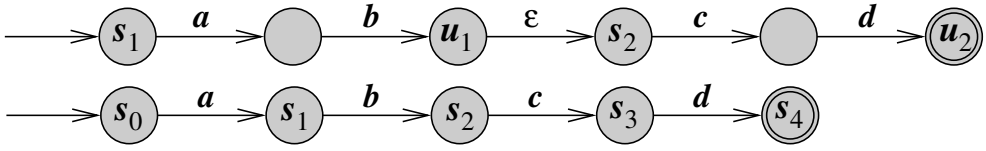
wobei $S = S_1 \cup S_2 \cup \{s_0, u_0\}$ und $E = E_1 \cup E_2$. Die Übergangsfunktion δ ist gegeben durch:

$$\begin{aligned} \delta(s,a) &= \delta_1(s,a) \text{ für } s \in \{S_1 - \{u_1\}\} \text{ und } a \in \{E_1 \cup \varepsilon\} \\ \delta(u_1,\varepsilon) &= \{s_2\} \\ \delta(s,a) &= \delta_2(s,a) \text{ für } s \in \{S_2\} \text{ und } a \in \{E_2 \cup \varepsilon\} \end{aligned}$$

Jeder Pfad von Zustand s_0 nach Zustand u_2 im Zustandsdiagramm von EA besteht aus einem Pfad von s_1 nach u_1, der mit der Zeichenkette w_1 markiert ist, gefolgt von einer ε-Transition von u_1 nach s_2, gefolgt von einem mit der Zeichenkette w_2 markierten Pfad von s_2 nach u_2. Für die Sprache dieses Automaten gilt also

$$\begin{aligned} L[A] &= L[\mathrm{EA}] = \{w_1.w_2 \mid w_1 \text{ ist Zeichenkette aus } L[\mathrm{EA}_1] \text{ und } w_2 \text{ ist Zeichenkette aus } L[\mathrm{EA}_2]\} \\ L[A] &= L[A_1.A_2] = \{w_1 w_2 \mid w_1 \in L[A_1] \wedge w_2 \in L[A_2]\} \\ L[A] &= L[A_1].L[A_2]. \end{aligned}$$

Beispiel 4.27 *Beispiel zur Verkettung*
Für den Ausdruck $A = \boldsymbol{ab.cd}$ mit der Sprachmenge $L[A] = \{abcd\}$ folgt der EA mit folgender Übergangsfunktion:



Durch Elimination der ε-Transitionen und Umbenennung erhält man den vereinfachten EA. □

(c) Iteration $A = A_1^*$
Der Ausdruck A_1^* für die Iteration kann für einen EA zwei Alternativen zulassen. Im ersten Fall kommt der EA direkt vom Startzustand über eine ε-Transition in den Endzustand. Im zweiten Fall passiert der EA mehrmals den Automaten für den Ausdruck A um danach in den Endzustand zu kommen (Abb. 4.9). Mit $\mathrm{EA}_1 = (S_1, E_1, \delta_1, s_1, \{u_1\})$ und $L[\mathrm{EA}_1] = L[A_1]$ ist der Automat für die Verkettung wie folgt definiert:

$$\mathrm{EA} = (S, E, \delta, s_0, \{u_0\})$$

wobei $S = S_1 \cup \{s_0, u_0\}$ und $E = E_1$. Die Übergangsfunktion δ ist gegeben durch:

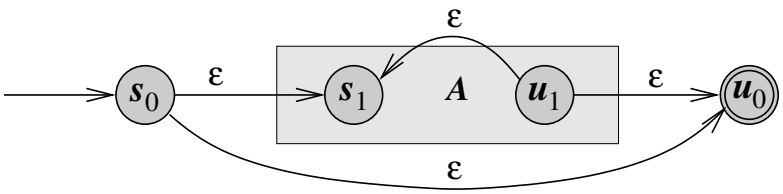


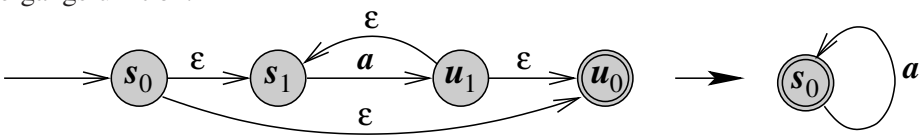
Abbildung 4.9: Konstruktion des Iterationsoperators

$$\begin{aligned} \delta(s_0,\varepsilon) &= \delta(u_1,\varepsilon) = \{s_1,u_0\} \\ \delta(s,a) &= \delta_1(s,a) \text{ für } s\in\{S_1-\{u_1\}\} \text{ und } a\in\{E_1\cup\varepsilon\} \end{aligned}$$

Jeder Pfad von Zustand s_0 nach Zustand u_0 im Zustandsdiagramm von EA besteht entweder aus einer direkten ε-Transition oder aus einem Pfad, der aus folgenden Komponenten besteht: einer ε-Transition von s_0 nach s_1, beliebig vielen (eventuell null) mit w_1-Elementen aus $L[A_1]$ markierten Transitionen von s_1 nach u_1 und mit ε zurück nach s_1, einer weiteren mit w_1 aus $L[A_1]$ markierten Transition von s_1 nach u_1, sowie einer ε-Transition von u_1 nach u_0. Also gibt es in EA einen mit w markierten Pfad von s_0 nach u_0, wenn $w = w_1w_2...w_i$ für ein $i\leq 0$ gilt ($i=0$ bedeutet $w=\varepsilon$), wobei jedes w_i aus $L[\mathrm{EA}_1]$ ist. Für die Sprache dieses Automaten gilt also

$$L[A] = L[\mathrm{EA}] = (L[\mathrm{EA}_1]^*) = (L[\mathrm{EA}_1])^* = (L[A_1])^*$$

Beispiel 4.28 *Beispiel zur Iteration*
Für den Ausdruck $A = \boldsymbol{a}^*$ mit der Sprachmenge $L[A] = \{\varepsilon, a, aa, aaa, aaab, ...\}$ folgt der EA mit folgender Übergangsfunktion:

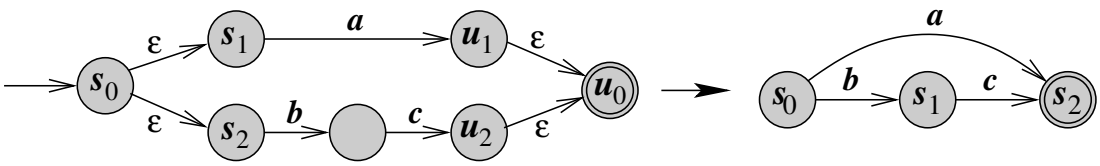


Mit Hilfe von Minimierungsregeln, die später noch eingeführt werden, sowie durch Elimination der ε-Transitionen und Umbenennung kommt man auf den daneben dargestellten vereinfachten EA. Es ist leicht zu überprüfen, dass die beiden EAs die gleiche Sprachmenge $L[A] = \{\varepsilon, a, aa, aaa, aaab, ...\}$ akzeptieren und daher verhaltensmäßig äquivalent sind. Wir werden aber im Verlauf dieses Kurses noch formale Regeln kennenlernen um diese Äquivalenz zu beweisen. □

Beispiel 4.29 *Regulärer Ausdruck und zugehöriger Automat*
Gegeben seien die folgenden Ausdrücke RA aus Beispiel 4.24 über dem Alphabet $E = \{a,b,c,d\}$, die Produkt und Vereinigung enthalten.
Gesucht sind endliche Automaten, welche die zu den folgenden regulären Ausdrücken gehörende reguläre Sprache $L[RA]$ besitzen.

1)	A_1	$= (\boldsymbol{a}+(\boldsymbol{b}.\boldsymbol{c}))$	4)	A_4	$= (\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{bc})$
2)	A_2	$= \boldsymbol{a}+\boldsymbol{bc}$	5)	A_5	$= (\boldsymbol{abc})(\boldsymbol{abc})$
3)	A_3	$= (\boldsymbol{a}+\boldsymbol{b})\boldsymbol{c}$	6)	A_6	$= (\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{d})$

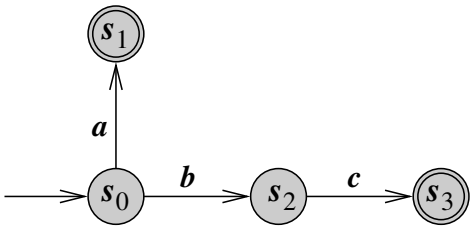
1) Ausdruck $A_1 = (\boldsymbol{a}+(\boldsymbol{b}.\boldsymbol{c}))$, $L[A_1] = \{a, bc\}$
Entsprechend den Regeln müssen wir den Ausdruck A_1 in zwei vereinigte Teilausdrücke aufspalten: $A_1 = R+S$, wobei $R = \boldsymbol{a}$ und $S = \boldsymbol{bc}$. Mit dem Konstrukt für die Vereinigung (Abb. 4.7) folgt dann:



Dabei haben wir die Verkettung von $b.c$ ohne dies besonders zu erwähnen gleich mit durchgeführt. ε-Transitionen vor oder nach regulären Transitionen können nach der Regel $\varepsilon w = w\varepsilon = w$ eliminiert

werden und es folgt der rechts danebenstehende Automat.

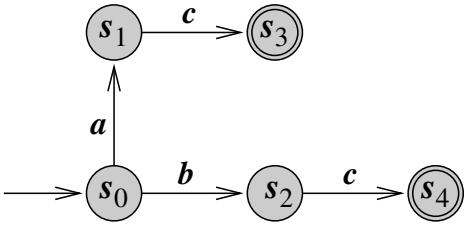
Man kann die Konstruktion dieses Automaten auch sehr leicht auf intuitiven Wege ausführen, wenn man sich das Verhalten eines Automaten zu diesem Ausdruck anschaulich vorstellt:
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a oder ein b gefolgt von einem c. Die Verarbeitung von a führt in einen Zustand s_1, der danach nicht mehr verlassen wird, die Verarbeitung von b führt in einen Zustand s_2, in dem ein c verarbeitet werden kann, woraufhin der Automat in einen weiteren Zustand s_3 gelangt, der ebenfalls nicht mehr verlassen werden kann. Es ergibt sich ein Automat mit der Zustandsmenge $S = \{s_0, s_1, s_2, s_3\}$, den Endzuständen $F = \{s_1, s_3\}$ und der folgenden Übergangsfunktion δ. Dabei können die beiden Endzustände s_1 und s_3 auch im Zustand s_1 oder s_3 zusammengelegt werden:

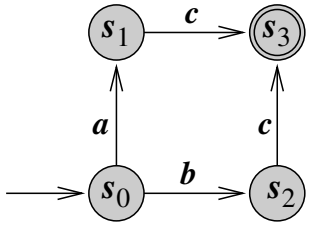


δ	a	b	c
s_0	s_1	s_2	–
s_1	–	–	–
s_2	–	–	s_3
s_3	–	–	–

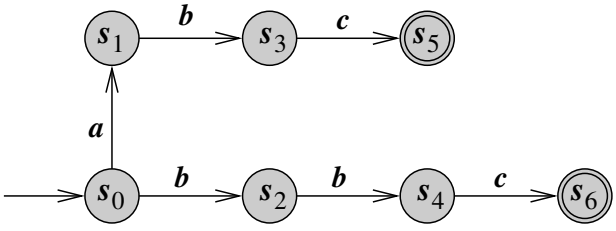
2) Ausdruck $A_2 = \boldsymbol{a} + \boldsymbol{bc}$, $L[A_2] = \{a, bc\}$
Der Ausdruck A_2 beschreibt die gleiche Sprache wie der Ausdruck A_1. Es ergibt sich der gleiche Automat.

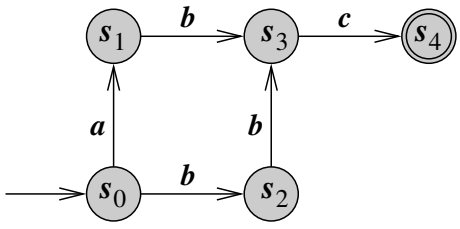
3) Ausdruck $A_3 = (\boldsymbol{a} + \boldsymbol{b})\boldsymbol{c}$, $L(A_3) = \{ac, bc\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a gefolgt von einem c oder ein b gefolgt von einem c. Man könnte den Ausdruck auch in der Form $A_3 = a.c + b.c$ schreiben. Die Punkte stehen sozusagen für die Zwischenzustände. Es gibt zwei äquivalente Lösungen, abhängig davon welcher Ausdruck umgesetzt wird.

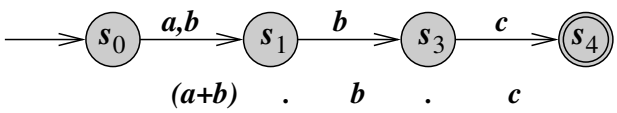




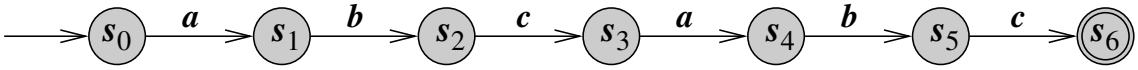
4) Ausdruck $A_4 = (\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{bc})$, $L[A_4] = \{abc, bbc\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a gefolgt von einem b und einem c oder ein b gefolgt von einem weiteren b und dann einem c. Man könnte den Ausdruck auch in der Form $A_3 = a.b.c + b.b.c$ schreiben. Es gibt wieder zwei äquivalente Lösungen, abhängig davon welcher Ausdruck umgesetzt wird. Dabei gibt es zur zweiten Lösung auch noch eine vereinfachende Darstellung.



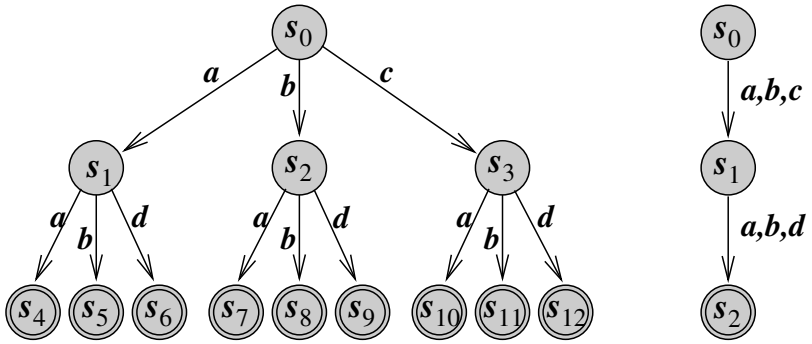




5) Ausdruck $A_5 = (\boldsymbol{abc})(\boldsymbol{abc})$, $L[A_5] = \{abcabc\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 nur ein a gefolgt von einem b und einem c. Im danach erreichten Zustand s_3 wird wieder nur ein a gefolgt von einem b und einem c akzeptiert. Der Ausdruck kann auch in der Form $A_5 = a.b.c.a.b.c$ geschrieben werden.



6) Ausdruck $A_6 = (\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{d})$, $L[A_6] = \{a,b,c\}\{a,b,d\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a und kommt in einen Zustand s_1, ein b und kommt in einen Zustand s_2 oder ein c und kommt in einen Zustand s_3. Sowohl im Zustand s_1 als auch im Zustand s_2 oder s_3 wird entweder ein a, b oder ein d akzeptiert. Es ergeben sich also 9 weitere Zustände, die auch Endzustände sind. Zu diesem Automaten gibt es auch wie daneben gezeigt eine vereinfachende Darstellung.



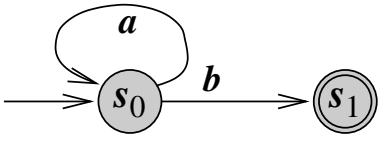
□

Wie wir im vorherigen Abschnitt schon gesehen haben, erzeugen Ausdrücke mit Iteration eine unendliche Sprachmenge. Bei einer unendliche Sprachmenge wird auch die Menge der Zustandsübergänge unendlich sein. Dennoch muss aber die Zustandsmenge nicht unendlich sein, denn die Transitionen können jeweils in die bereits existierenden Zustände zurückführen.

Beispiel 4.30 *Regulärer Ausdruck und zugehöriger Automat*
Gegeben seien die folgenden Ausdrücke RA aus Beispiel 4.25 über dem Alphabet $E = \{a,b,c,u,v,w\}$, die Produkt, Vereinigung und Iteration enthalten.
Gesucht sind endliche Automaten, welche die zu den folgenden regulären Ausdrücken gehörende reguläre Sprache $L(RA)$ besitzen.

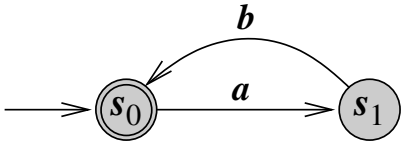
1) $A_1 = ((\boldsymbol{a})^*)\boldsymbol{b}$
2) $A_2 = (\boldsymbol{ab})^*$
3) $A_3 = ((\boldsymbol{a})^*)+\boldsymbol{b}$
4) $A_4 = (\boldsymbol{a}+\boldsymbol{b})^*$
5) $A_5 = (\boldsymbol{abc})^*$
6) $A_6 = \boldsymbol{a}^*+\boldsymbol{b}^*+\boldsymbol{c}^*$
7) $A_7 = (\boldsymbol{a}+\boldsymbol{b})^*+(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})^*$

1) Ausdruck $A_1 = ((\boldsymbol{a})^*)\boldsymbol{b}$, $L[A_1] = \{b,ab,aab,aaab,aaaab,...\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a oder ein b. Nach einem a kann er darauffolgendend immer wieder ein a oder ein b verarbeiten. Der Automat verarbeitet also ein a und kehrt damit wieder in den Anfangszustand s_0 zurück wo er wiederholt gleich immer wieder das nächste a verarbeiten kann. Der Automat weist eine Schleife auf, die nur den Zustand s_0 umfasst. Ein Zyklus, der nur den Zustand selbst umfasst, bezeichnet man auch Schlinge. Wird im Ausgangszustand s_0 jedoch ein b verarbeitet, dann kommt der Automat in einen weiteren Zustand s_1, der ein Endzustand ist. Es ergibt sich ein Automat mit der Zustandsmenge $S = \{s_0, s_1\}$, dem Endzustand $F = \{s_1\}$ und der Übergangsfunktion δ:



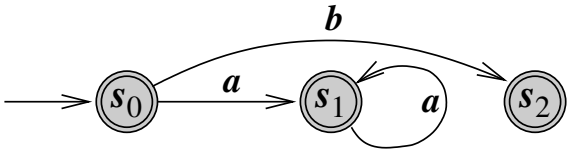
δ	a	b
s_0	s_0	s_1
s_1	–	–

2) Ausdruck $A_2 = (\boldsymbol{ab})^*$, $L[A_2] = \{\varepsilon, ab, abab, ababab, abababab, ...\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 ausschließlich ein a. Nach einem a kann er ausschließlich nur ein b verarbeiten. Danach wieder nur ein a und wieder ein b. Mit einem a kommt der Automat von Ausgangszustand s_0 in einen Zustand s_1. Von s_1 aus kommt der Automat mit einem b in den Ausgangszustand s_0 zurück. Der Automat besitzt eine Schleife, die aus den beiden Zuständen s_0 und s_1 besteht. Dabei stellt der Zustand s_0 einen Endzustand dar, denn immer wenn ein b verarbeitet wurde, dann wurde ein Wort akzeptiert. Es ergibt sich ein Automat mit der Zustandsmenge $S = \{s_0, s_1\}$, dem Endzustand $F = \{s_0\}$ und der Übergangsfunktion δ:



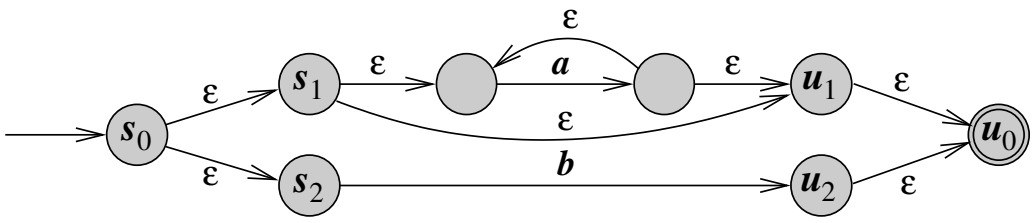
δ	a	b
s_0	s_1	–
s_1	–	s_0

3) Ausdruck $A_3 = ((\boldsymbol{a})^*) + \boldsymbol{b}$, $L[A_3] = \{\varepsilon, a, aa, aaa, aaaa, ..., b\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a oder ein b oder wegen $(a)^0$ die leere Transition ε. Nach einem a kann er darauffolgendend immer wieder ein a verarbeiten. Nach einem b kann er keine weitere Transition mehr ausführen. Der Automat verarbeitet also ein a und kommt damit in einen weiteren Zustand s_1, wo er wiederholt gleich immer wieder das nächste a verarbeiten kann. Er kann im Zustand s_1 beliebig viele a verarbeiten, aber er kann, nachdem er das erste a verarbeitet hat, kein b mehr verarbeiten. Er kann also den Zustand s_1 auch nicht mehr verlassen, der Zustand s_1 ist ein Endzustand. Der Automat weist somit eine Schleife auf, die nur den Endzustand s_1 umfasst. Wird im Ausgangszustand s_0 ein b verarbeitet, dann kommt der Automat in einen weiteren Zustand s_2, der ebenfalls ein Endzustand ist. Der Automat besitzt die Zustandsmenge $S = \{s_0, s_1, s_2\}$, die Endzustände $F = \{s_0, s_1, s_2\}$ und die Übergangsfunktion δ:



δ	a	b
s_0	s_0	s_2
s_1	s_1	–
s_2	–	–

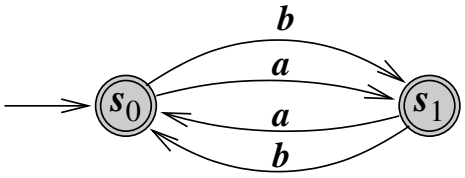
Wir wollen den EA zu diesem Ausdruck $A_3 = ((\boldsymbol{a})^*) + \boldsymbol{b}$ noch einmal formal mit Hilfe der Konstruktionsregeln herleiten um zu zeigen, dass unsere Überlegungen richtig sind. Dazu müssen wir den Ausdruck A_3 in zwei vereinigte Teilausdrücke aufspalten: $A_3 = R + S$, wobei $R = \boldsymbol{a}^*$ und $S = \boldsymbol{bc}$. Mit den Konstrukten für die Vereinigung und die Iteration folgt dann:

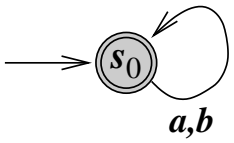


Wir können zeigen, dass dieser EA die Sprachmenge $L[A_3] = \{\varepsilon, a, aa, aaa, aaaa, ..., b\}$ akzeptiert. Wir sind aber noch nicht in der Lage, diesen EA auf die andere einfachere Form zu bringen. Insbesondere wird aber deutlich warum wir bei den vorher hergeleiteten EAs die Konstruktionsregeln nicht angewendet, sondern diese intuitiv hergeleitet haben. Die Konstruktionsregeln liefern uns sicher einen korrekten EA zum gegebenen regulären Ausdruck, dieser ist jedoch zumeist sehr viel komplexer als wir uns dies wünschen würden. Der formal hergeleitete EA muss noch erheblich reduziert werden.

4) Ausdruck $A_4 = (\boldsymbol{a} + \boldsymbol{b})^*$,
$L[A_4] = \{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, aaaa, ...\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a oder ein b. Nach einem a oder einem b kann er darauffolgendend immer wieder ein a oder ein b verarbeiten. Der Automat verarbeitet also ein a oder ein b und kommt in einen Zustand s_1. In diesem Zustand s_1 kann er

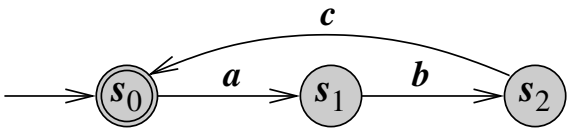
ebenso ein a oder ein b verarbeiten und kehrt damit wieder in den Anfangszustand s_0 zurück. Wichtig ist hier die Tatsache, dass dieser EA mit jeder weiteren Ausführung des Iterationsausdrucks sowohl in den Zweig mit a als auch mit b kommen kann. Der Automat weist zwei Zustände s_0 und s_1 auf, von Zustand s_0 gehen zwei Transitionen mit a und b nach s_1 und von s_1 gehen zwei Transitionen mit a und b nach s_0. Dabei sind beide Zustände Endzustände. Der Automat besitzt also die Zustandsmenge $S = \{s_0, s_1\}$, die Endzustände $F = \{s_0, s_1\}$ und die Übergangsfunktion δ. Alternativ kann dieser Automat auch durch einen einzigen Zustand beschrieben werden.





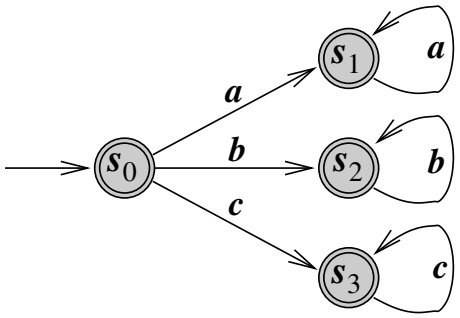
δ	a	b
s_0	s_1	s_1
s_1	s_0	s_0

5) Ausdruck $A_5 = (\boldsymbol{abc})^*$, $L[A_5] = \{\varepsilon, abc, abcabc, abcabcabc, abcabcabcabc, ...\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 ausschließlich ein a. Nach einem a kann er ausschließlich nur ein b verarbeiten. Nach dem b kann er nur ein c verarbeiten. Anschließend beginnt der Zyklus wieder von vorne, nach dem c akzeptiert er nur ein a, dann ein b und wieder ein c. Mit einem a kommt der Automat von Ausgangszustand s_0 in einen Zustand s_1. Von s_1 aus kommt der Automat mit einem b in einen Zustand s_2 und von s_2 mit einem c in den Ausgangszustand s_0 zurück. Der Automat besitzt eine Schleife, die aus den drei Zuständen s_0, s_1 und s_2 besteht. Dabei stellt der Ausgangszustand s_0 auch einen Endzustand dar, denn immer wenn ein c verarbeitet wurde, dann wurde ein Wort akzeptiert. Es ergibt sich ein Automat mit der Zustandsmenge $S = \{s_0, s_1, s_2\}$, dem Endzustand $F = \{s_0\}$ und der Übergangsfunktion δ:



δ	a	b	c
s_0	s_1	–	–
s_1	–	s_2	–
s_2	–	–	s_0

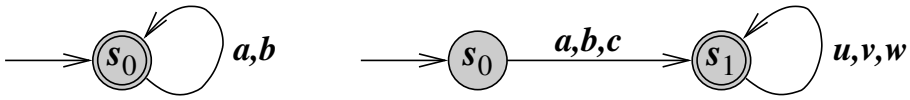
6) Ausdruck $A_6 = \boldsymbol{a}^* + \boldsymbol{b}^* + \boldsymbol{c}^*$, $L[A_6] = \{a\}^* \cup \{b\}^* \cup \{c\}^*$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 entweder ein a, ein b oder ein c oder wegen $(a)^0$ die leere Transition ε. Nach einem a kann er darauffolgendend immer wieder nur ein weiteres a verarbeiten. Nach einem b kann er wiederholt nur ein b verarbeiten und nach einem c kann er wiederholt nur ein c verarbeiten. Der Automat weist von Ausgangszustand s_0 ausgehend drei Folgezustände s_1, s_2 und s_3 auf, in die er jeweils mit a, b oder c gelangt. In jedem dieser drei Zustände weist der Automat eine Schlinge mit der entsprechenden Eingabe a, b oder c auf. Von diesen drei Zuständen aus kommt der Automat in keinen anderen Zustand mehr, auch nicht in den Ausgangszustand zurück. Alle drei Zustände s_1, s_2 und s_3 sind Endzustände. Der Automat besitzt die Zustandsmenge $S = \{s_0, s_1, s_2, s_3\}$, die Endzustände $F = \{s_0, s_1, s_2, s_3\}$ unddie Übergangsfunktion δ:



δ	a	b	c
s_0	s_1	s_2	s_3
s_1	s_1	–	–
s_2	–	s_2	–
s_3	–	–	s_3

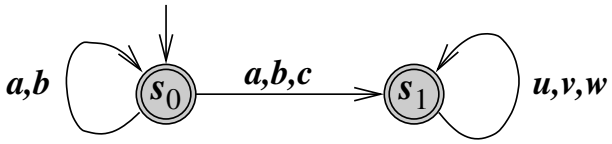
7) Ausdruck $A_7 = (\boldsymbol{a}+\boldsymbol{b})^* + (\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w})^*$, $L[A_7] = \{a,b\}^* \cup \{a,b,c\}\{u,v,w\}^*$
In diesem etwas komplexeren Ausdruck ist besonders auf die Klammerung sowie die Bindung der Operatoren zu achten. Wir betrachten zunächst die beiden Ausdrücke vor und nach dem ODER (+).

Der Ausdruck $R = (a+b)^*$ vor dem ODER erzeugt einen einzelnen Zustand (Anfangs- und Endzustand s_0) mit zwei Schleifen, eine mit der Transition a und eine mit der Transition b.



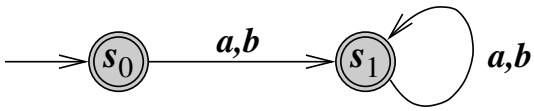
Der Ausdruck $S = (a+b+c)(u+v+w)^*$ nach dem ODER erzeugt zunächst neben dem Startzustand s_0 einen weiteren Zustand (Endzustand) s_1, der mit a, b oder c erreicht werden kann. In diesem Endzustand s_1 wiederum gibt es drei Schleifen, eine mit der Transition u, eine mit v und eine mit der Transition w.

Wenn wir nun die beiden Teilautomaten ohne Beachtung der Vereinigungsregel aus Abb. 4.7 vereinigen, dann ensteht der folgende EA:

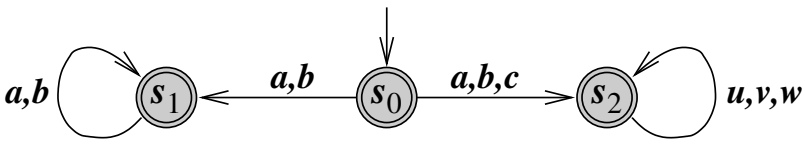


Betrachtet man das Verhalten dieses EA jedoch etwas genauer, dann fällt Folgendes auf: Der EA kann nach Ausführung eines a oder b aus dem Ausdruck $(a+b)^*$ immer noch ein a, b oder c aus dem Ausdruck $(a+b+c)$ und damit auch anschließend $(u+v+w)^*$ ausführen. Der reguläre Ausdruck A_7 hingegen sieht dies nicht vor, da es sich bei dem $+$ zwischen R und S um einen wechselseitigen Ausschluss handelt. Nach dem Beginn der Ausführung von $(a+b)^*$ darf der EA nicht mehr zurückkommen in den anderen Zweig, indem er auf a oder b noch $(u+v+w)^*$ ausführen kann. Das Problem ist, dass wir hier wie auch schon in Ausdruck $A_3 = ((a)^*)+b$ oder $A_6 = a^*+b^*+c^*$ dieser Aufgabe keine Zustände vereinigen dürfen, von denen der eine Schlinge aufweist, (s_0 von Automat R) die der andere nicht besitzt.

Um dieses Problem zu lösen und den Automaten korrekt zu spezifizieren, muss der Automat R für den $(a+b)^*$-Zweig verhaltensäquivalent, aber syntaktisch anders beschrieben werden, so dass sich die Schlinge nicht mehr im Startzustand befindet. Sobald der Automat R ein a oder ein b aus dem Ausdruck $(a+b)^*$ verarbeitet hat, muss er den Startzustand verlassen und im Folgezustand das gleiche Verhalten wie im Startzustand zeigen. Eine andere verhaltensäquivalente Darstellung sieht also folgendermaßen aus:



Vereinigt man nun diesen Automaten für R mit dem Automaten für den anderen Zweig S, dann ergibt sich exakt das gewünschte Verhalten. Der Automat besitzt die Zustandsmenge $S = \{s_0, s_1, s_2\}$, die Endzustände $F = \{s_1, s_2\}$ und die Übergangsfunktion δ:



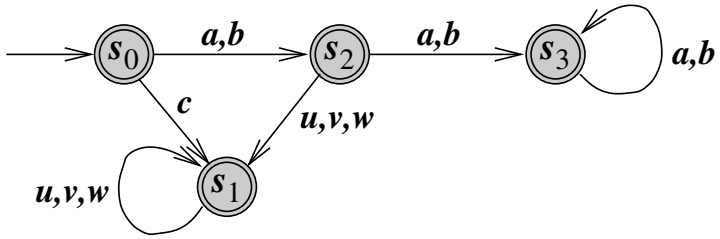
Man erkennt auch, dass wir durch den Ausdruck A_7 einen nichtdeterministischen Automaten beschrieben haben, d.h. vom Zustand s_0 aus kann der Automat mit einem a oder einem b sowohl nach Zustand s_1 oder nach Zustand s_2 übergehen. Die tatsächlich ausgeführte Transition ist rein zufällig. Nichtdeterministische Automaten sollen aber nicht mehr Gegenstand dieses elementaren Kurses sein, der Ausdruck A_7 soll lediglich einen Hinweis darauf geben, dass solche Eigenschaften existieren.

Man kann den Ausdruck A_7 auch in eine andere, äquivalente Form bringen:

$$\begin{aligned} A_7 &= (a+b)^* + (a+b+c)(u+v+w)^* \\ &= (a+b)^* + a.(u+v+w)^* + b.(u+v+w)^* + c.(u+v+w)^* \end{aligned}$$

Dieser Ausdruck wird durch einen Automaten mit fünf Zuständen realisiert, wobei die zwei zusätzlich entstandenen Zustände zu einem anderen Zustand äquivalent sind.

Ferner kann dieser Nichtdeterminismus durch eine verhaltensäquivalente Umformung wie in der folgenden Abbildung auch beseitigt werden.

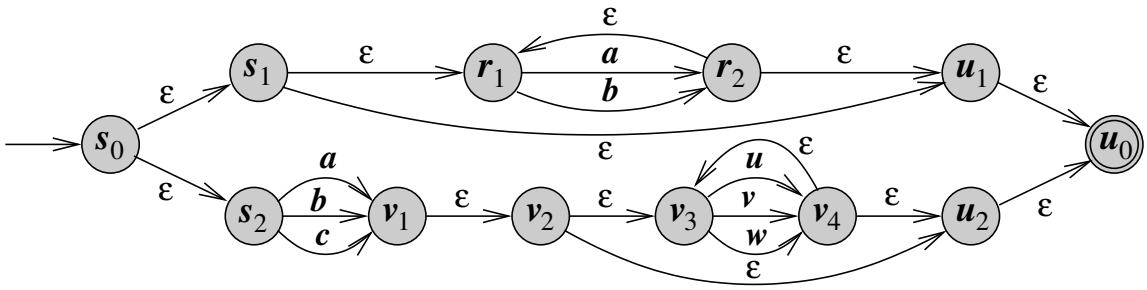


Der Ausdruck

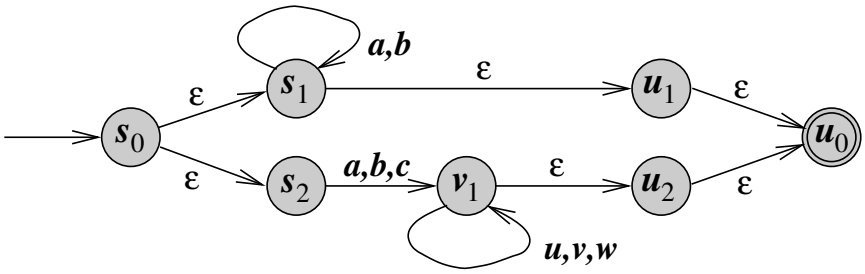
$$\begin{aligned} A_7 &= (a+b+\varepsilon).[(a+b+\varepsilon).(a+b)^* + (u+v+w+\varepsilon).(u+v+w)^*] + c.[(u+v+w)^*] \\ &= (a+b+\varepsilon).[(a+b)^* + (u+v+w)^*] + c.(u+v+w)^* \end{aligned}$$

beseitigt also den Nichtdeterminismus, der Automat weist zwar andere Zustände und einen anderen regulären Ausdruck, aber ein äquivalentes Verhalten auf. Man sagt auch die Automaten sind semantisch äquivalent und syntaktisch nicht.

Würden wir den formalen Konstruktionsweg gehen und auf den Ausdruck $A_7 = (a+b)^* + (a+b+c)(u+v+w)^*$ die Kombination aus den entsprechenden Operatoren Vereinigung, Verkettung und Iteration anwenden, dann würde dabei der folgende EA entstehen:



Würden wir hingegen auf die beiden obigen Teilautomaten für $R = (a+b)^*$ und $S = (a+b+c)(u+v+w)^*$ zur Vereinigung formal die Vereinigungsregel anwenden, dann würde dabei der folgende EA entstehen:



Der Grund für das Einfügen der ε-Transition im oberen Zweig ist, dass es keine Transitionen aus den beiden Endzuständen u_1 und u_2 heraus geben darf. □

Aufgabe 4.9 *Regulärer Ausdruck und zugehöriger EA*
Gegeben seien die folgenden regulären Ausdrücke RA über dem Alphabet $E = \{a,b\}$, die Produkt, Vereinigung und Iteration enthalten.
Gesucht sind endliche Automaten, welche die zu den folgenden regulären Ausdrücken gehörende reguläre Sprache $L(RA)$ besitzen.

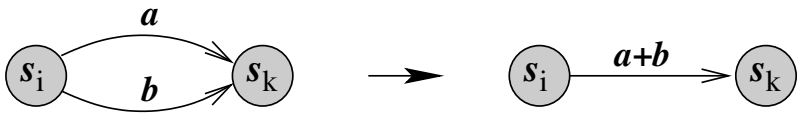
1) $A_1 = (ab)^* + (ab)^*$ 2) $A_2 = (a+\varepsilon)(ab)^*$

◇

4.4.4 Bestimmung eines regulären Ausdrucks zu einem EA

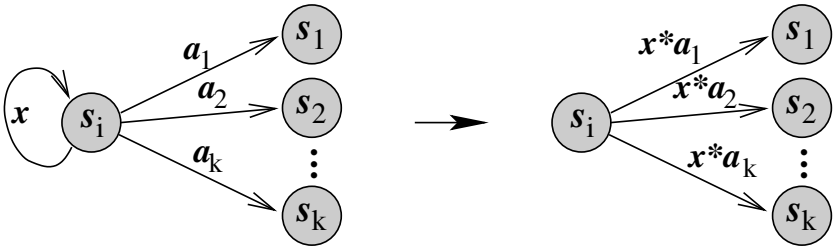
Das umgekehrte Vorgehen, die Konstruktion eines regulären Ausdrucks zu einem gegebenen EA, ist etwas schwieriger. Wir wollen dies aber auch behandeln, da der Entwurf von EA häufig anhand einer Verhaltensbeschreibung durch das Zustandsdiagramm erfolgt, wie wir dies am Beispiel des Getränkeautomaten mehrmals praktiziert haben. Diese Darstellung ist zwar anschaulich, intuitiv und leicht realisierbar, die vom EA akzeptierte Sprache aber nicht so leicht zu erkennen. Das folgende schematisierte Verfahren führt schrittweise von einem Zustandsdiagramm zum entsprechenden regulären Ausdruck. Dazu gibt es drei sog. Elimininationsregeln:

1. *Kantenelimination*:
Führen zwei Transitionen mit den Eingaben a und b von einem Zustand s_i in den gleichen Folgezustand s_k, dann können diese mit der Bezeichnung $a+b$ zusammengelegt werden.



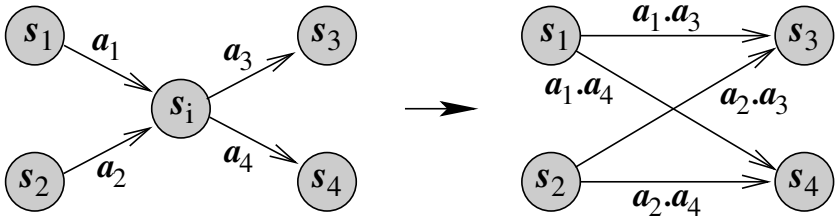
Der Automat kann diese Transition mit der Eingabe a oder mit der Eingabe b ausführen, dies ist semantisch äquivalent zu den beiden Transitionen mit a und b.

2. *Schleifenelimination*:
Weist ein Zustand s_i eine Schlinge auf, d.h. eine Transition, die in den Zustand zurückführt, dann kann diese beliebig oft ausführbare Schlinge mit den aus diesem Zustand herausführenden Transitionen konkateniert werden.



x^* steht für eine beliebige Anzahl an Transitionen mit der Eingabe x, die von den herausführenden Transitionen $(x^*.a_1, x^*.a_2, ..., x^*.a_k)$ gefolgt werden. Die rechts dargestellte Umformung ist also semantisch äquivalent zur Schlinge am Zustand s_i.

3. *Knotenelimination*:
Aufeinanderfolgende, durch einen Zustand getrennte Transitionen können konkateniert werden, um den Zwischenzustand zu eliminieren.

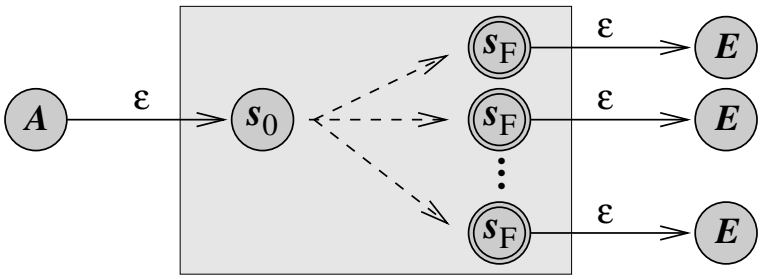


Dabei haben wir natürlich alle möglichen Kombinationen von Konkatenationen zu berücksichtigen, es muss also jede in den Zustand s_i eingehende Transition mit jeder aus dem Zustand s_i herausgehenden Transition kombiniert werden.

Um mit Hilfe dieser drei Regeln von einem endlichen Automaten $\text{EA} = (S, E, \delta, s_0, F)$ zum entsprechenden regulären Ausdruck RA zu kommen, gehen wir in folgenden Schritten vor:

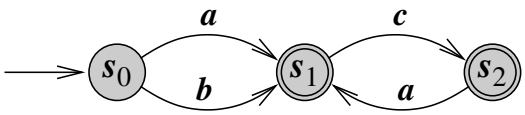
1. *Initialisierung:* Dem Zustandsraum des EA werden zwei Hilfszustände A und E hinzugefügt, der Anfangshilfszustand A besitzt eine leere ε-Transition auf den Startzustand s_0 und von allen

Endzuständen $s \in F$ geht eine leere ε-Transition auf den Endhilfszustand E. Durch die beiden Zustände A und E sowie die leeren ε-Transitionen wird die Semantik des gegebenen Automatendiagramms nicht verändert.

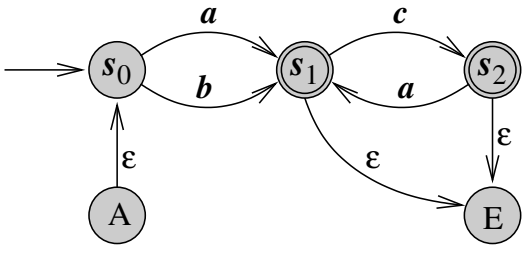


2. Mit der *Kantenelimination* werden soweit wie möglich alle Kanten zusammengelegt.
3. Mit der *Schleifenelimination* werden soweit wie möglich alle Schleifen eliminiert.
4. Mit der *Knotenelimination* werden soweit wie möglich alle Knoten eliminiert.
5. Die Schritte 2 bis 4 werden so lange wiederholt, bis keine Elimination mehr möglich ist. Es verbleiben also nur die Zustände A und E.
6. Formulierung des *regulären Ausdrucks:* Existiert zum Schluss keine Kante mehr zwischen A und E, dann ist der gesuchte reguläre Ausdruck leer. Ansonsten ist der reguläre Ausdruck RA durch die Kantenbeschriftung zwischen A und E gegeben.

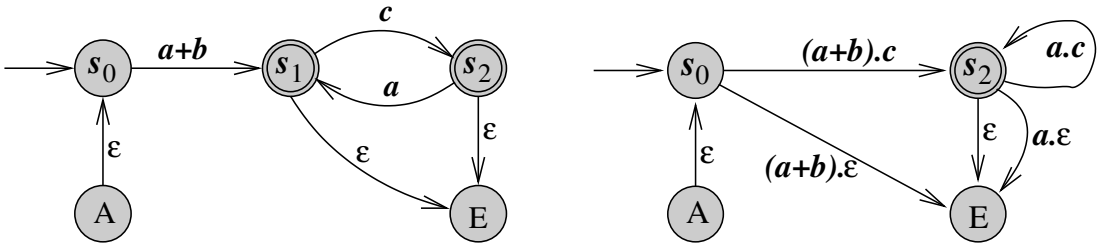
Beispiel 4.31 *Bestimmung des regulären Ausdrucks*
Gegeben sei der endliche Automaten EA$= (S,E,\delta,s_0,F)$ mit der Zustandsmenge $S = \{s_0,s_1,s_2\}$, dem Eingabealphabet $E = \{a,b,c\}$, den Endzuständen $F = \{s_1,s_2\}$ und der Übergangsfunktion δ. Gesucht ist der reguläre Ausdruck zu diesem Automaten EA.



Nach Schritt 1 führen wir die Initialisierung durch. Dazu werden die zwei Zustände A und E hinzugefügt. Eine leere Transition ε führt von Zustand A in den Anfangszustand s_0 und je eine leere Transition ε führt von den beiden Endzuständen s_1 und s_2 in den Zustand E.



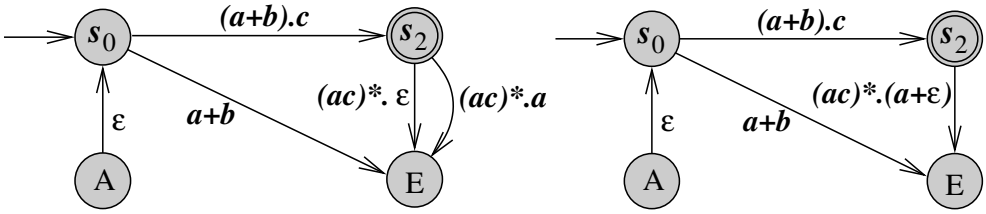
Nach Schritt 2 führen wir eine Kantenelimination durch. Dazu werden die beiden Transitionen mit a und mit b von Zustand s_0 nach Zustand s_1 zu einer Transition $a+b$ zusammengelegt.



Da keiner der Zustände eine Schlinge aufweist, eliminieren wir nun gemäß Schritt 4 den Zustand s_1. Dazu müssen die folgenden Modifikationen ausgeführt werden:

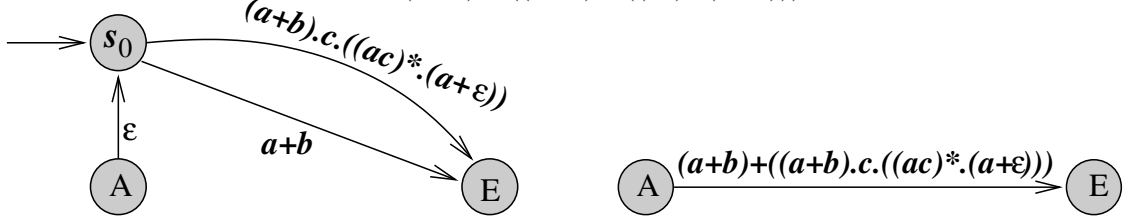
1. Die beiden Transitionen mit $a+b$ von s_0 nach s_1 und mit c von s_1 nach s_2 müssen konkateniert werden. Daraus ergibt sich eine Transition $(a+b).c$ von s_0 nach s_2.
2. Die beiden Transitionen mit $a+b$ von s_0 nach s_1 und mit ε von s_1 nach E müssen konkateniert werden. Daraus ergibt sich eine Transition $(a+b).\varepsilon$ von s_0 nach E.
3. Die beiden Transitionen mit a von s_2 nach s_1 und mit ε von s_1 nach E müssen konkateniert werden. Daraus ergibt sich eine Transition $a.\varepsilon$ von s_2 nach E.
4. Die beiden Transitionen mit a von s_2 nach s_1 und mit c von s_1 nach s_2 müssen konkateniert werden. Daraus ergibt sich eine Schlinge am Zustand s_2 mit $a.c$. Da ε die leere Transiton beschreibt, ergibt sich aus der Konkatenation einer beliebigen Transition w mit ε die Transition w, es gilt also $w\varepsilon = w$. Dementsprechend folgt vereinfachend $(a+b).\varepsilon = a+b$ und $a.\varepsilon = a$.

In dieser Darstellung weist der Zustand s_2 eine Schlinge auf, die wir nun entsprechend Regel 2 (Schritt 3) entfernen. Die Transitionen dieser beliebig oft ausführbaren Schlinge werden mit den aus diesem Zustand herausführenden Transitionen konkateniert. Da die Schlinge beliebig oft durchlaufen werden kann, ergibt sich aus ac zunächst $(ac)^*$. Der Zustand s_2 kann über die zwei Transitionen mit a und mit ε in den Zustand E verlassen werden. Nach der Elimination der Schlinge werden daraus die Transitionen $(ac)^*.a$ und $(ac)^*.\varepsilon$, wobei $(ac)^*.\varepsilon = (ac)^*$.



Im nächsten Schritt können nach der Regel der Kantenelimination die beiden Transitionen mit $(ac)^*.a$ und $(ac)^*$ von Zustand s_2 nach Zustand E zu einer Transition $(ac)^*.a + (ac)^*$ zusammengelegt werden. Dieser Ausdruck kann natürlich auch als $(\boldsymbol{ac})^*.\boldsymbol{a} + (\boldsymbol{ac})^* = (\boldsymbol{ac})^*.\boldsymbol{a} + (\boldsymbol{ac})^*.\varepsilon = (\boldsymbol{ac})^*.(\boldsymbol{a}+\varepsilon)$ geschrieben werden.
Mit einer weiteren Knotenelimination kann der Zustand s_2 entfernt werden. Und durch je eine weitere Kanten- und Knotenelimination folgt schließlich der reguläre Ausdruck, der aus der Kantenmarkierung zwischen den Zuständen A und E besteht:

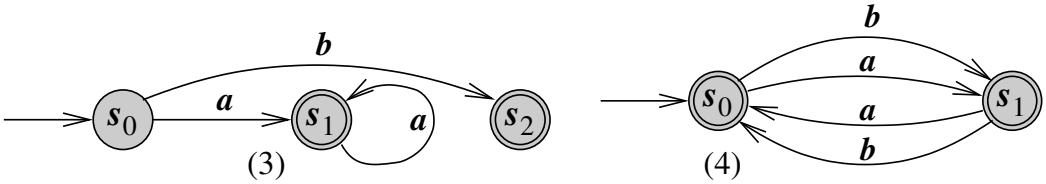
$$\mathrm{RA} = (\boldsymbol{a}+\boldsymbol{b}) + ((\boldsymbol{a}+\boldsymbol{b}).\boldsymbol{c}.((\boldsymbol{ac})^*.(\boldsymbol{a}+\varepsilon)))$$



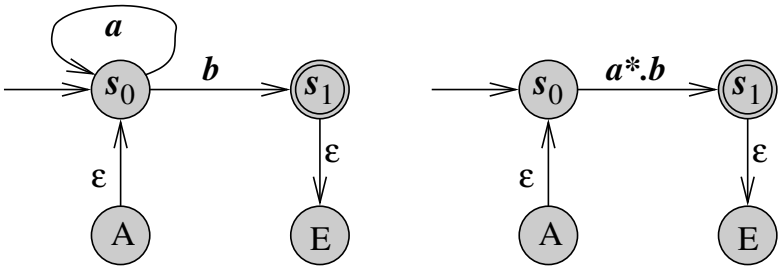
Dieser Ausdruck kann alternativ auch in eine andere Darstellung gebracht werden, wobei auch noch aufgrund der Prioritäten einige Klammern weggelassen werden können:

$$\mathrm{RA} = (\boldsymbol{a}+\boldsymbol{b}) + ((\boldsymbol{a}+\boldsymbol{b}).\boldsymbol{c}.((\boldsymbol{ac})^*.\boldsymbol{a}+(\boldsymbol{ac})^*)) = \boldsymbol{a}+\boldsymbol{b}+(\boldsymbol{a}+\boldsymbol{b}).\boldsymbol{c}.((\boldsymbol{ac})^*.\boldsymbol{a}+(\boldsymbol{ac})^*) \qquad \square$$

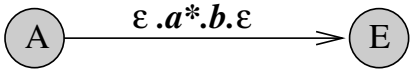
Beispiel 4.32 *Bestimmung des regulären Ausdrucks*
Man bestimme nach dem vorgegebenen Schema zu den folgenden vier EAs die regulären Ausdrücke



Automat (1): Wir führen zuerst die Initialisierung durch:

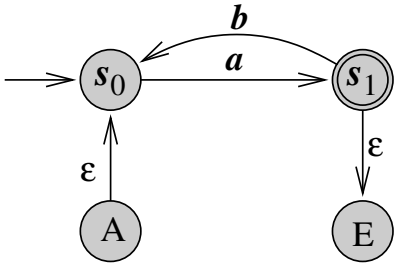


Eine Kanteneliminination ist nicht notwendig. Der Zustand s_0 weist eine Schlinge auf. Diese wird mit der Schleifenelimination entfernt, woraus eine Konkatenation von a^* an die Transition b von s_0 nach s_1 resultiert. Danach können mit einer Knotenelimination direkt die beiden Zustände s_0 und s_1 entfernt werden.

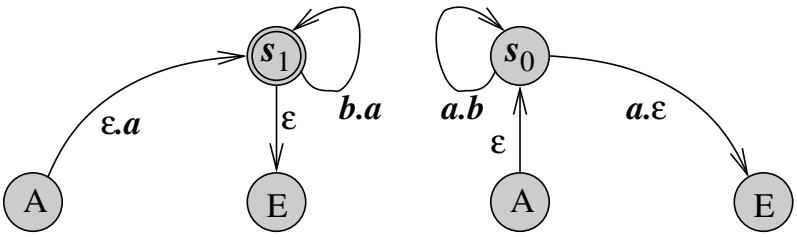


Daraus folgt: $\mathrm{RA}_1 = \varepsilon.a^*.b.\varepsilon = a^*.b = a^*b$

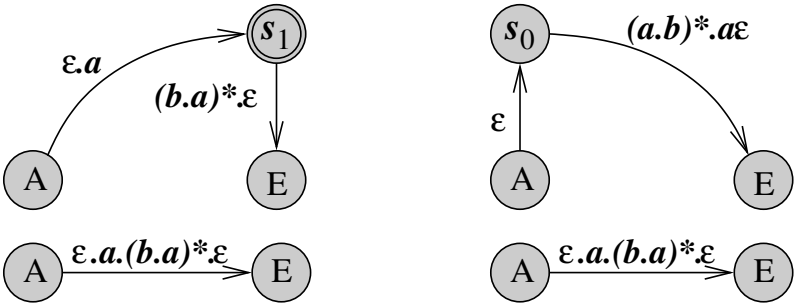
Automat (2): Zuerst Initialisierung



Mit einer Knotenelimination kann zuerst der Zustand s_0 entfernt werden. Die Elimination von s_0 hat eine Schlinge mit $b.a$ an s_1 und die Konkatenation von a an die Transition von A nach s_0 zur Folge.
Alternativ kann zuerst der Zustand s_1 entfernt werden, was eine Schlinge mit $a.b$ an s_0 und die Konkatenation von a an die Transition von s_0 nach E zur Folge hat.



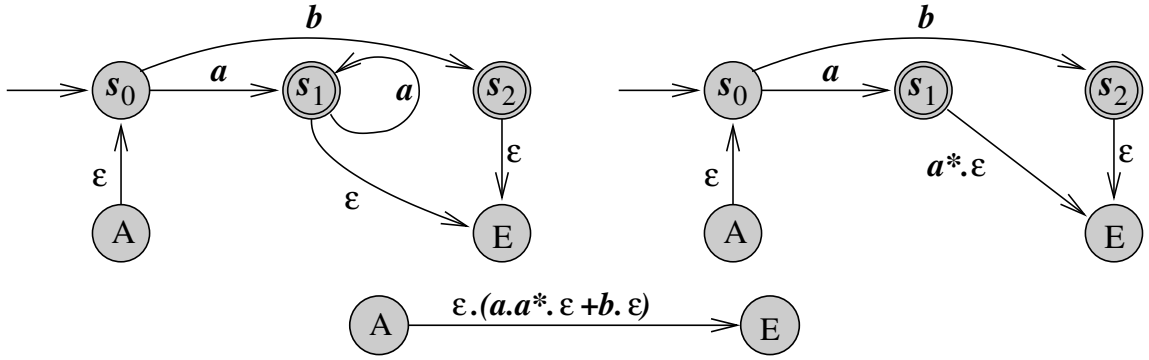
Wir verfolgen an diesem Beispiel beide Alternativen (links zuerst s_0, rechts zuerst s_1). Die Schlinge am Zustand s_1 bzw. s_0 wird mit der Schleifenelimination entfernt.



Mit einer Knotenelimination wird nun noch der Zustand s_1 bzw. s_0 entfernt. Es folgt somit:

$$\mathrm{RA}_2 = \varepsilon.\boldsymbol{a}.(\boldsymbol{ba})^*.\varepsilon = \varepsilon.(\boldsymbol{ab})^*.\boldsymbol{a}.\varepsilon = \boldsymbol{a}.(\boldsymbol{ba})^* = (\boldsymbol{ab})^*.\boldsymbol{a}$$

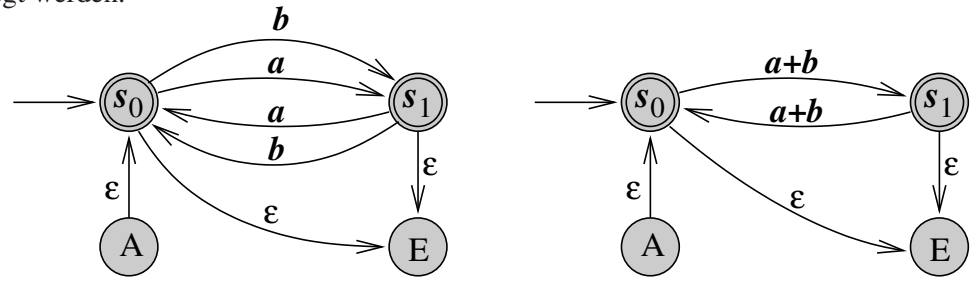
Automat (3): Zuerst Initialisierung
Der Zustand s_1 weist eine Schlinge auf. Diese wird mit der Schleifenelimination entfernt.



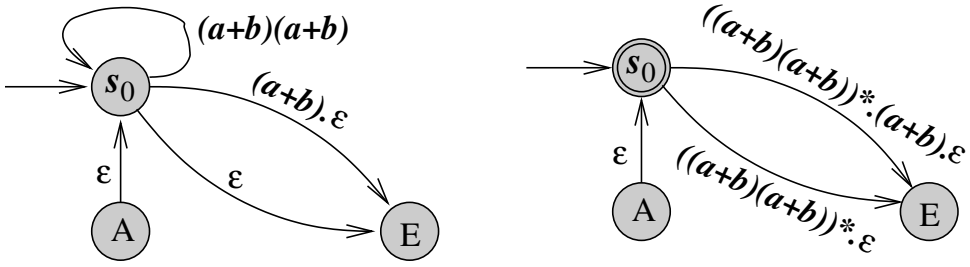
Danach können mit einer Knotenelimination die drei Zustände s_0 und s_1 und s_2 entfernt werden. Es folgt somit:

$$\mathrm{RA}_3 = \varepsilon.(\boldsymbol{a}.\boldsymbol{a}^*.\varepsilon + \boldsymbol{b}.\varepsilon) = \boldsymbol{a}.\boldsymbol{a}^*.\varepsilon + \boldsymbol{b}.\varepsilon = \boldsymbol{a}.\boldsymbol{a}^* + \boldsymbol{b} = \boldsymbol{a}^* + \boldsymbol{b}.$$

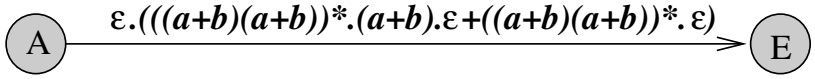
Automat (4): Zuerst Initialisierung
Mit der Kanteneliminination können je zwei Transitionen zwischen den Zuständen s_0 und s_1 zusammengelegt werden.



Danach kann mit einer Knotenelimination der Zustand s_0 oder s_1 entfernt werden. Dadurch weist der Zustand s_1 bzw. s_0 eine Schlinge auf, die mit der Schleifenelimination entfernt wird.



Zusammenlegen der Kanten von s_0 nach E ergibt: $((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^*.(\boldsymbol{a}+\boldsymbol{b}).\varepsilon + ((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^*.\varepsilon$
Schließlich kann mit der Knotenelimination der Zustand s_0 entfernt werden, so dass folgt:



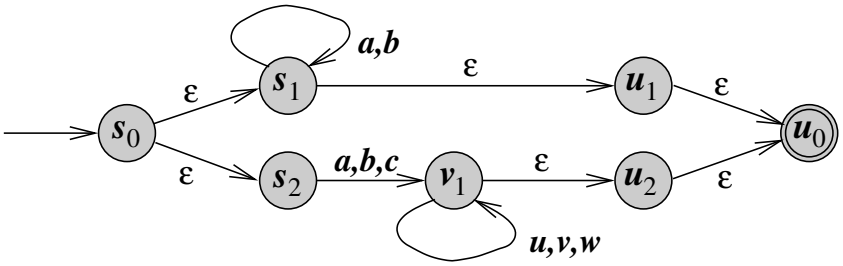
$$\begin{aligned}\mathrm{RA}_4 &= \varepsilon.(((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^*.(\boldsymbol{a}+\boldsymbol{b}).\varepsilon + ((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^*.\varepsilon) \\ &= ((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^*.(\boldsymbol{a}+\boldsymbol{b}) + ((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^*\end{aligned}$$

Nun ist $((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^*.(\boldsymbol{a}+\boldsymbol{b}) = (\boldsymbol{a}+\boldsymbol{b})^n$ mit $n = 1,3,5,7,...$
und $((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}))^* = (\boldsymbol{a}+\boldsymbol{b})^n$ mit $n = 0,2,4,6,...$
Zusammen haben wir also

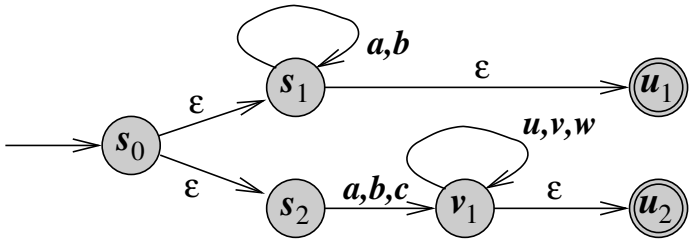
$$\mathrm{RA}_4 = (\boldsymbol{a}+\boldsymbol{b})^*$$

□

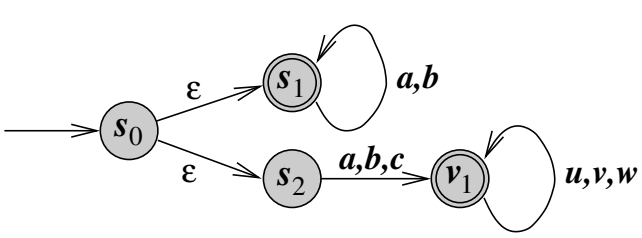
Beispiel 4.33 *Vereinfachen von EAs*
Wir wollen im Folgenden einige Regeln dazu nutzen, um die zum Ausdruck A_7 des Beispiels 4.30 formal hergeleiteten zwei EAs zu vereinfachen.

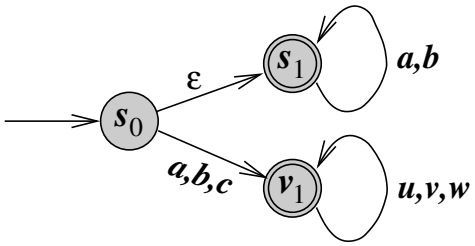


In diesem EA (wir nennen ihn EA_1) kann zuerst der Endzustand u_0 eliminiert werden. Die Elimination basiert auf der Regel $\varepsilon.\varepsilon = \varepsilon$, wodurch die beiden von u_1 und u_2 aus in den Endzustand u_0 führenden ε-Transitionen weggelassen werden können. Damit werden die Zustände u_1 und u_2 zu Endzuständen.

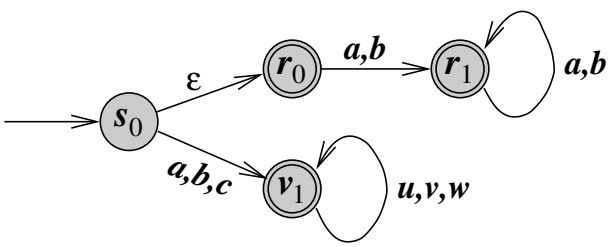


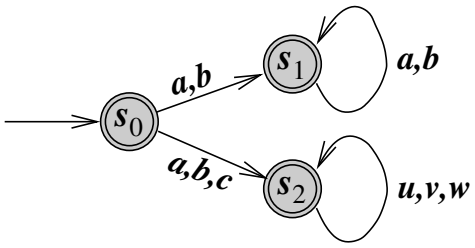
Ebenso können basierend auf der Regel $\varepsilon.w = w.\varepsilon = w$ die beiden von s_1 nach u_1 und von v_1 nach u_2 führenden ε-Transitionen weggelassen werden. Damit können die Zustände u_1 und u_2 eliminiert werden, die Zustände s_1 und v_1 werden zu Endzuständen. Anschließend können dann mit den Regeln $a.\varepsilon = a$, $b.\varepsilon = b$ und $c.\varepsilon = c$ die ε-Transitionen zwischen den Zuständen s_0 und s_2 weggelassen werden. Damit wird der Zustand s_2 eliminiert.





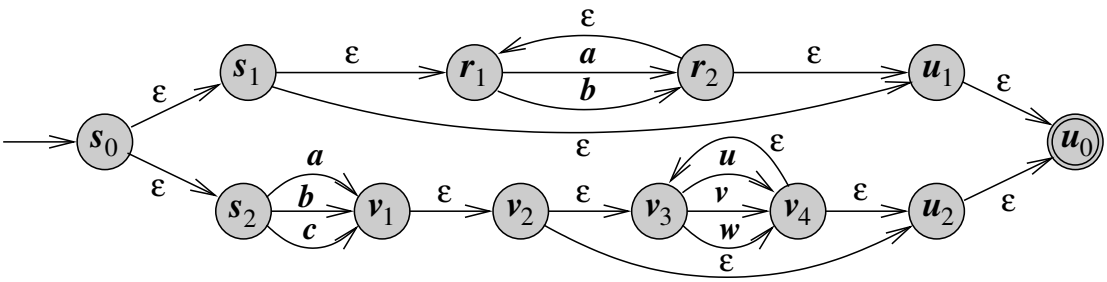
Die ε-Transition zwischen den Zuständen s_0 und s_1 kann aber so nicht direkt entfernt werden. Hier kommen wir aber mit einem kleinen Trick weiter. Durch die Äquivalenzen $a^* = \varepsilon + a.a^*$ bzw. $b^* = \varepsilon + .b.b^*$ kann der Zustand s_1 in zwei Zustände r_0 und r_1 aufgespalten werden, die durch die Transitionen a und b verbunden sind.





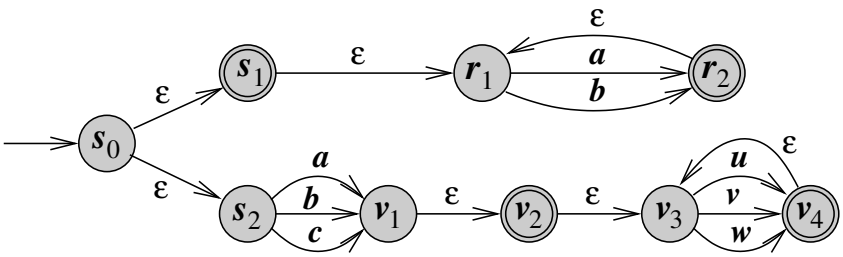
Anschließend kann die ε-Transition von Zusand s_0 nach Zustand r_0 aufgrund der Regeln $.\varepsilon.a = a$ und $\varepsilon.b = b$ weggelassen werden. Dadurch wird der Zustand r_0 eliminiert, der Zustand s_0 wird zum Endzustand. Nach Umbenennung der Zustände ergibt sich dann der rechts dargestellte EA.

Betrachten wir nun den zweiten etwas komplexeren Automaten EA_2.

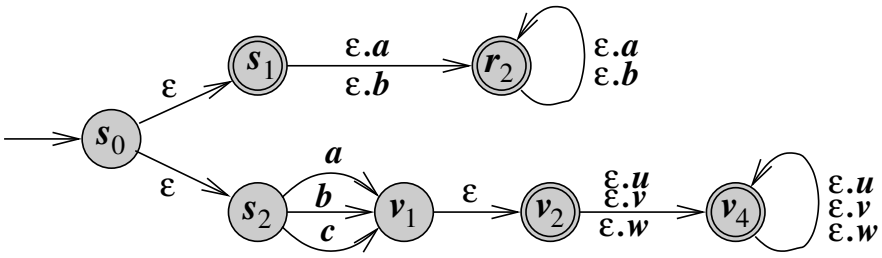


Auch hier können zuerst die beiden von u_1 und u_2 in den Endzustand u_0 führenden ε-Transitionen weggelassen werden, wodurch der Zustand u_0 eliminiert wird.

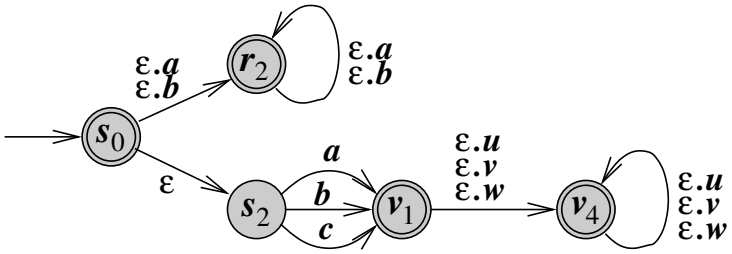
Dann können die beiden von s_1 und r_2 nach u_1 führenden ε-Transitionen aufgrund der Regeln $a.\varepsilon = a$ und $b.\varepsilon = b$ bzw. $\varepsilon.\varepsilon = \varepsilon$ weggelassen werden, wodurch der Zustand u_1 eliminiert wird. Ebenso können die beiden von v_2 und v_4 nach u_2 führenden ε-Transitionen aufgrund der Regeln $u.\varepsilon = u$, $v.\varepsilon = v$ und $w.\varepsilon = w$ bzw. $\varepsilon.\varepsilon = \varepsilon$ weggelassen werden, wodurch der Zustand u_2 eliminiert wird. Damit werden die Zustände r_2 und v_4 sowie s_1 und v_2 zu Endzuständen.



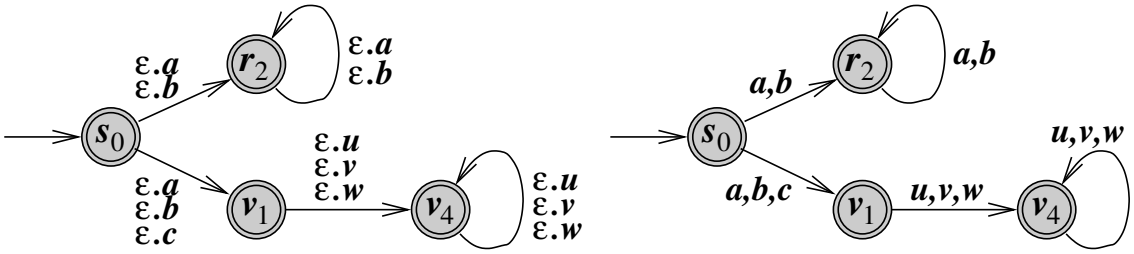
Als nächstes können durch eine Knotenelimination die Zustände r_1 und v_3 eliminiert werden. Die Elimination von Zustand r_1 führt zu zwei Schleifen mit $\varepsilon.a = a$ und mit $\varepsilon.b = b$ am Zustand r_2. Die Elimination von Zustand v_3 führt zu drei Schleifen mit $\varepsilon.u = u$, mit $\varepsilon.v = v$ und mit $\varepsilon.w = w$ am Zustand v_4.



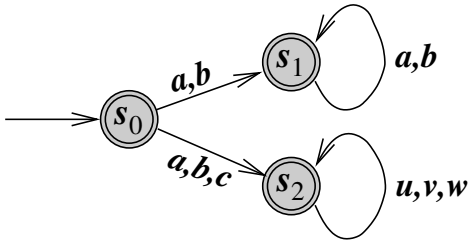
Wir erkennen sofort, dass wir nach der Regel $\varepsilon.\varepsilon = \varepsilon$ die zwei Zustände s_1 und v_2 elimnieren können. Damit werden die Zustände s_0 und v_1 zu Endzuständen.



Weiter kann mit einer Knotenelimination der Zustand s_2 und eliminiert werden. Entfernen der ε-Transitionen ergibt den EA rechts daneben.

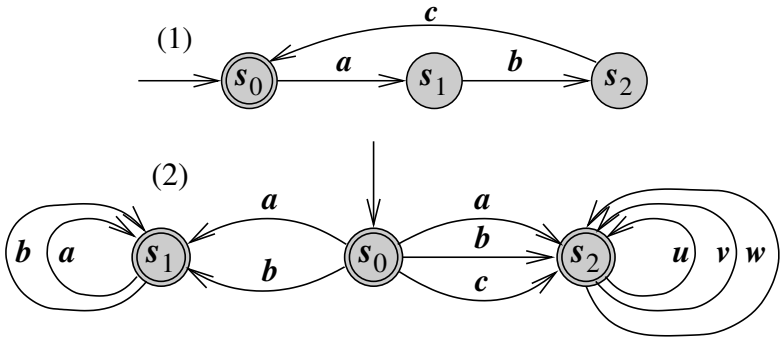


Die Elimination von Zustand v_1 ergibt dann bis auf Umbenennung der Zustände den EA, der bereits aus EA_1 erzeugt wurde.



□

Aufgabe 4.10 *Bestimmung des regulären Ausdrucks*
Man bestimme nach dem vorgegebenen Schema zu den folgenden zwei Automaten die regulären Ausdrücke.



◇

4.5 Weitere Aspekte und Varianten endlicher Automaten

4.5.1 Äquivalenz zweier endlicher Automaten

Das Verhalten oder die Funktion eines endlichen Automaten (EA) wird durch die Menge der Zustände und die Zustandsübergangsfunktion δ festgelegt. Die Funktion kann natürlich auch durch die von diesem EA akzeptierte Sprache L angegeben werden. Im vorherigen Abschnitt haben wir zu einer gegebenen Sprache L einen Automaten mit der Zustandsmange S und der Übergangsfunktion δ konstruiert, der diese Sprache L akzeptiert, also $L(EA) = L$. Dabei haben wir erkannt, dass es möglich ist, zu einer Sprache L verschiedene Automaten zu konstruieren. Tatsächlich kann es sogar unendliche viele verschiedene Automaten geben, für die $L(EA) = L$ gilt. Offensichtlich liegen die Unterschiede dieser von der Funktion her äquivalenten Automaten in der Zustandsmenge S und in der Übergangsfunktion δ. Im Eingabealphabet können sich diese Automaten nicht unterscheiden, denn unterschiedliche Eingabealphabete erzeugen unterschiedliche Sprachen. Diese Erkenntnis führt unmittelbar zu der Vermutung, dass es zu jedem endlichen Automaten EA einen äquivalenten *minimalen Automaten* EA_M gibt.

In der Praxis sind wir stets darin interessiert, zu einem gegebenen Problem einen vom Aufbau her möglichst einfachen Automaten zu entwerfen, der technisch und wirtschaftlich am einfachsten zu realisieren ist. Dabei ist es naheliegend, eine möglichst geringe Anzahl interner Zustände als Kriterium für einen einfachen Automaten anzusehen. Deswegen wollen wir in diesem Abschnitt den oben aufgeworfenen folgenden Fragen genauer nachgehen:

- Gibt es zu einem Automaten EA einen Automaten EA_M, der dasselbe Verhalten aufweist sowie die gleiche Sprache akzeptiert, aber mit weniger Zuständen auskommt?
- Gibt es zu jedem Automaten EA einen äquivalenten *minimalen Automaten* EA_M, der eine minimale Anzahl Zustände aufweist?
- Kann dieser Minimalautomat bis auf die Unterschiede in der Bezeichnung der Zustände eindeutig bestimmt werden?
- Wenn es den Minimalautomaten gibt, kann man einen effektiven endlichen Algorithmus angeben, um diesen Minimalautomaten zu bestimmen?

Wenn es zu jedem EA einen äquivalenten *minimalen* EA_M gibt, dann müssen wir zuerst formal festlegen, was wir genau unter der Äquivalenz zweier EAs verstehen:

Definition 4.9 *Äquivalenz zweier Automaten*
Zwei endliche Automaten $EA_1 = \{E, S_1, \delta_1, s_{01}, F_1\}$ *und* $EA_2 = \{E, S_2, \delta_2, s_{02}, F_2\}$ *mit demselben Eingabealphabet E heißen äquivalent, abgekürzt* $EA_1 \sim EA_2$*, wenn sie dieselbe Sprache akzeptieren, d.h. wenn* $L(EA_1) = L(EA_2)$ *gilt.*

Dabei stellt die Relation $\sim$ eine Äquivalenzrelation dar, d.h. die Relation $\sim$ ist reflexiv, symmetrisch und transitiv. Wir werden im Folgenden sehen, dass zwei Automaten nur dann äquivalent sein können, wenn deren Zustände äquivalent sind. Das heisst nicht, dass die beiden Automaten gleich viele Zustände enthalten müssen, sondern dass es zu jedem Zustand des einen Automaten einen entsprechenden äquivalenten Zustand des anderen Automaten geben muss. Dabei können n Zustände des einen Automaten auf einen Zustand des anderen Automaten abgebildet werden. Unter einem äquivalenten Zustand verstehen wir dabei einen Zustand mit äquivalentem Verhalten.

Um die Bestimmung der Äquivalenz zweier EAs möglichst einfach zu halten, entfernen wir die sog. *nicht erreichbaren* Zustände aus unserer Betrachtung. Damit sind diejenigen Zustände gemeint, die ein Automat niemals durchlaufen kann, da in diesen Zuständen keine Transition endet. Diese Zustände können, egal welches Eingabewort auch immer anliegt, niemals erreicht werden. Für diese Zustände s gilt: $\forall w \in E^* : \delta(s_0, w) \neq s$. Sie sind damit für das uns interessierende Verhalten des Automaten nicht von Bedeutung. Abgesehen davon sind solche Zustände in unseren bisherigen Betrachtungen auch noch nicht vorgekommen.

Da die Äquivalenz von EAs unmittelbar mit der Äquivalenz ihrer Zustände verknüpft ist, gehen wir für diese Betrachtung von der sehr viel leichter überschaubaren Zustandsäquivalenz aus, um von dort aus auf die Automatenäquivalenz zu folgern. Zu diesem Zweck werden wir zunächst versuchen, die Äquivalenz von Zuständen genauer zu fixieren. Zwei Zustände sind äquivalent, wenn sie das gleiche Verhalten aufweisen. Das Verhalten in einem Zustand wird bestimmt durch die Transitionsmöglichkeiten in diesem Zustand, d.h. durch die Transitionen, die aus diesem Zustand herausführen. Zwei äquivalente Zustände müssen also die gleichen Transitionen aus diesem Zustand heraus aufweisen.

Darauf stellt sich natürlich unmittelbar die Frage, was genau gleiche Transitionen bedeutet. Transitionen bzw. Zustandsübergänge wiederum sind bestimmt durch die Eingabe E und den Folgezustand s^{n+1}. Daraus folgt also weiter, dass zwei äquivalente Zustände Transitionen mit der gleichen Eingabe und dem gleichen Folgezustand aufweisen müssen. Die gleiche Eingabe ist offensichtlich leicht verifizierbar. Was aber bedeutet gleicher Folgezustand? Und da kommen wir wieder zum Ausgangspunkt zurück. Zwei Folgezustände sind äquivalent, wenn sie das gleiche Verhalten aufweisen. An dieser Stelle wird offensichtlich, dass der Äquivalenzbestimmung eine Rekursion zugrunde liegen muss. Diese rekursive Bestimmung der Äquivalenz muss natürlich in Zuständen ansetzen, dessen Verhalten wir unmittelbar festlegen können und von denen wir damit behaupten können, dass sie äquivalent sind. Solche Zustände sind die Endzustände in F. Wenn wir also zwei Automaten auf ihre Äquivalenz hin untersuchen wollen, müssen wir uns als erstes zwei zueinander äquivalente Endzustände suchen und dort unsere rekursive Äquivalenzanalyse beginnen.

Da die Endzustände Ausgangspunkt unserer Äquivalenzbestimmung sind, können wir als nächstes die Zustände miteinander vergleichen, die Transitionen in die Endzustände besitzen. Zwei Vorgängerzustände zu diesen Endzuständen sind äquivalent, wenn sie Transitionen mit den gleichen Eingaben und den gleichen Endzuständen besitzen. Wir bezeichnen diese Äquivalenz als 1-Äquivalenz, weil hier nur genau je eine Transition relevant ist. Weiter sind zwei Vorvorgängerzustände äquivalent, wenn sie Transitionen mit den gleichen Eingaben und den äquivalenten Folgezuständen besitzen, die wiederum Transitionen mit den gleichen Eingaben in die gleichen Endzustände aufweisen. Analog handelt es sich um eine 2-Äquivalenz, weil hier eine Sequenz aus genau zwei Transition relevant ist. Setzt man diesen Algorithmus fort, so kommt man allgemein zu einer k-Äquivalenz.

Zwei Zustände sind tatsächlich gleich oder äquivalent zueinander, wenn die betrachtete k-Sequenz für beliebige $k \in \mathbb{N}_0$ im äquivalenten Endzustand endet. Ist dies nur für bestimmte k der Fall, dann sind sie nur k-äquivalent. Wir wollen mit diesen Erklärungen eine geeignete Aussage über die Äquivalenz von Zuständen in einer Definition festhalten:

Definition 4.10 *k-Äquivalenz von Zuständen*
Es seien $EA_1 = \{E, S_1, \delta_1, s_{01}, F_1\}$ *und* $EA_2 = \{E, S_2, \delta_2, s_{02}, F_2\}$ *zwei endliche Automaten mit demselben Eingabealphabet* E.

Zwei Zustände $s_1 \in S_1$ *und* $s_2 \in S_2$ *heißen k-äquivalent* ($k \in \mathbb{N}_0$)*, abgekürzt durch* $s_1 \overset{k}{\sim} s_2$*, wenn für alle* $w \in E^*$ *mit* $|w| \leq k$ *gilt:* $s_1^k = \delta_1(s_1, w)$ *und* $s_2^k = \delta_2(s_2, w)$ *sind entweder beide Endzustände oder sie sind beide keine Endzustände.*

Mit Hilfe dieser Definition können wir nun auch die Äquivalenz zweier Zustände definieren:

Definition 4.11 *Äquivalenz von Zuständen*
Es seien $EA_1 = \{E, S_1, \delta_1, s_{01}, F_1\}$ *und* $EA_2 = \{E, S_2, \delta_2, s_{02}, F_2\}$ *zwei endliche Automaten mit demselben Eingabealphabet* E.

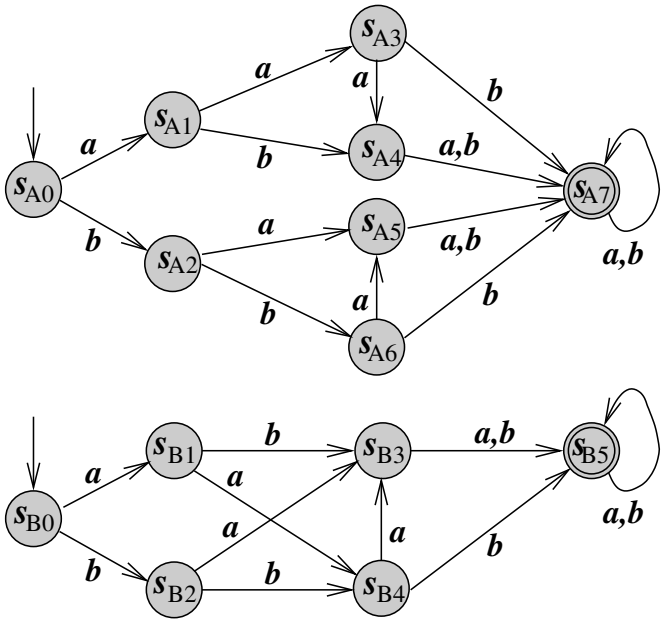
Zwei Zustände $s_1 \in S_1$ *und* $s_2 \in S_2$ *heißen äquivalent zueinander, abgekürzt durch* $s_1 \sim s_2$*, wenn für alle* $k \in \mathbb{N}_0$ *gilt:* $s_1 \overset{k}{\sim} s_2$

Diese Definition gilt insbesondere auch für zwei Zustände s_1 und s_2 ein- und desselben Automaten $\mathrm{EA}_1 = \mathrm{EA}_2 = \mathrm{EA}$. Anhand der Eigenschaften reflexiv, symmetrisch und transitiv ist leicht nachprüfbar, dass es sich bei den Relationen $\sim$ und $\overset{k}{\sim}$ auf der Zustandsmenge eines Automaten um Äquivalenzrelationen im mathematischen Sinne handelt.

Ferner gilt die Definition 4.10 nur für vollständig definierte Automaten, d.h. in jedem Zustand ist für jedes Element des Eingangsalphabets eine Transition festgelegt.

Das folgende Beispiel soll Ihnen zeigen, dass mit der vorliegenden Definition zwar eine formale Beschreibung der Äquivalenz vorliegt, diese Definition aber in dieser Form für die Bestimmung der Äquivalenz unbrauchbar ist. Wir müssen also einen geeigneten Algorithmus herleiten, der die Äquivalenz auf Basis dieser Definition berechnet.

Beispiel 4.34 *Äquivalente Automaten*
Gegeben seien die folgenden beiden endlichen Automaten EA_A und EA_B mit dem Eingabealphabet $E_A = E_B = \{a,b\}$, den Endzuständen $F_A = \{s_{A7}\}$, $F_B = \{s_{B5}\}$ und den folgenden Übergangsfunktionen:



a) Welche Zustände innerhalb des Automaten EA_A sind k-äquivalent und welche sind äquivalent?

b) Welche Zustände von EA_A sind k-äquivalent zu welchen Zuständen in EA_B?

c) Welche Zustände von EA_A sind äquivalent zu welchen Zuständen in EA_B?

a) k-äquivalente Zustände innerhalb EA_A
Wir wollen k-äquivalente Zustände innerhalb des Automaten EA_A entsprechend der Definition 4.10 finden. Wir beginnen mit der trivialen 0-Äquivalenz $\overset{0}{\sim}$, bei der nur die Zustände selbst gegeneinander bzgl. der Eigenschaft *Endzustand* $s \in F$ oder *kein Endzustand* $s \notin F$ verglichen werden. Da nur s_{A7} ein Endzustand ist, sind bis auf den Zustand s_{A7} alle anderen Zustände s_{A0}, s_{A1}, s_{A2}, s_{A3}, s_{A4}, s_{A5} und s_{A6} paarweise zueinander 0-äquivalent $\overset{0}{\sim}$. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A1}		$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A2}			$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A3}				$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A4}					$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A5}						$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A6}							$\overset{0}{\sim}$	
s_{A7}								$\overset{0}{\sim}$

Für die Bestimmung der 1-Äquivalenz $\overset{1}{\sim}$ müssen nun für alle möglichen Zustandspaare (s_i, s_j) deren Folgezustände $s_i^1 = \delta(s_i, w)$ und $s_j^1 = \delta(s_j, w)$ für alle $w \in E^*$ mit $|w| = 1$ untereinander bzgl. der

Eigenschaft *Endzustand* $s \in F$ oder *kein Endzustand* verglichen werden. Dazu stellen wir uns alle Folgezustände tabellarisch dar:

$\delta(s_{A0},a) = s_{A1}$	$\delta(s_{A0},b) = s_{A2}$
$\delta(s_{A1},a) = s_{A3}$	$\delta(s_{A1},b) = s_{A4}$
$\delta(s_{A2},a) = s_{A5}$	$\delta(s_{A2},b) = s_{A6}$
$\delta(s_{A3},a) = s_{A4}$	$\delta(s_{A3},b) = s_{A7} \in F$
$\delta(s_{A4},a) = s_{A7} \in F$	$\delta(s_{A4},b) = s_{A7} \in F$
$\delta(s_{A5},a) = s_{A7} \in F$	$\delta(s_{A5},b) = s_{A7} \in F$
$\delta(s_{A6},a) = s_{A5}$	$\delta(s_{A6},b) = s_{A7} \in F$
$\delta(s_{A7},a) = s_{A7} \in F$	$\delta(s_{A7},b) = s_{A7} \in F$

In jeder Zeile stehen jeweils die möglichen 1-Übergänge eines Zustandes. 1-äquivalent sind die Zustände, die bei beiden Transitionen $\delta(s_i,w)$ mit $w=a$ und $w=b$ das gleiche Folgezustandsmuster mit *Endzustand* $s \in F$ oder *kein Endzustand* aufweisen und 0-äquivalent sind. Dies ist bei den Zuständen s_{A0}, s_{A1} und s_{A2} der Fall, die Zustände s_{A0}, s_{A1} und s_{A2} sind 1-äquivalent. Ebenso sind die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} 1-äquivalent. Der Zustand s_{A7} kann zu keinem anderen Zustand 1-äquivalent sein, da dieser zu keinem anderen Zustand 0-äquivalent ist. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{A1}		$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{A2}			$\overset{1}{\sim}$					
s_{A3}				$\overset{1}{\sim}$			$\overset{1}{\sim}$	
s_{A4}					$\overset{1}{\sim}$	$\overset{1}{\sim}$		
s_{A5}						$\overset{1}{\sim}$		
s_{A6}							$\overset{1}{\sim}$	
s_{A7}								$\overset{1}{\sim}$

Bei der 2-Äquivalenz $\overset{2}{\sim}$ müssen alle möglichen Zustandspaare (s_i,s_j) hinsichtlich $s_i^2 = \delta(s_i,w)$ und $s_j^2 = \delta(s_j,w)$ für alle $w \in E^*$ mit $|w|=2$ untereinander bzgl. der Eigenschaft *Endzustand* oder *kein Endzustand* verglichen werden.

$\delta(s_{A0},aa) = s_{A3}$	$\delta(s_{A0},ab) = s_{A4}$	$\delta(s_{A0},ba) = s_{A5}$	$\delta(s_{A0},bb) = s_{A6}$
$\delta(s_{A1},aa) = s_{A4}$	$\delta(s_{A1},ab) = s_{A7} \in F$	$\delta(s_{A1},ba) = s_{A7} \in F$	$\delta(s_{A1},bb) = s_{A7} \in F$
$\delta(s_{A2},aa) = s_{A7} \in F$	$\delta(s_{A2},ab) = s_{A7} \in F$	$\delta(s_{A2},ba) = s_{A5}$	$\delta(s_{A2},bb) = s_{A7} \in F$
$\delta(s_{A3},aa) = s_{A7} \in F$	$\delta(s_{A3},ab) = s_{A7} \in F$	$\delta(s_{A3},ba) = s_{A7} \in F$	$\delta(s_{A3},bb) = s_{A7} \in F$
$\delta(s_{A6},aa) = s_{A7} \in F$	$\delta(s_{A6},ab) = s_{A7} \in F$	$\delta(s_{A6},ba) = s_{A7} \in F$	$\delta(s_{A6},bb) = s_{A7} \in F$
$\delta(s_{A4},aa) = s_{A7} \in F$	$\delta(s_{A4},ab) = s_{A7} \in F$	$\delta(s_{A4},ba) = s_{A7} \in F$	$\delta(s_{A4},bb) = s_{A7} \in F$
$\delta(s_{A5},aa) = s_{A7} \in F$	$\delta(s_{A5},ab) = s_{A7} \in F$	$\delta(s_{A5},ba) = s_{A7} \in F$	$\delta(s_{A5},bb) = s_{A7} \in F$
$\delta(s_{A7},aa) = s_{A7} \in F$	$\delta(s_{A7},ab) = s_{A7} \in F$	$\delta(s_{A7},ba) = s_{A7} \in F$	$\delta(s_{A7},bb) = s_{A7} \in F$

2-äquivalent sind die Zustände, die bei allen vier Transitionen $\delta(s_i,w)$ mit $w=aa$, $w=ab$, $w=ba$ und $w=bb$ das gleiche Folgezustandsmuster mit *Endzustand* oder *kein Endzustand* aufweisen und 1-äquivalent sind. Damit sind die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} 2-äquivalent. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{2}{\sim}$							
s_{A1}		$\overset{2}{\sim}$						
s_{A2}			$\overset{2}{\sim}$					
s_{A3}				$\overset{2}{\sim}$			$\overset{2}{\sim}$	
s_{A4}					$\overset{2}{\sim}$	$\overset{2}{\sim}$		
s_{A5}						$\overset{2}{\sim}$		
s_{A6}							$\overset{2}{\sim}$	
s_{A7}								$\overset{2}{\sim}$

Bei der 3-Äquivalenz $\overset{3}{\sim}$ müssen alle möglichen Zustandspaare (s_i, s_j) hinsichtlich $s_i^3 = \delta(s_i, w)$ und $s_j^3 = \delta(s_j, w)$ für alle $w \in E^*$ mit $|w| = 3$ untereinander bzgl. der Eigenschaft *Endzustand* oder *kein Endzustand* verglichen werden.

$\delta(s_{A0}, aaa) = s_{A4}$ $\delta(s_{A0}, baa) = s_{A7} \in F$	$\delta(s_{A0}, aab) = s_{A7} \in F$ $\delta(s_{A0}, bab) = s_{A7} \in F$	$\delta(s_{A0}, aba) = s_{A7} \in F$ $\delta(s_{A0}, bba) = s_{A5}$	$\delta(s_{A0}, abb) = s_{A7} \in F$ $\delta(s_{A0}, bbb) = s_{A7} \in F$
$\delta(s_{A1}, aaa) = s_{A7} \in F$ $\delta(s_{A1}, baa) = s_{A7} \in F$	$\delta(s_{A1}, aab) = s_{A7} \in F$ $\delta(s_{A1}, bab) = s_{A7} \in F$	$\delta(s_{A1}, aba) = s_{A7} \in F$ $\delta(s_{A1}, bba) = s_{A7} \in F$	$\delta(s_{A1}, abb) = s_{A7} \in F$ $\delta(s_{A1}, bbb) = s_{A7} \in F$
$\delta(s_{A2}, aaa) = s_{A7} \in F$ $\delta(s_{A2}, baa) = s_{A7} \in F$	$\delta(s_{A2}, aab) = s_{A7} \in F$ $\delta(s_{A2}, bab) = s_{A7} \in F$	$\delta(s_{A2}, aba) = s_{A7} \in F$ $\delta(s_{A2}, bba) = s_{A7} \in F$	$\delta(s_{A2}, abb) = s_{A7} \in F$ $\delta(s_{A2}, bbb) = s_{A7} \in F$
$\delta(s_{A3}, aaa) = s_{A7} \in F$ $\delta(s_{A3}, baa) = s_{A7} \in F$	$\delta(s_{A3}, aab) = s_{A7} \in F$ $\delta(s_{A3}, bab) = s_{A7} \in F$	$\delta(s_{A3}, aba) = s_{A7} \in F$ $\delta(s_{A3}, bba) = s_{A7} \in F$	$\delta(s_{A3}, abb) = s_{A7} \in F$ $\delta(s_{A3}, bbb) = s_{A7} \in F$
$\delta(s_{A6}, aaa) = s_{A7} \in F$ $\delta(s_{A6}, baa) = s_{A7} \in F$	$\delta(s_{A6}, aab) = s_{A7} \in F$ $\delta(s_{A6}, bab) = s_{A7} \in F$	$\delta(s_{A6}, aba) = s_{A7} \in F$ $\delta(s_{A6}, bba) = s_{A7} \in F$	$\delta(s_{A6}, abb) = s_{A7} \in F$ $\delta(s_{A6}, bbb) = s_{A7} \in F$
$\delta(s_{A4}, aaa) = s_{A7} \in F$ $\delta(s_{A4}, baa) = s_{A7} \in F$	$\delta(s_{A4}, aab) = s_{A7} \in F$ $\delta(s_{A4}, bab) = s_{A7} \in F$	$\delta(s_{A4}, aba) = s_{A7} \in F$ $\delta(s_{A4}, bba) = s_{A7} \in F$	$\delta(s_{A4}, abb) = s_{A7} \in F$ $\delta(s_{A4}, bbb) = s_{A7} \in F$
$\delta(s_{A5}, aaa) = s_{A7} \in F$ $\delta(s_{A5}, baa) = s_{A7} \in F$	$\delta(s_{A5}, aab) = s_{A7} \in F$ $\delta(s_{A5}, bab) = s_{A7} \in F$	$\delta(s_{A5}, aba) = s_{A7} \in F$ $\delta(s_{A5}, bba) = s_{A7} \in F$	$\delta(s_{A5}, abb) = s_{A7} \in F$ $\delta(s_{A5}, bbb) = s_{A7} \in F$
$\delta(s_{A7}, aaa) = s_{A7} \in F$ $\delta(s_{A7}, baa) = s_{A7} \in F$	$\delta(s_{A7}, aab) = s_{A7} \in F$ $\delta(s_{A7}, bab) = s_{A7} \in F$	$\delta(s_{A7}, aba) = s_{A7} \in F$ $\delta(s_{A7}, bba) = s_{A7} \in F$	$\delta(s_{A7}, abb) = s_{A7} \in F$ $\delta(s_{A7}, bbb) = s_{A7} \in F$

3-äquivalent sind die Zustände, die bei den acht Transitionen $\delta(s_i, w)$ mit $w = aaa$, $w = aab$, $w = aba$ und $w = abb$ $w = baa$, $w = bab$, $w = bba$ und $w = bbb$ das gleiche Folgezustandsmuster mit *Endzustand* oder *kein Endzustand* aufweisen und 2-äquivalent sind. Damit sind die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} 3-äquivalent. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{3}{\sim}$							
s_{A1}		$\overset{3}{\sim}$						
s_{A2}			$\overset{3}{\sim}$					
s_{A3}				$\overset{3}{\sim}$			$\overset{3}{\sim}$	
s_{A4}					$\overset{3}{\sim}$	$\overset{3}{\sim}$		
s_{A5}						$\overset{3}{\sim}$		
s_{A6}							$\overset{3}{\sim}$	
s_{A7}								$\overset{3}{\sim}$

Für die 3-Äquivalenz ergibt sich die gleiche Tabelle wie für die 2-Äquivalenz. Wenn wir nun noch die 4-Äquivalenz $\overset{4}{\sim}$ für betrachten würden, dann müssten wir alle möglichen Zustandspaare (s_i, s_j) hinsichtlich $s_i^4 = \delta(s_i, w)$ und $s_j^4 = \delta(s_j, w)$ für alle $w \in E^*$ mit $|w| = 4$ untereinander bzgl. der Eigenschaft *Endzustand* oder *kein Endzustand* vergleichen. Diesen Aufwand mit je 16 Transitionen pro Zustand wollen wir hier nicht mehr vorführen. Dies ist auch nicht notwendig, da man von jedem Zustand aus mit je vier beliebigen Transitionen, zusammengesetzt aus den Elementen a und b des Eingabealphabetes, immer im Endzustand s_{A7} ankommt, d.h. alle $s_i^4 = \delta(s_i, w)$ sind Endzustände. Die Entscheidung darüber ob zwei Zustände 4-äquivalent sind hängt also nur davon ab, ob diese vorher schon 3-äquivalent waren. Somit ergibt sich, dass die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} 4-äquivalent sind.

Die gleiche Argumentation lässt sich auf alle weiteren k-Äquivalenzen übertragen. Dies soll hier als Beweis ausreichend sein.

Es bleibt noch die Frage, welche Zustände *äquivalent* sind bzgl. der Definition 4.11. Diese besagt, dass zwei Zustände $s_i, s_j \in S$ äquivalent ($s_i \sim s_j$) zueinander sind wenn für alle $k \in \mathbb{N}_0$ gilt: $s_i \overset{k}{\sim} s_j$. Da wir mit der obigen Argumentation für alle $k \geq 3$ die gleiche Äquivalenzrelation (Tabelle der Äquivalenzen) erhalten würden wie für $k = 3$, folgt daraus: Für alle Zustandspaare, für die $s_i \overset{3}{\sim} s_j$ gilt, für die gilt auch $s_i \sim s_j$. Damit sind in EA_A die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} äquivalent.

b) k-äquivalente Zustände von EA_A und EA_B

Wir wollen k-äquivalente Zustände der beiden Automaten EA_A und EA_B entsprechend der Definition 4.10 finden. Wir beginnen wieder mit der trivialen 0-Äquivalenz $\overset{0}{\sim}$, bei der nur die Zustände selbst gegeneinander bzgl. der Eigenschaft *Endzustand* $s \in F$ oder *kein Endzustand* $s \notin F$ verglichen werden. Da nur die Zustände s_{A7} in EA_A und s_{B5} in EA_B Endzustände sind und alle anderen nicht, sind diese zueinander 0-äquivalent. Andererseits sind alle Nicht-Endzustände der beiden Automaten paarweise zueinander 0-äquivalent. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B1}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B2}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B3}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B4}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B5}								$\overset{0}{\sim}$

Für die Bestimmung der 1-Äquivalenz $\overset{1}{\sim}$ müssen nun für alle möglichen Zustandspaare (s_i, s_j) deren Folgezustände $s^1_{Ai} = \delta_A(s_{Ai}, w)$ und $s^1_{Bi} = \delta_B(s_{Bi}, w)$ für alle $w \in E^*$ mit $|w| = 1$ untereinander bzgl. der Eigenschaft *Endzustand* $s \in F$ oder *kein Endzustand* verglichen werden. Dazu stellen wir uns alle Folgezustände tabellarisch dar:

$\delta(s_{A0}, a) = s_{A1}$	$\delta(s_{A0}, b) = s_{A2}$
$\delta(s_{A1}, a) = s_{A3}$	$\delta(s_{A1}, b) = s_{A4}$
$\delta(s_{A2}, a) = s_{A5}$	$\delta(s_{A2}, b) = s_{A6}$
$\delta(s_{A3}, a) = s_{A4}$	$\delta(s_{A3}, b) = s_{A7} \in F$
$\delta(s_{A4}, a) = s_{A7} \in F$	$\delta(s_{A4}, b) = s_{A7} \in F$
$\delta(s_{A5}, a) = s_{A7} \in F$	$\delta(s_{A5}, b) = s_{A7} \in F$
$\delta(s_{A6}, a) = s_{A5}$	$\delta(s_{A6}, b) = s_{A7} \in F$
$\delta(s_{A7}, a) = s_{A7} \in F$	$\delta(s_{A7}, b) = s_{A7} \in F$

$\delta(s_{B0}, a) = s_{B1}$	$\delta(s_{B0}, b) = s_{B2}$
$\delta(s_{B1}, a) = s_{B4}$	$\delta(s_{B1}, b) = s_{B3}$
$\delta(s_{B2}, a) = s_{B3}$	$\delta(s_{B2}, b) = s_{B4}$
$\delta(s_{B3}, a) = s_{B5} \in F$	$\delta(s_{B3}, b) = s_{B5} \in F$
$\delta(s_{B4}, a) = s_{B3}$	$\delta(s_{B4}, b) = s_{B5} \in F$
$\delta(s_{B5}, a) = s_{B5} \in F$	$\delta(s_{B5}, b) = s_{B5} \in F$

In jeder Zeile stehen jeweils die möglichen 1-Übergänge eines Zustandes, links die des Automaten EA_A und rechts die des Automaten EA_B. 1-äquivalent sind die Zustände, die bei beiden Transitionen $\delta(s_i, w)$ mit $w = a$ und $w = b$ bei beiden Automaten das gleiche Folgezustandsmuster mit *Endzustand* $s \in F$ oder *kein Endzustand* aufweisen und 0-äquivalent sind.

Das gleiche Folgezustandsmuster weisen die Zustände s_{A0}, s_{B0}, s_{A1}, s_{B1} und s_{A2}, s_{B2} auf. Aufgrund der gegebenen 0-Äquivalenz sind die Zustände s_{A0}, s_{A1} und s_{A2} paarweise zu s_{B0}, s_{B1} und s_{B2} 1-äquivalent.

Das gleiche Folgezustandsmuster weisen auch die Zustände s_{A3}, s_{A6} und s_{B3} auf. Aufgrund der gegebenen 0-Äquivalenz sind die Zustände s_{A3} und s_{A6} paarweise zu s_{B4} 1-äquivalent. Die gleiche Argumentation gilt auch für die Zustände s_{A4}, s_{A5} und s_{B3}, s_{A4} und s_{A5} sind paarweise zu s_{B3} 1-äquivalent. Schließlich sind auch noch die Zustände s_{A7} und s_{B5} aufgrund des gleichen Folgezustandsmusters und der vorhandenen 0-Äquivalenz 1-äquivalent. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{B1}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{B2}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{B3}					$\overset{1}{\sim}$	$\overset{1}{\sim}$		
s_{B4}				$\overset{1}{\sim}$			$\overset{1}{\sim}$	
s_{B5}								$\overset{1}{\sim}$

Für die Bestimmung der 2-Äquivalenz $\overset{2}{\sim}$ müssen für alle möglichen Zustandspaare (s_i, s_j) deren Folgezustände $s^2_{Ai} = \delta_A(s_{Ai}, w)$ und $s^2_{Bi} = \delta_B(s_{Bi}, w)$ für alle $w \in E^*$ mit $|w| = 2$ untereinander bzgl. der Eigenschaft *Endzustand* $s \in F$ oder *kein Endzustand* verglichen werden.

$\delta(s_{A0},aa)=s_{A3}$	$\delta(s_{A0},ab)=s_{A4}$
$\delta(s_{A0},ba)=s_{A5}$	$\delta(s_{A0},bb)=s_{A6}$
$\delta(s_{A1},aa)=s_{A4}$	$\delta(s_{A1},ab)=s_{A7}$
$\delta(s_{A1},ba)=s_{A7}\in F$	$\delta(s_{A1},bb)=s_{A7}\in F$
$\delta(s_{A2},aa)=s_{A7}\in F$	$\delta(s_{A2},ab)=s_{A7}\in F$
$\delta(s_{A2},ba)=s_{A5}$	$\delta(s_{A2},bb)=s_{A7}\in F$
$\delta(s_{A3},aa)=s_{A7}\in F$	$\delta(s_{A3},ab)=s_{A7}\in F$
$\delta(s_{A3},ba)=s_{A7}\in F$	$\delta(s_{A3},bb)=s_{A7}\in F$
$\delta(s_{A4},aa)=s_{A7}\in F$	$\delta(s_{A4},ab)=s_{A7}\in F$
$\delta(s_{A4},ba)=s_{A7}\in F$	$\delta(s_{A4},bb)=s_{A7}\in F$
$\delta(s_{A5},aa)=s_{A7}\in F$	$\delta(s_{A5},ab)=s_{A7}\in F$
$\delta(s_{A5},ba)=s_{A7}\in F$	$\delta(s_{A5},bb)=s_{A7}\in F$
$\delta(s_{A6},aa)=s_{A7}\in F$	$\delta(s_{A6},ab)=s_{A7}\in F$
$\delta(s_{A6},ba)=s_{A7}\in F$	$\delta(s_{A6},bb)=s_{A7}\in F$
$\delta(s_{A7},aa)=s_{A7}\in F$	$\delta(s_{A7},ab)=s_{A7}\in F$
$\delta(s_{A7},ba)=s_{A7}\in F$	$\delta(s_{A7},bb)=s_{A7}\in F$

$\delta(s_{B0},aa)=s_{B4}$	$\delta(s_{B0},ab)=s_{B3}$
$\delta(s_{B0},ba)=s_{B3}$	$\delta(s_{B0},bb)=s_{B4}$
$\delta(s_{B1},aa)=s_{B3}$	$\delta(s_{B1},ab)=s_{B5}$
$\delta(s_{B1},ba)=s_{B5}\in F$	$\delta(s_{B1},bb)=s_{B5}\in F$
$\delta(s_{B2},aa)=s_{B5}\in F$	$\delta(s_{B2},ab)=s_{B5}\in F$
$\delta(s_{B2},ba)=s_{B3}$	$\delta(s_{B2},bb)=s_{B5}\in F$
$\delta(s_{B3},aa)=s_{B5}\in F$	$\delta(s_{B3},ab)=s_{B5}\in F$
$\delta(s_{B3},ba)=s_{B5}\in F$	$\delta(s_{B5},bb)=s_{B7}\in F$
$\delta(s_{B4},aa)=s_{B5}\in F$	$\delta(s_{B4},ab)=s_{B5}\in F$
$\delta(s_{B4},ba)=s_{B5}\in F$	$\delta(s_{B4},bb)=s_{B5}\in F$
$\delta(s_{B5},aa)=s_{B5}\in F$	$\delta(s_{B5},ab)=s_{B5}\in F$
$\delta(s_{B5},ba)=s_{B5}\in F$	$\delta(s_{B5},bb)=s_{B5}\in F$

Aus der Tabelle erkennen wir, dass der Zustand s_{A0} zu s_{B0} 2-äquivalent ist, der Zustand s_{A1} ist zu s_{B1} 2-äquivalent und der Zustand s_{A2} ist zu s_{B2} 2-äquivalent. Weiter sind die Zustände s_{A3} und s_{A6} paarweise zu s_{B4} 2-äquivalent, ebenso sind die Zustände s_{A4} und s_{A5} paarweise zu s_{B3} 2-äquivalent. Schließlich sind auch noch die Zustände s_{A7} und s_{B5} 2-äquivalent. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{2}{\sim}$							
s_{B1}		$\overset{2}{\sim}$						
s_{B2}			$\overset{2}{\sim}$					
s_{B3}					$\overset{2}{\sim}$	$\overset{2}{\sim}$		
s_{B4}				$\overset{2}{\sim}$			$\overset{2}{\sim}$	
s_{B5}								$\overset{2}{\sim}$

Für die Bestimmung der 3-Äquivalenz $\overset{2}{\sim}$ müssen für alle möglichen Zustandspaare (s_i,s_j) deren Folgezustände $s^3_{Ai}=\delta_A(s_{Ai},w)$ und $s^3_{Bi}=\delta_B(s_{Bi},w)$ für alle $w\in E^*$ mit $|w|=3$ untereinander bzgl. der Eigenschaft *Endzustand* $s\in F$ oder *kein Endzustand* verglichen werden.

$\delta(s_{A0},aaa)=s_{A4}$	$\delta(s_{A0},aab)=s_{A7}\in F$
$\delta(s_{A0},aba)=s_{A7}\in F$	$\delta(s_{A0},abb)=s_{A7}\in F$
$\delta(s_{A0},baa)=s_{A7}\in F$	$\delta(s_{A0},bab)=s_{A7}\in F$
$\delta(s_{A0},bba)=s_{A5}$	$\delta(s_{A0},bbb)=s_{A7}\in F$
$\delta(s_{A1},aaa)=s_{A7}\in F$	$\delta(s_{A1},aab)=s_{A7}\in F$
$\delta(s_{A1},aba)=s_{A7}\in F$	$\delta(s_{A1},abb)=s_{A7}\in F$
$\delta(s_{A1},baa)=s_{A7}\in F$	$\delta(s_{A1},bab)=s_{A7}\in F$
$\delta(s_{A1},bba)=s_{A7}\in F$	$\delta(s_{A1},bbb)=s_{A7}\in F$
$\delta(s_{A2},aaa)=s_{A7}\in F$	$\delta(s_{A2},aab)=s_{A7}\in F$
$\delta(s_{A2},aba)=s_{A7}\in F$	$\delta(s_{A2},abb)=s_{A7}\in F$
$\delta(s_{A2},baa)=s_{A7}\in F$	$\delta(s_{A2},bab)=s_{A7}\in F$
$\delta(s_{A2},bba)=s_{A7}\in F$	$\delta(s_{A2},bbb)=s_{A7}\in F$
$\delta(s_{A3},aaa)=s_{A7}\in F$	$\delta(s_{A3},aab)=s_{A7}\in F$
$\delta(s_{A3},aba)=s_{A7}\in F$	$\delta(s_{A3},abb)=s_{A7}\in F$
$\delta(s_{A3},baa)=s_{A7}\in F$	$\delta(s_{A3},bab)=s_{A7}\in F$
$\delta(s_{A3},bba)=s_{A7}\in F$	$\delta(s_{A3},bbb)=s_{A7}\in F$
$\delta(s_{A4},aaa)=s_{A7}\in F$	$\delta(s_{A4},aab)=s_{A7}\in F$
$\delta(s_{A4},aba)=s_{A7}\in F$	$\delta(s_{A4},abb)=s_{A7}\in F$
$\delta(s_{A4},baa)=s_{A7}\in F$	$\delta(s_{A4},bab)=s_{A7}\in F$
$\delta(s_{A4},bba)=s_{A7}\in F$	$\delta(s_{A4},bbb)=s_{A7}\in F$

$\delta(s_{B0},aaa)=s_{B3}$	$\delta(s_{B0},aab)=s_{B5}\in F$
$\delta(s_{B0},aba)=s_{B5}\in F$	$\delta(s_{B0},abb)=s_{B5}\in F$
$\delta(s_{B0},baa)=s_{B5}\in F$	$\delta(s_{B0},bab)=s_{B5}\in F$
$\delta(s_{B0},bba)=s_{B3}$	$\delta(s_{B0},bbb)=s_{B5}\in F$
$\delta(s_{B1},aaa)=s_{B5}\in F$	$\delta(s_{B1},aab)=s_{B5}\in F$
$\delta(s_{B1},aba)=s_{B5}\in F$	$\delta(s_{B1},abb)=s_{B5}\in F$
$\delta(s_{B1},baa)=s_{B5}\in F$	$\delta(s_{B1},bab)=s_{B5}\in F$
$\delta(s_{B1},bba)=s_{B5}\in F$	$\delta(s_{B1},bbb)=s_{B5}\in F$
$\delta(s_{B2},aaa)=s_{B5}\in F$	$\delta(s_{B2},aab)=s_{B5}\in F$
$\delta(s_{B2},aba)=s_{B5}\in F$	$\delta(s_{B2},abb)=s_{B5}\in F$
$\delta(s_{B2},baa)=s_{B5}\in F$	$\delta(s_{B2},bab)=s_{B5}\in F$
$\delta(s_{B2},bba)=s_{B5}\in F$	$\delta(s_{B2},bbb)=s_{B5}\in F$
$\delta(s_{B3},aaa)=s_{B5}\in F$	$\delta(s_{B3},aab)=s_{B5}\in F$
$\delta(s_{B3},aba)=s_{B5}\in F$	$\delta(s_{B3},abb)=s_{B5}\in F$
$\delta(s_{B3},baa)=s_{B5}\in F$	$\delta(s_{B3},bab)=s_{B5}\in F$
$\delta(s_{B3},bba)=s_{B5}\in F$	$\delta(s_{B3},bbb)=s_{B5}\in F$
$\delta(s_{B4},aaa)=s_{B5}\in F$	$\delta(s_{B4},aab)=s_{B5}\in F$
$\delta(s_{B4},aba)=s_{B5}\in F$	$\delta(s_{B4},abb)=s_{B5}\in F$
$\delta(s_{B4},baa)=s_{B5}\in F$	$\delta(s_{B4},bab)=s_{B5}\in F$
$\delta(s_{B4},bba)=s_{B5}\in F$	$\delta(s_{B4},bbb)=s_{B5}\in F$

$\delta(s_{A5}, aaa) = s_{A7} \in F$	$\delta(s_{A5}, aab) = s_{A7} \in F$
$\delta(s_{A5}, aba) = s_{A7} \in F$	$\delta(s_{A5}, abb) = s_{A7} \in F$
$\delta(s_{A5}, baa) = s_{A7} \in F$	$\delta(s_{A5}, bab) = s_{A7} \in F$
$\delta(s_{A5}, bba) = s_{A7} \in F$	$\delta(s_{A5}, bbb) = s_{A7} \in F$
$\delta(s_{A6}, aaa) = s_{A7} \in F$	$\delta(s_{A6}, aab) = s_{A7} \in F$
$\delta(s_{A6}, aba) = s_{A7} \in F$	$\delta(s_{A6}, abb) = s_{A7} \in F$
$\delta(s_{A6}, baa) = s_{A7} \in F$	$\delta(s_{A6}, bab) = s_{A7} \in F$
$\delta(s_{A6}, bba) = s_{A7} \in F$	$\delta(s_{A6}, bbb) = s_{A7} \in F$
$\delta(s_{A7}, aaa) = s_{A7} \in F$	$\delta(s_{A7}, aab) = s_{A7} \in F$
$\delta(s_{A7}, aba) = s_{A7} \in F$	$\delta(s_{A7}, abb) = s_{A7} \in F$
$\delta(s_{A7}, baa) = s_{A7} \in F$	$\delta(s_{A7}, bab) = s_{A7} \in F$
$\delta(s_{A7}, bba) = s_{A7} \in F$	$\delta(s_{A7}, bbb) = s_{A7} \in F$

$\delta(s_{B5}, aaa) = s_{B5} \in F$	$\delta(s_{B5}, aab) = s_{B5} \in F$
$\delta(s_{B5}, aba) = s_{B5} \in F$	$\delta(s_{B5}, abb) = s_{B5} \in F$
$\delta(s_{B5}, baa) = s_{B5} \in F$	$\delta(s_{B5}, bab) = s_{B5} \in F$
$\delta(s_{B5}, bba) = s_{B5} \in F$	$\delta(s_{B5}, bbb) = s_{B5} \in F$

Aus der Tabelle erkennen wir, dass der Zustand s_{A0} 3-äquivalent zu s_{B0} ist. Für alle anderen Zuständen kommt man mit je drei beliebigen Transitionen, zusammengesetzt aus den Elementen a und b des Eingabealphabetes, immer im Endzustand s_{A7} an, d.h. alle $s_i^4 = \delta(s_i, w)$ sind Endzustände. Die restlichen 3-äquivalenten Zustände bestimmen sich also vollständig aus der 2-Äquivalenz. Der Zustand s_{A1} ist 3-äquivalent zu s_{B1} und der Zustand s_{A2} ist 3-äquivalent zu s_{B2}. Weiter sind die Zustände s_{A3} und s_{A6} paarweise zu s_{B4} 3-äquivalent, ebenso sind die Zustände s_{A4} und s_{A5} paarweise zu s_{B3} 3-äquivalent. Schließlich sind auch noch die Zustände s_{A7} und s_{B5} 3-äquivalent. Dargestellt in Tabellenform:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{3}{\sim}$							
s_{B1}		$\overset{3}{\sim}$						
s_{B2}			$\overset{3}{\sim}$					
s_{B3}					$\overset{3}{\sim}$	$\overset{3}{\sim}$		
s_{B4}				$\overset{3}{\sim}$			$\overset{3}{\sim}$	
s_{B5}								$\overset{3}{\sim}$

Wenn wir nun noch die 4-Äquivalenz betrachten würden, dann würde sich dabei herausstellen das man von allen Zuständen aus mit je vier beliebigen Transitionen, zusammengesetzt aus den Elementen a und b des Eingabealphabetes, immer im Endzustand s_{A7} ankommt, d.h. alle $s_i^4 = \delta(s_i, w)$ sind Endzustände. Alle 4-äquivalenten Zustände bestimmen sich also vollständig aus der 3-Äquivalenz. Für die 4-Äquivalenz ergibt sich die gleiche Tabelle wie für die 3-Äquivalenz. Die gleiche Argumentation gilt analog für alle weiteren k-Äquivalenzen.

c) Äquivalente Zustände von EA_A und EA_B
Laut der Definition 4.11 sind zwei Zustände $s_i \in S_1$ und $s_j \in S_2$ äquivalent ($s_i \sim s_j$) zueinander wenn für alle $k \in \mathbb{N}_0$ gilt: $s_i \overset{k}{\sim} s_j$.
Da wir mit der obigen Argumentation für alle $k \geq 3$ die gleiche Äquivalenzrelation (Tabelle der Äquivalenzen) erhalten würden wie für $k = 3$, folgt daraus: Für alle Zustandspaare, für die $s_i \overset{3}{\sim} s_j$ gilt, für die gilt auch $s_i \sim s_j$. Damit sind die folgenden Zustände äquivalent:

$$
\begin{aligned}
&s_{A0} \sim s_{B0}\\
&s_{A1} \sim s_{B1}\\
&s_{A2} \sim s_{B2}\\
&s_{A3} \sim s_{A6} \sim s_{B4}\\
&s_{A4} \sim s_{A5} \sim s_{B3}\\
&s_{A7} \sim s_{B5}
\end{aligned}
$$

□

Um die Herleitung des Algorithmus nicht weiter zu unterbrechen haben wir ein weiteres Beispiel für Interessierte in den Anhang verschoben.

Das Beispiel zeigt, dass die Bestimmung der Äquivalenz von Zuständen aufgrund der Definition sehr aufwendig ist. Das weitere Ziel besteht also darin, ein effizientes systematisches Verfahren zur Bestimmung äquivalenter Zustände zu entwickeln. Dies soll uns dann im nächsten Schritt in die Lage

versetzen, sowohl die Anzahl der Zustände eines endlichen Automaten EA zu minimieren als auch die Äquivalenz zweier EAs abzuleiten.

Aufgrund der Definition gilt für die Äquivalenz zweier Zustände s_1 und s_2 zweier Automaten EA_1 und EA_2:

$$s_1 \sim s_2, \text{ wenn für alle } k \in \mathbb{N}_0 \text{ gilt: } s_1 \overset{k}{\sim} s_2$$

Wendet man diese Aussage auf die Anfangszustände s_{01} und s_{02} der zwei Automaten EA_1 und EA_2 an, dann folgt:

$$s_{01} \sim s_{02}, \text{ wenn für alle } k \in \mathbb{N}_0 \text{ gilt: } s_{01} \overset{k}{\sim} s_{02}$$

Sind die zwei Anfangszustände s_{01} und s_{02} äquivalent, also $s_{01} \sim s_{02}$, dann muss dies nach Definition auch für alle Folgezustände bis hin zu den Endzuständen gelten. Die beiden Automaten EA_1 und EA_2 werden also durch das gleiche Eingabewort w in zwei äquivalente Endzustände $(s_1 \in S_1) \sim (s_2 \in S_2)$ gebracht. Wir können also unmittelbar die folgende Folgerung ableiten:

Satz 4.1 *Äquivalenz zweier Automaten*
Zwei endliche Automaten $EA_1 = \{E, S_1, \delta_1, s_{01}, F_1\}$ und $EA_2 = \{E, S_2, \delta_2, s_{02}, F_2\}$ mit demselben Eingabealphabet E akzeptieren dieselbe Sprache $L(EA_1) = L(EA_2)$, wenn die Anfangszustände s_{01} und s_{02} äquivalent sind, also $s_{01} \sim s_{02}$.
Zwei endliche Automaten EA_1 und EA_2 sind äquivalent, $EA_1 \sim EA_2$, wenn $s_{01} \sim s_{02}$ gilt.

4.5.2 Die Bildung von Äquivalenzklassen

Die Minimierung der Zustandsmenge betrachtet einen Automaten $EA = \{E, S, \delta, s_0, F\}$, der Zustandspaare (s_i, s_j) enthält, die zueinander äquivalent sind, d.h. für die gilt $s_i \sim s_j$. Von den beiden zueinander äquivalenten Zuständen kann einer aus dem Automaten entfernt werden, ohne dass sich das Verhalten des Automaten bzw. sich die akzeptierte Sprache verändert.

Das Verfahren zur Minimierung der Zustandsmenge erfolgt mit Hilfe von Zerlegungen der Zustandsmenge S in Teilmengen mit äquivalenten Zuständen basierend auf den Äquivalenzrelationen für Zustände. Durch diese Zerlegungen entstehen Äquivalenzklassen $[s]$ von Zuständen, also Mengen mit äquivalenten Zuständen s' zu einem bestimmten Zustand s.

$$[s] = \{s' \in S \mid s' \sim s\}$$

Wir gehen von einer Zustandsmenge S eines endlichen Automaten EA aus, auf der die Relation k-äquivalenter Zustände $\overset{k}{\sim}$ definiert ist. Die Zerlegung der Zustandsmenge S in Äquivalenzklassen $[s_k]$ durch die Äquivalenzrelation $\overset{k}{\sim}$ sei gegeben durch

$$\begin{aligned} [s_k] &= \{s' \in S \mid s' \overset{k}{\sim} s\} \quad \text{für } k \in \mathbb{N} \\ P_k &= \{[s_k] \mid s \in S\} \end{aligned}$$

Die Zerlegung P_k stellt die Menge der Äquivalenzklassen dar, diese Zerlegung ist um so feiner, je größer k ist. Für $k_2 \geq k_1$ ist die Zerlegung P_{k2} feiner als die Zerlegung P_{k1}, $P_{k2} \prec P_{k1}$, oder gleichwertig.

Beispiel 4.35 *Zerlegung in Äquivalenzklassen*
Gegeben sei die folgende Zerlegung P_k in Äquivalenzklassen $P_k = \{\{s_1, s_2, s_3\}, \{s_4, s_5\}\}$.
Die Zerlegung $P_{k+1} = \{\{s_1, s_2\}, \{s_3\}, \{s_4, s_5\}\}$ ist eine feinere Zerlegung von P_k. Dabei sind die Äquivalenzklassen von P_{k+1} Teilmengen der Äquivalenzklassen von P_k.
Die Zerlegung $\{\{s_1, s_4\}, \{s_2, s_3\}, \{s_5\}\}$ hingegen ist keine feinere Zerlegung von P_k, da die Äquivalenzklassen keine Teilmengen der Äquivalenzklassen von P_k sind. □

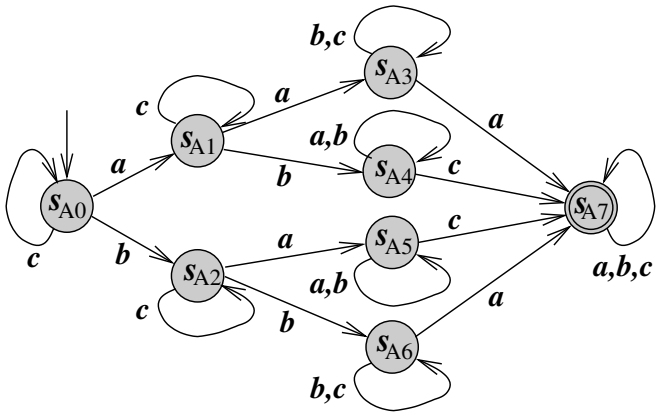
Da die Äquivalenz $\sim$ gilt, wenn die k-Äquivalenz $\overset{k}{\sim}$ für alle $k \in \mathbb{N}_0$ gilt, folgt für die Zerlegung der Zustandsmenge S in Äquivalenzklassen $[s]$ durch die Äquivalenzrelation $\sim$:

$$P \prec P_k \quad \text{für } k \in \mathbb{N}$$

Die Äquivalenz $\sim$ erzeugt die feinste Zerlegung, deren Anzahl an Äquivalenzklassen der minimalen Anzahl an Zuständen des EA entspricht.

Wenn wir nun eine solche Zerlegung in Äquivalenzklassen $P_k = \{[s_k]\}$ durchführen, dann haben wir die feinste Zerlegung erreicht, wenn sich die Äquivalenzklassen für das nächsthöhere k gegenüber den Äquivalenzklassen für $k-1$ nicht mehr verändern, wenn also $P_k = P_{k-1}$ ist. Dieser Punkt ist offensichtlich spätestens dann erreicht, wenn es so viele Äquivalenzklassen wie Zustände gibt. In diesem Fall enthält der EA keine äquivalenten Zustände und ist damit nicht reduzierbar. Wird die feinste Zerlegung für $k<n$ erreicht, damit gibt es mindestens eine Äquivalenzklasse, die mehr als einen Zustand enthält. Dieser EA ist um die in einer Äquivalenzklasse mehrfach enthaltenen Zustände reduzierbar ohne das Verhalten zu verändern. Diese Erkenntnis kann für die Entwicklung eines Algorithmus zur Reduzierung der Zustände eines EA genutzt werden.

Beispiel 4.36 *Zerlegung in Äquivalenzklassen*
Die Zustände des Automaten EA_A aus Beispiel 4.34 sollen in Äquivalenzklassen zerlegt werden.



Für die einzelnen Zerlegungen gilt:

$$\begin{aligned} P_0 &= \{\{S_{A0},S_{A1},S_{A2},S_{A3},S_{A4},S_{A5},S_{A6}\},\{S_{A7}\}\} \\ P_1 &= \{\{S_{A0},S_{A1},S_{A2}\},\{S_{A3},S_{A6}\},\{,S_{A4},S_{A5}\},\{S_{A7}\}\} \\ P_2 &= \{\{S_{A0}\},\{S_{A1}\},\{S_{A2}\},\{S_{A3},S_{A6}\},\{,S_{A4},S_{A5}\},\{S_{A7}\}\} = P \end{aligned}$$

□

4.5.3 Bestimmungsverfahren für die Äquivalenz von Zuständen

Bei einem reduzierten Automaten werden alle Zustände tatsächlich benötigt, d.h. man kann keine Zustände entfernen, ohne die Sprache des Automaten zu verändern. Wir wollen zuerst in einer Definition festhalten, was wir unter der Reduktion eines EA genau verstehen.

Definition 4.12 *Reduzierter EA*
Ein endlicher Automat $EA = (E,S,\delta,s_0,F)$ heisst reduziert, wenn er vereinfacht ist und alle Zustände paarweise nicht äquivalent zueinander sind.

Die folgende Betrachtung bezieht sich auf zwei Automaten $EA_1 = \{E,S_1,\delta_1,s_{01},F_1\}$ und $EA_2 = \{E,S_2,\delta_2,s_{02},F_2\}$ mit demselben Eingabealphabet E. Dabei besitze der eine Automat EA_1 äquivalente Zustandspaare (s_i,s_j), von denen ein Zustand s_i oder s_j eliminiert werden kann, so dass letzlich der reduzierte äquivalente Automat EA_2 entsteht. Zu diesem Zweck wollen wir einen Algorithmus entwickeln, der es uns ermöglicht, eventuell vorhandene äquivalente Zustände eines Automaten zu erkennen. Damit können wir anschließend den zum betrachteten Automaten EA_1 äquivalenten reduzierten Automaten EA_2 konstruieren. Ausgangspunkt ist die Definition 4.10, in der zwei k-äquivalente Zustände der Automaten $EA_1 = \{E,S_1,\delta_1,s_{01},F_1\}$ und $EA_2 = \{E,S_2,\delta_2,s_{02},F_2\}$ mit demselben Eingabealphabet E wie folgt bestimmt sind:
Zwei Zustände $s_1 \in S_1$ und $s_2 \in S_2$ sind k-äquivalent wenn für alle $w \in E^*$ mit $|w| \leq k$ gilt: $s_1^k = \delta_1(s_1,w)$ und $s_2^k = \delta_2(s_2,w)$ sind entweder beide Endzustände oder sie sind beide keine Endzustände.

$$\begin{aligned} s_1 \overset{k}{\sim} s_2 &\iff \forall w \in E^*, |w| \leq k : \delta_1(s_1,w) \text{ und } \delta_2(s_2,w) \text{ sind entweder beide Endzustände,} \\ &\qquad\qquad \text{oder sie sind beide keine Endzustände} \\ &\iff \forall w \in E^*, |w| \leq k : [\delta_1(s_1,w) \in F_1 \wedge \delta_2(s_2,w) \in F_2] \\ &\qquad\qquad \vee [\delta_1(s_1,w) \notin F_1 \wedge \delta_2(s_2,w) \notin F_2] \end{aligned} \tag{4.1}$$

Daraus kann man insbesondere für $k=0$, d.h. es findet keine Transition statt und die Aussage bezieht sich auf die zwei Zustände s_1 und s_2 selbst, folgern:

$$\begin{aligned} s_1 \overset{0}{\sim} s_2 &\iff \forall w \in E^*, |w| \leq 0 : [s_1 \in F_1 \wedge s_2 \in F_2] \vee [s_1 \notin F_1 \wedge s_2 \notin F_2] \\ &\iff [s_1 \in F_1 \wedge s_2 \in F_2] \vee [s_1 \notin F_1 \wedge s_2 \notin F_2] \end{aligned}$$

Zwei Zustände $s_1 \in S_1$ und $s_2 \in S_2$ sind 0-äquivalent, wenn entweder beide Endzustände oder beide keine Endzustände sind. Wendet man auf beiden Seiten der Äquivalenz $s_1 \overset{0}{\sim} s_2$ auf die Zustände s_1 und s_2 die Übergangsfunktion δ an, dann folgt daraus weiter:

$$\delta_1(s_1,w) \overset{0}{\sim} \delta_2(s_2,w) \iff [\delta_1(s_1,w) \in F_1 \wedge \delta_2(s_2,w) \in F_2] \vee [\delta_1(s_1,w) \notin F_1 \wedge \delta_2(s_2,w) \notin F_2]$$

Wir erkennen, dass sich auf der rechten Seite von $\iff$ genau die Aussage wiederfindet, die in der Definition (Gl. 4.1) zweier k-äquivalenter Zustände $s_1 \overset{k}{\sim} s_2$ steht. Wir können diesen Teil in der Definition ersetzen und es folgt:

$$\begin{aligned} s_1 \overset{k}{\sim} s_2 &\iff \forall w \in E^*, |w| \leq k : \delta_1(s_1,w) \in F_1 \wedge \delta_2(s_2,w) \in F_2 \vee \delta_1(s_1,w) \notin F_1 \wedge \delta_2(s_2,w) \notin F_2 \\ &\iff \forall w \in E^*, |w| \leq k : \delta_1(s_1,w) \overset{0}{\sim} \delta_2(s_2,w) \end{aligned}$$

Mit dieser Definitionsgleichung haben wir die Basis geschaffen für die rekursive Berechnung der k-Äquivalenz von Zuständen. Ersetzen wir die k-Äquivalenz durch die $(k+1)$-Äquivalenz und verwenden dabei ein Wort v der Länge $(k+1)$, dann gilt natürlich auch:

$$\begin{aligned} s_1 \overset{k}{\sim} s_2 &\iff \forall w \in E^*, |w| \leq k : \delta_1(s_1,w) \overset{0}{\sim} \delta_2(s_2,w) \\ s_1 \overset{k+1}{\sim} s_2 &\iff \forall v \in E^*, |v| \leq k+1 : \delta_1(s_1,v) \overset{0}{\sim} \delta_2(s_2,v) \end{aligned} \tag{4.2}$$

4.5.4 Minimierung der Zustandsmenge S eines Automaten

Da wir uns bei der Zustandsreduktion auf ein und denselben Automaten EA beziehen, gilt insbesondere $\delta_1 = \delta_2 = \delta$. Ebenso sind natürlich auch die beiden Zustände s_1 und s_2 Zustände ein und desselben Automaten EA, $s_1, s_2 \in S$. Damit folgt aus Gl. 4.2:

$$s_1 \overset{k+1}{\sim} s_2 \iff \forall v \in E^*, |v| \leq k+1 : \delta(s_1,v) \overset{0}{\sim} \delta(s_2,v) \tag{4.3}$$

Die dem folgenden induktiven Verfahren zur Bestimmung der Äquivalenz zweier Automaten liegt die Idee zugrunde, von einer k-Äquivalenz auf eine $(k+1)$-Äquivalenz zu schließen. Eine dazu notwendige Rekursionsgleichung liegt vor, wenn wir die $(k+1)$-Äquivalenz durch die k-Äquivalenz ausdrücken können. Zu diesem Zweck kann die Zeichenreihe $v \in E^*$ in Gl. 4.3 in die zwei Teile $e \in E$ und $w \in E^*$ aufgespalten werden. Da dabei die leere Zeichenreihe ε in $e \in E$ nicht enthalten ist, muss eine entsprechende Teilaussage ergänzt werden:

$$\begin{aligned} s_1 \overset{k+1}{\sim} s_2 &\iff [\forall e \in E, \forall w \in E^*, |w| \leq k : \delta(s_1,ew) \overset{0}{\sim} \delta(s_2,ew)] \wedge [\delta(s_1,\varepsilon) \overset{0}{\sim} \delta(s_2,\varepsilon)] \\ s_1 \overset{k+1}{\sim} s_2 &\iff [\forall e \in E, \forall w \in E^*, |w| \leq k : \delta(s_1,ew) \overset{0}{\sim} \delta(s_2,ew)] \wedge [s_1 \overset{0}{\sim} s_2] \end{aligned}$$

Durch einfache Umformung mit $\delta(s,ew) = \delta(\delta(s,e),w)$ folgt weiter:

$$s_1 \overset{k+1}{\sim} s_2 \iff [\forall e \in E, \forall w \in E^*, |w| \leq k : \delta(\delta(s_1,e),w) \overset{0}{\sim} \delta(\delta(s_2,e),w)] \wedge [s_1 \overset{0}{\sim} s_2] \tag{4.4}$$

Das Ziel ist es, die $(k+1)$-Äquivalenz durch die k-Äquivalenz auszudrücken. Um diese Gleichung weiter in Richtung Ziel umzuformen, nehmen wir die Gleichung (4.2) zu Hilfe. Diese lautet:

$$s_1 \overset{k}{\sim} s_2 \iff \forall w \in E^*, |w| \leq k : \delta_1(s_1,w) \overset{0}{\sim} \delta_2(s_2,w)$$

Da wir uns auf ein und denselben Automaten EA beziehen, gilt auch hier $\delta_1 = \delta_2 = \delta$ und $s_1, s_2 \in S$:

$$s_1 \overset{k}{\sim} s_2 \iff \forall w \in E^*, |w| \leq k : \ \delta(s_1, w) \overset{0}{\sim} \delta(s_2, w)$$

Da $s_1, s_2 \in S$ in dieser Gleichung allgemein für einen beliebigen Zustand des EA stehen, können wir s_1 auch durch $\delta(s_1, e)$ und s_2 durch $\delta(s_2, e)$ ersetzen:

$$\delta(s_1, e) \overset{k}{\sim} \delta(s_2, e) \iff \forall w \in E^*, |w| \leq k : \ \delta(\delta(s_1, e), w) \overset{0}{\sim} \delta(\delta(s_2, e), w)$$

Dies hat uns nun einen Schritt weitergebracht, denn die rechte Seite dieser Gleichung tritt genau so auf der rechten Seite unserer Gleichung 4.4 für $s_1 \overset{k+1}{\sim} s_2$ auf und wir können diese durch die linke Seite ersetzen:

$$\begin{aligned} s_1 \overset{k+1}{\sim} s_2 &\iff [\forall e \in E, \underbrace{\forall w \in E^*, |w| \leq k : \delta(\delta(s_1, e), w) \overset{0}{\sim} \delta(\delta(s_2, e), w)}_{\delta(s_1, e) \overset{k}{\sim} \delta(s_2, e)}] \wedge [s_1 \overset{0}{\sim} s_2] \\ &\iff [\forall e \in E : \delta(s_1, e) \overset{k}{\sim} \delta(s_2, e)] \wedge [s_1 \overset{0}{\sim} s_2] \end{aligned}$$

Damit haben wir das erste Ziel erreicht, wir können die $(k+1)$-Äquivalenz vollständig durch die k-Äquivalenz ausdrücken. Wir können induktiv aus der k-Äquivalenz zweier Zustände $s_1, s_2 \in S$ auf die $(k+1)$-Äquivalenz zweier Vorgängerzustände schließen.

Wir werden jedoch später sehen, dass der umgekehrte Weg, nämlich von der Nicht-k-Äquivalenz zweier Folgezustände $\delta(s_1, e)$ und $\delta(s_2, e)$ auf die Nicht-$(k+1)$-Äquivalenz der Zustände s_1 und s_2 zu schließen, praktisch einfacher auszuführen ist. Um das entsprechende Lösungsverfahren herzuleiten, müssen wir die Rekursionsgleichung negieren. Ein direkte Negation führt aber nicht zum Ziel, da sich aus der Negation der Äquivalenz eine Antivalenz ergibt. Für das Lösungsverfahren wird jedoch nur der Schluss von k nach $(k+1)$ benötigt, d.h. die Gültigkeit der Rekursionsgleichung wird nur in eine Richtung benötigt. Wir können daher einfach die Äquivalenz $\iff$ durch eine Implikation $\Longrightarrow$ ersetzen und die entstandene Aussage negieren:

$$\underbrace{s_1 \overset{k+1}{\sim} s_2}_{A} \implies \underbrace{\forall e \in E : \delta(s_1, e) \overset{k}{\sim} \delta(s_2, e) \wedge s_1 \overset{0}{\sim} s_2}_{B}$$

Die aussagenlogische Implikation $A \Longrightarrow B$ ist nur genau dann *falsch*, wenn die Prämisse A *richtig* und die Folgerung B *falsch* ist. Ansonsten ist die Implikation immer *richtig*. Die folgende Wahrheitstabelle gibt uns den Zusammenhang zwischen der Implikation $A \Longrightarrow B$ und der benötigten Negation unserer Rekursion:

A	B	$X = (A \Longrightarrow B)$	$X = \neg A \vee B$	$X = (\neg A \Longleftarrow \neg B)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

$$(A \Longrightarrow B) = (\neg A \Longleftarrow \neg B)$$

Damit folgt für unsere Rekursionslösung:

$$s_1 \overset{k+1}{\sim} s_2 \implies [\forall e \in E : \delta(s_1, e) \overset{k}{\sim} \delta(s_2, e)] \wedge [s_1 \overset{0}{\sim} s_2]$$

$$= \neg \left(s_1 \overset{k+1}{\sim} s_2 \right) \Longleftarrow \neg \left([\forall e \in E : \delta(s_1, e) \overset{k}{\sim} \delta(s_2, e)] \wedge [s_1 \overset{0}{\sim} s_2] \right)$$

Weiter wird durch die Negation zweier UND-verknüpfter Teilaussagen, wie wir sie auf der rechten Seite der Gleichung wiederfinden, nach der DeMorgan-Regel eine ODER-Verknüpfung der negierten Teilaussagen:

$$\neg(A \wedge B) = \neg A \vee \neg B$$

Durch die Ausführung dieser Operationen wird aus der k-Äquivalenz die Nicht-k-Äquivalenz:

$$s_1 \overset{k+1}{\not\sim} s_2 \Longleftarrow \neg \left(\forall e \in E : \delta(s_1, e) \overset{k}{\sim} \delta(s_2, e) \right) \vee \neg \left(s_1 \overset{0}{\sim} s_2 \right)$$

Durch die Negation wird in der ersten Teilaussage der rechten Seite aus der Allquantifizierung $\forall$ eine Existenzquantifizierung $\exists$:

$$s_1 \overset{k+1}{\not\sim} s_2 \iff [\exists e \in E : \delta(s_1,e) \overset{k}{\not\sim} \delta(s_2,e)] \vee [s_1 \overset{0}{\not\sim} s_2]$$

Diese Aussage müssen wir nur noch anders herum aufschreiben, damit wir sie in der Form haben, in der wir sie anwenden wollen:

$$s_1 \overset{0}{\not\sim} s_2 \vee \exists e \in E : \delta(s_1,e) \overset{k}{\not\sim} \delta(s_2,e) \Longrightarrow s_1 \overset{k+1}{\not\sim} s_2 \tag{4.5}$$

Damit können wir nun tatsächlich von der Nicht-k-Äquivalenz zweier Folgezustände $\delta(s_1,e)$ und $\delta(s_2,e)$ auf die Nicht-$(k+1)$-Äquivalenz der Zustände s_1 und s_2 schließen. Der Algorithmus liefert als Ergebnis die äquivalenten Zustandspaare eines Automaten $EA_1 = \{E, S_1, \delta_1, s_{01}, F_1\}$, mit dem wir den zum Automaten EA_1 reduzierten Automaten $EA_2 = \{E, S_2, \delta_2, s_{02}, F_2\}$ bestimmen können.

Zur praktischen Ausführung der Ermittlung der äquivalenten Zustände verwenden wir ein Dreiecksschema, ähnlich wie die Übergangstabelle. Darin sehen wir für jedes Zustandspaar (s_i, s_j) des zu reduzierenden Automaten EA_1 mit $i \neq j$ einen Eintrag vor:

s_1						
s_2						
s_3						
s_4						
…						
s_n						
	s_0	s_1	s_2	s_3	…	s_{n-1}

In diesem Schema wird schrittweise jedes Kästchen zu einem Paar (s_i, s_j) mit einem Kreuz markiert, für das $\delta(s_1,e) \overset{k+1}{\not\sim} \delta(s_2 e)$ aufgrund der Gültigkeit von $s_i \overset{k}{\not\sim} s_j$ nicht gelten kann.
Wenn sich nach endlich vielen Iterationen keine Änderung mehr ergibt, geben die nichtmarkierten Kästchen genau die äquivalenten Zustandspaare an.

Der Algorithmus läuft folgendermaßen ab:

Im ersten Schritt werden alle Zustandspaare (s_i, s_j) ermittelt, die nicht 0-äquivalent ($s_i \overset{0}{\not\sim} s_j$) sein können. Dies ist genau dann der Fall, wenn der eine Zustand ein Endzustand ist und der andere nicht. Die nicht 0-äquivalenten Paare werden durch ein X_0 markiert.

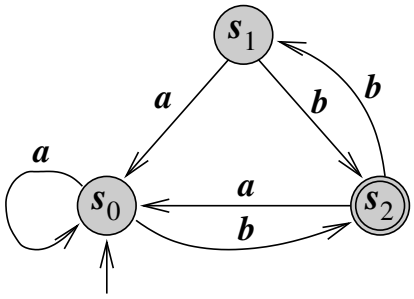
Im zweiten Schritt werden alle Zustandspaare (s_i, s_j) ermittelt, die nicht 1-äquivalent ($s_i \overset{1}{\not\sim} s_j$) sein können. Dabei müssen nur noch die Paare überprüft werden, die im ersten Schritt keine Markierung erhalten haben. Das Zustandspaar (s_i, s_j) ist nicht 1-äquivalent, wenn das Folgezustandspaar zu einer Eingabe bereits als nicht 0-äquivalent markiert ist. Die nicht 1-äquivalenten Paare werden durch ein X_1 markiert.

Im dritten Schritt werden alle Zustandspaare (s_i, s_j) ermittelt, die nicht 2-äquivalent ($s_i \overset{2}{\not\sim} s_j$) sein können. Dies ist der Fall, wenn das Folgezustandspaar zu einer Eingabe bereits als nicht 0- oder 1-äquivalent markiert ist. Die nicht 2-äquivalenten Paare werden durch ein X_2 markiert.

Das Verfahren wird weiter fortgesetzt, im k-ten Schritt werden alle Zustandspaare (s_i, s_j) ermittelt, die nicht $(k-1)$-äquivalent sein können, usw.

Das Verfahren setzt genau die obige Aussage um, dass falls es ein Eingabezeichen $e \in E$ gibt, so dass $\delta(s_1,e) \overset{k}{\not\sim} \delta(s_2,e)$ gilt, dann folgt daraus $s_1 \overset{k+1}{\not\sim} s_2$. Die Berechnung nicht äquivalenter Zustände erfolgt also induktiv durch das Schließen von k auf $(k+1)$. Der Algorithmus bricht ab, falls in einen Schritt keine weitere Markierung mehr entsteht.

Beispiel 4.37 *Äquivalente Zustände*
Gegeben sei der endliche Automat EA$=(S,E,\delta,s_0,F)$, der durch $S=\{s_0,s_1,s_2\}$, $E=\{a,b\}$, $F=\{s_2\}$ und die folgende Übergangsfunktion δ beschrieben ist.



δ	a	b
s_0	s_0	s_2
s_1	s_0	s_2
s_2	s_0	s_1

Gesucht sind äquivalente Zustände in diesem Automaten.

Im 1. Schritt bekommen alle Zustandspaare (s_i,s_j), bei denen der eine ein Endzustand ist und der andere nicht, eine Markierung mit X_0.

$$s_i \overset{0}{\sim} s_j \iff s_i\in F \wedge s_j\in F \vee s_i\notin F \wedge s_j\notin F$$

s_1		
s_2	X_0	X_0
	s_0	s_1

(s_0,s_1) $s_0\notin F$, $s_1\notin F$, $\Rightarrow$ keine Markierung
(s_0,s_2) $s_0\notin F$, $s_2\in F$, $\Rightarrow$ Markierung von (s_0,s_2) mit X_0
(s_1,s_2) $s_0\notin F$, $s_2\in F$, $\Rightarrow$ Markierung von (s_1,s_2) mit X_0

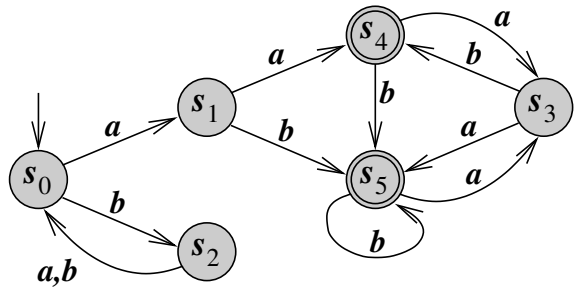
Im 2. Schritt bekommen alle Zustandspaare (s_i,s_j) eine Markierung X_1, bei denen das Folgezustandspaar $\delta(s_i,e) \overset{0}{\not\sim} \delta(s_j,e)$ für ein beliebiges $e\in E$ bereits eine Markierung X_0 besitzt. Dabei müssen nur die noch nicht markierten Zustandspaare (s_i,s_j) untersucht werden.

$$s_i \overset{0}{\not\sim} s_j \vee \exists e\in E : \delta(s_i,e) \overset{0}{\not\sim} \delta(s_j,e) \Longrightarrow s_i \overset{1}{\not\sim} s_j$$

(s_0,s_1) $(\delta(s_0,a),\delta(s_1,a))=(s_0,s_0)$, $(\delta(s_0,b),\delta(s_1,b))=(s_2,s_2)$, $\Rightarrow$ keine Markierung X_1

Da im 2. Schritt keine weitere Markierung entsteht, ist das Verfahren bereits abgeschlossen.
Äquivalent ist das Zustandspaar (s_0,s_1). □

Beispiel 4.38 *Äquivalente Zustände*
Gegeben sei der endliche Automat EA$=(S,E,\delta,s_0,F)$, der durch $S=\{s_0,s_1,s_2,s_3,s_4,s_5\}$, $E=\{a,b\}$, $F=\{s_4,s_5\}$ und die folgende Übergangsfunktion δ beschrieben ist.



δ	a	b
s_0	s_1	s_2
s_1	s_4	s_5
s_2	s_0	s_0
s_3	s_5	s_4
s_4	s_3	s_5
s_5	s_3	s_5

Gesucht sind äquivalente Zustände in diesem Automaten.

1. Schritt

$$s_i \overset{0}{\sim} s_j \Longrightarrow s_i\in F \wedge s_j\in F \vee s_i\notin F \wedge s_j\notin F$$

(s_0,s_1) $s_0\notin F,\ s_1\notin F,\ \Rightarrow$ keine Markierung X_0
(s_0,s_2) $s_0\notin F,\ s_2\notin F,\ \Rightarrow$ keine Markierung X_0
(s_0,s_3) $s_0\notin F,\ s_3\notin F,\ \Rightarrow$ keine Markierung X_0
(s_0,s_4) $s_0\notin F,\ s_4\in F,\ \Rightarrow$ Markierung von (s_0,s_4) mit X_0
(s_0,s_5) $s_0\notin F,\ s_5\in F,\ \Rightarrow$ Markierung von (s_0,s_5) mit X_0
(s_1,s_2) $s_1\notin F,\ s_2\notin F,\ \Rightarrow$ keine Markierung X_0
(s_1,s_3) $s_1\notin F,\ s_3\notin F,\ \Rightarrow$ keine Markierung X_0
(s_1,s_4) $s_1\notin F,\ s_4\in F,\ \Rightarrow$ Markierung von (s_1,s_4) mit X_0

(s_1,s_5) $s_1 \notin F,\ s_5 \in F,\ \Rightarrow$ Markierung von (s_1,s_5) mit X_0
(s_2,s_3) $s_2 \notin F,\ s_3 \notin F,\ \Rightarrow$ keine Markierung X_0
(s_2,s_4) $s_2 \notin F,\ s_4 \in F,\ \Rightarrow$ Markierung von (s_2,s_4) mit X_0
(s_2,s_5) $s_2 \notin F,\ s_5 \in F,\ \Rightarrow$ Markierung von (s_2,s_5) mit X_0
(s_3,s_4) $s_3 \notin F,\ s_4 \in F,\ \Rightarrow$ Markierung von (s_3,s_4) mit X_0
(s_3,s_5) $s_3 \notin F,\ s_5 \in F,\ \Rightarrow$ Markierung von (s_3,s_5) mit X_0
(s_4,s_5) $s_4 \in F,\ s_5 \in F,\ \Rightarrow$ keine Markierung X_0

s_1					
s_2					
s_3					
s_4	X_0	X_0	X_0	X_0	
s_5	X_0	X_0	X_0	X_0	
	s_0	s_1	s_2	s_3	s_4

s_1	X_1				
s_2	X_2	X_1			
s_3	X_1		X_1		
s_4	X_0	X_0	X_0	X_0	
s_5	X_0	X_0	X_0	X_0	
	s_0	s_1	s_2	s_3	s_4

2. Schritt $s_i \overset{0}{\not\sim} s_j \vee \exists e \in E : \delta(s_i,e) \overset{0}{\not\sim} \delta(s_j,e) \Longrightarrow s_i \overset{1}{\not\sim} s_j$

(s_i,s_j)	$(\delta(s_i,a),\delta(s_j,a))$	$(\delta(s_i,b),\delta(s_j,b))$	Markierung	wegen
(s_0,s_1)	(s_1,s_4)	(s_2,s_5)	X_1	$(s_1,s_4),\ (s_2,s_5)$
(s_0,s_2)	(s_1,s_0)	(s_2,s_0)		
(s_0,s_3)	(s_1,s_5)	(s_2,s_4)	X_1	$(s_1,s_5),\ (s_2,s_4)$
(s_1,s_2)	(s_4,s_0)	(s_5,s_0)	X_1	$(s_4,s_0),\ (s_5,s_0)$
(s_1,s_3)	(s_4,s_5)	(s_5,s_4)		
(s_2,s_3)	(s_0,s_5)	(s_0,s_4)	X_1	$(s_0,s_5),\ (s_0,s_4)$
(s_4,s_5)	(s_3,s_3)	(s_5,s_5)		

3. Schritt $s_i \overset{0}{\not\sim} s_j \vee \exists e \in E : \delta(s_i,e) \overset{1}{\not\sim} \delta(s_j,e) \Longrightarrow s_i \overset{2}{\not\sim} s_j$

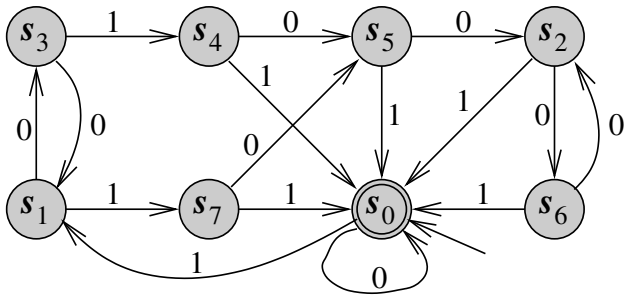
(s_i,s_j)	$(\delta(s_i,a),\delta(s_j,a))$	$(\delta(s_i,b),\delta(s_j,b))$	Markierung	wegen
(s_0,s_2)	(s_1,s_0)	(s_2,s_0)	X_2	(s_1,s_0)
(s_1,s_3)	(s_4,s_5)	(s_5,s_4)		
(s_4,s_5)	(s_3,s_3)	(s_5,s_5)		

4. Schritt $s_i \overset{0}{\not\sim} s_j \vee \exists e \in E : \delta(s_i,e) \overset{2}{\not\sim} \delta(s_j,e) \Longrightarrow s_i \overset{3}{\not\sim} s_j$

(s_i,s_j)	$(\delta(s_i,a),\delta(s_j,a))$	$(\delta(s_i,b),\delta(s_j,b))$	Markierung	wegen
(s_1,s_3)	(s_4,s_5)	(s_5,s_4)		
(s_4,s_5)	(s_3,s_3)	(s_5,s_5)		

Da im 4. Schritt keine weitere Markierung entsteht, ist das Verfahren abgeschlossen.
Äquivalent sind die Zustandspaare (s_1,s_3) und (s_4,s_5). □

Beispiel 4.39 *Äquivalente Zustände*
Gegeben sei der endliche Automat EA$= (S,E,\delta,s_0,F)$, der durch $S = \{s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7\}$, $E = \{0,1\}$, $F = \{s_0\}$ und die folgende Übergangsfunktion δ beschrieben ist.



δ	0	1
s_0	s_0	s_1
s_1	s_3	s_7
s_2	s_6	s_0
s_3	s_1	s_4

δ	0	1
s_4	s_5	s_0
s_5	s_2	s_0
s_6	s_2	s_0
s_7	s_5	s_0

Gesucht sind äquivalente Zustände in diesem Automaten.

1. Schritt

$$s_i \overset{0}{\sim} s_j \Longrightarrow s_i \in F \wedge s_j \in F \vee s_i \notin F \wedge s_j \notin F$$

$$\begin{array}{ll}
(s_0,s_1) & s_0 \in F,\ s_1 \notin F,\ \Rightarrow \text{ Markierung von } (s_0,s_1) \text{ mit } X_0 \\
(s_0,s_2) & s_0 \in F,\ s_2 \notin F,\ \Rightarrow \text{ Markierung von } (s_0,s_2) \text{ mit } X_0 \\
(s_0,s_3) & s_0 \in F,\ s_3 \notin F,\ \Rightarrow \text{ Markierung von } (s_0,s_3) \text{ mit } X_0 \\
(s_0,s_4) & s_0 \in F,\ s_4 \notin F,\ \Rightarrow \text{ Markierung von } (s_0,s_4) \text{ mit } X_0 \\
(s_0,s_5) & s_0 \in F,\ s_5 \notin F,\ \Rightarrow \text{ Markierung von } (s_0,s_5) \text{ mit } X_0 \\
(s_0,s_6) & s_0 \in F,\ s_6 \notin F,\ \Rightarrow \text{ Markierung von } (s_0,s_6) \text{ mit } X_0 \\
(s_0,s_7) & s_0 \in F,\ s_7 \notin F,\ \Rightarrow \text{ Markierung von } (s_0,s_7) \text{ mit } X_0
\end{array}$$

Alle anderen Paare erhalten keine Markierung mit X_0.

s_1	X_0						
s_2	X_0						
s_3	X_0						
s_4	X_0						
s_5	X_0						
s_6	X_0						
s_7	X_0						
	s_0	s_1	s_2	s_3	s_4	s_5	s_6

s_1	X_0						
s_2	X_0	X_1					
s_3	X_0		X_1				
s_4	X_0	X_1		X_1			
s_5	X_0	X_1		X_1			
s_6	X_0	X_1		X_1			
s_7	X_0	X_1		X_1			
	s_0	s_1	s_2	s_3	s_4	s_5	s_6

2. Schritt

$$s_i \overset{0}{\not\sim} s_j \vee \exists e \in E : \delta(s_i,e) \overset{0}{\not\sim} \delta(s_j,e) \Longrightarrow s_i \overset{1}{\not\sim} s_j$$

(s_i,s_j)	$(\delta(s_i,0),\delta(s_j,0))$	$(\delta(s_i,1),\delta(s_j,1))$	Markierung	wegen
(s_1,s_2)	(s_3,s_6)	(s_7,s_0)	X_1	(s_7,s_0)
(s_1,s_3)	(s_3,s_1)	(s_7,s_4)		
(s_1,s_4)	(s_3,s_5)	(s_7,s_0)	X_1	(s_7,s_0)
(s_1,s_5)	(s_3,s_2)	(s_7,s_0)	X_1	(s_7,s_0)
(s_1,s_6)	(s_3,s_2)	(s_7,s_0)	X_1	(s_7,s_0)
(s_1,s_7)	(s_3,s_5)	(s_7,s_0)	X_1	(s_7,s_0)
(s_2,s_3)	(s_6,s_1)	(s_0,s_4)	X_1	(s_0,s_4)
(s_2,s_4)	(s_6,s_5)	(s_0,s_0)		
(s_2,s_5)	(s_6,s_2)	(s_0,s_0)		
(s_2,s_6)	(s_6,s_2)	(s_0,s_0)		
(s_2,s_7)	(s_6,s_5)	(s_0,s_0)		
(s_3,s_4)	(s_1,s_5)	(s_4,s_0)	X_1	(s_4,s_0)
(s_3,s_5)	(s_1,s_2)	(s_4,s_0)	X_1	(s_4,s_0)
(s_3,s_6)	(s_1,s_2)	(s_4,s_0)	X_1	(s_4,s_0)
(s_3,s_7)	(s_1,s_5)	(s_4,s_0)	X_1	(s_4,s_0)
(s_4,s_5)	(s_5,s_2)	(s_0,s_0)		
(s_4,s_6)	(s_5,s_2)	(s_0,s_0)		
(s_4,s_7)	(s_5,s_5)	(s_0,s_0)		
(s_5,s_6)	(s_2,s_2)	(s_0,s_0)		
(s_5,s_7)	(s_2,s_5)	(s_0,s_0)		
(s_6,s_7)	(s_2,s_5)	(s_0,s_0)		

3. Schritt

$$s_i \overset{0}{\not\sim} s_j \vee \exists e \in E : \delta(s_i,e) \overset{1}{\not\sim} \delta(s_j,e) \Longrightarrow s_i \overset{2}{\not\sim} s_j$$

(s_i,s_j)	$(\delta(s_i,0),\delta(s_j,0))$	$(\delta(s_i,1),\delta(s_j,1))$	Markierung	wegen
(s_1,s_3)	(s_3,s_1)	(s_7,s_4)		

(s_i,s_j)	$(\delta(s_i,0),\delta(s_j,0))$	$(\delta(s_i,1),\delta(s_j,1))$	Markierung	wegen
(s_2,s_4)	(s_6,s_5)	(s_0,s_0)		
(s_2,s_5)	(s_6,s_2)	(s_0,s_0)		
(s_2,s_6)	(s_6,s_2)	(s_0,s_0)		
(s_2,s_7)	(s_6,s_5)	(s_0,s_0)		
(s_4,s_5)	(s_5,s_2)	(s_0,s_0)		
(s_4,s_6)	(s_5,s_2)	(s_0,s_0)		
(s_4,s_7)	(s_5,s_5)	(s_0,s_0)		
(s_5,s_6)	(s_2,s_2)	(s_0,s_0)		
(s_5,s_7)	(s_2,s_5)	(s_0,s_0)		
(s_6,s_7)	(s_2,s_5)	(s_0,s_0)		

Da im 3. Schritt keine weitere Markierung entsteht, ist das Verfahren abgeschlossen.
Äquivalent sind die Zustandspaare

$$\begin{array}{lllll} (s_1,s_3) & (s_2,s_4) & (s_4,s_5) & (s_5,s_6) & (s_6,s_7) \\ & (s_2,s_5) & (s_4,s_6) & (s_5,s_7) & \\ & (s_2,s_6) & (s_4,s_7) & & \\ & (s_2,s_7) & & & \end{array}$$

□

Wir haben ein Verfahren beschrieben, mit dem wir die äquivalenten Zustandspaare in einem endlichen Automaten EA_1 bestimmen können. Damit sind wir in der Lage, den zu EA_1 reduzierten Automaten EA_2 anzugeben. Aufgrund der äquivalenten Zustandspaare können wir für den Automaten EA_1 Klassen äquivalenter Zustände angeben. Aus diesen Äquivalenzklassen ergibt sich dann die Zustandsmenge S_2 des reduzierten Automaten EA_2:

$$\begin{array}{rcl} S_2 & = & \{[s] \mid \{s_j \in S_1 \wedge s_j \sim s_i \in S_1\} \\ s_{02} & = & [s_0] \\ F_2 & = & \{[s] \mid \{s_j \in F_1 \wedge s_j \sim s_i \in F_1\} \end{array}$$

Beispiel 4.40 *Reduzierter Automat*
Wir betrachten den EA$=(S,E,\delta,s_0,F)$ aus Beispiel 4.37 mit $S=\{s_0,s_1,s_2\}$, $E=\{a,b\}$, $F=\{s_2\}$ und der Übergangsfunktion δ.
Zu diesem Automaten soll die reduzierte Zustandsmenge bestimmt werden.

Das einzige äquivalente Zustandspaar dieses Automaten ist (s_0,s_1). Damit folgt:

$$\begin{array}{rcl} S_2 & = & \{[s_0],[s_2]\} = \{\{s_0,s_1\},\{s_2\}\} \equiv \{s_0,s_2\} \\ s_{02} & = & [s_0] = \{s_0\} \\ F_2 & = & \{[s_2]\} = \{s_2\} \end{array}$$

Den der Äquivalenzklasse $\{s_0,s_1\}$ entsprechenden Zustand haben wir der Einfachheit halber mit s_0 bezeichnet. □

Beispiel 4.41 *Reduzierter Automat*
Wir betrachten den EA$=(S,E,\delta,s_0,F)$ aus Beispiel 4.38 mit $S=\{s_0,s_1,s_2,s_3,s_4,s_5\}$, $E=\{a,b\}$, $F=\{s_4,s_5\}$ und der Übergangsfunktion δ.

Die äquivalenten Zustandspaare dieses Automaten sind (s_1,s_3) und (s_4,s_5). Damit folgt:

$$\begin{array}{rcl} S_2 & = & \{[s_0],[s_1],[s_2],[s_4]\} = \{\{s_0\},\{s_1,s_3\},\{s_2\},\{s_4,s_5\},\} \equiv \{s_0,s_1,s_2,s_4\} \\ s_{02} & = & [s_0] = \{s_0\} \\ F_2 & = & \{[s_4]\} = \{s_4\} \end{array}$$

Den der Äquivalenzklasse $[s_1]$ entsprechenden Zustand haben wir mit s_1 und den $[s_4]$ entsprechenden Zustand mit s_4 bezeichnet. □

Beispiel 4.42 *Reduzierter Automat*
Wir betrachten den EA$= (S,E,\delta,s_0,F)$ aus Beispiel 4.39 mit $S = \{s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7\}$, $E = \{0,1\}$, $F = \{s_0\}$ und der Übergangsfunktion δ.

Die äquivalenten Zustandspaare dieses Automaten sind

$$\begin{array}{lllll} (s_1,s_3) & (s_2,s_4) & (s_4,s_5) & (s_5,s_6) & (s_6,s_7) \\ & (s_2,s_5) & (s_4,s_6) & (s_5,s_7) & \\ & (s_2,s_6) & (s_4,s_7) & & \\ & (s_2,s_7) & & & \end{array}$$

Damit folgt:

$$\begin{aligned} S_2 &= \{[s_0],[s_1],[s_2]\} = \{\{s_0\},\{s_1,s_3\},\{s_2,s_4,s_5,s_6,s_7\},\} \equiv \{s_0,s_1,s_2\} \\ s_{02} &= [s_0] = \{s_0\} \\ F_2 &= \{[s_0]\} = \{s_0\} \end{aligned}$$

Es fällt sofort auf, dass der Zustand s_2 äquivalent ist zu den Zuständen s_4, s_5, s_6 und s_7. Wenn dies der Fall ist, dann müssen natürlich auch die Zustände s_4, s_5, s_6 und s_7 untereinander äquivalent sein wie das Beispiel bestätigt. Dementsprechend befinden sich diese fünf Zustände in einer Äquivalenzklasse $[s_2]$. □

Um den reduzierten Automaten im Zustandsdiagramm darszustellen, müssen wir schließlich noch dessen Übergangsfunktion festlegen. Die aus den eliminierten Zuständen herausführenden Transitionen fallen mit den entsprechenden Zuständen einfach weg, da diese von den entsprechenden äquivalenten Zuständen ausgehen. Die auf den eliminierten Zuständen endenden Transitionen fallen jedoch nicht weg, sie müssen nach der Reduzierung auf den entsprechenden äquivalenten Zuständen enden. Wir wollen die Übergangsfunktion $\delta_2 : S_2 \times E \longrightarrow S_2$ des reduzierten Automaten in einer Definition festlegen:

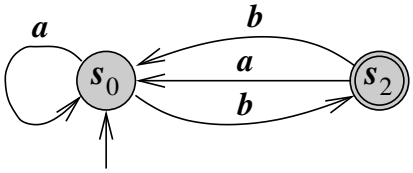
Definition 4.13 *Übergangsfunktion eines reduzierten Automaten*

$$\delta_2 : S_2 \times E \longrightarrow S_2 \quad \textit{mit} \quad \delta_2([s],e) = [\delta(s,e)]$$

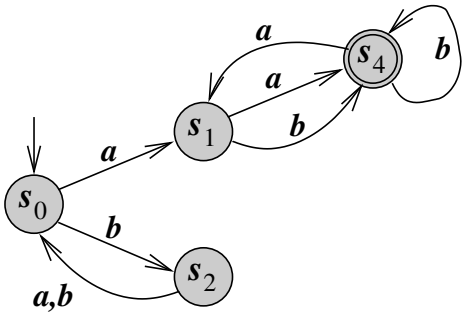
Die Übergangsfunktion δ_2 angewendet auf die in einer Äquivalenzklasse zusammengefassten Zustände ergibt die Äquivalenzklasse der Übergangsfunktion δ angewendet auf einen Zustand dieser Äquivalenzklasse.

Beispiel 4.43 *Übergangsfunktion eines reduzierten Automaten*
Man gebe die Übergangsfunktionen der reduzierten Automaten aus den drei Beispielen 4.37, 4.38 und 4.39 an.

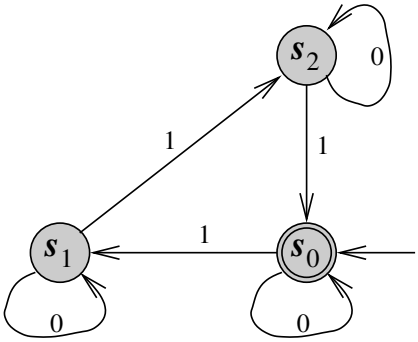
Bei dem Automaten aus Beispiel 4.37 wird der Zustand s_1 eliminiert. Die vom Zustand s_1 ausgehende Transition mit a nach Zustand s_0 fällt weg, deren Verhalten wird durch die Schlinge am Zustand s_0 beschrieben. Die vom Zustand s_1 ausgehende Transition mit b nach Zustand s_2 fällt weg, deren Verhalten wird durch die von Zustand s_0 nach Zustand s_2 verlaufende Transition mit b beschrieben. Die vom Zustand s_2 ausgehende und in Zustand s_1 endende Transition mit b endet im reduzierten Automaten in dem zum Zustand s_1 äquivalenten Zustand s_0.



Bei dem Automaten aus Beispiel 4.38 werden die Zustände s_3 und s_5 eliminiert. Die vom Zustand s_3 ausgehenden Transitionen mit a nach Zustand s_5 und mit b nach Zustand s_4 fallen weg. Ebenso fällt die vom Zustand s_5 ausgehende Transition mit a nach Zustand s_3 und sowie die Schlinge am Zustand s_5 weg. Die vom Zustand s_4 ausgehende und in Zustand s_3 endende Transition mit a endet im reduzierten Automaten in dem zum Zustand s_3 äquivalenten Zustand s_1. Die vom Zustand s_1 ausgehende und in Zustand s_5 endende Transition mit b endet im reduzierten Automaten in dem zum Zustand s_5 äquivalenten Zustand s_4. Die vom Zustand s_4 ausgehende und in Zustand s_5 endende Transition mit b wird zu einer Schlinge am Zustand s_4.



Bei dem Automaten aus Beispiel 4.39 werden die Zustände s_3, s_4, s_5, s_6, und s_7 eliminiert. Die vom Zustand s_1 ausgehende und in Zustand s_3 endende Transition mit 0 wird zu einer Schlinge am Zustand s_1. Die vom Zustand s_2 ausgehende und in Zustand s_6 endende Transition mit 0 wird zu einer Schlinge am Zustand s_2. Die vom Zustand s_1 ausgehende und in Zustand s_7 endende Transition mit 1 endet im reduzierten Automaten in dem zum Zustand s_7 äquivalenten Zustand s_2.



□

4.5.5 Endlicher Automat mit Ausgabe

In diesem Abschnitt wollen wir unsere bisherigen Betrachtungen zu EAs um eine Ausgabe erweitern. Einen endlichen Automaten mit Ausgabe bezeichnet man auch als endliche Maschine. Damit haben wir alle Merkmale, die der im ersten Abschnitt behandelte Getränkeautomat aufweist, auch formal eingeführt. In unserer anschaulischen Vorstellung von einem endlichen Automaten stellt die endliche Maschine auch eine Kontrolleinheit dar, die sich in einem bestimmten Zustand s_i befindet, ein Symbol vom Eingabeband liest und in Abhängigkeit vom gelesenen Symbol e_i und dem momentanen Zustand s_i in einen Folgezustand s_j übergeht. Daneben gibt es bei der endliche Maschine noch einen Schreibkopf, der mit jeder Eingabe e_i ein Zeichen z_j auf ein Ausgabeband schreibt.

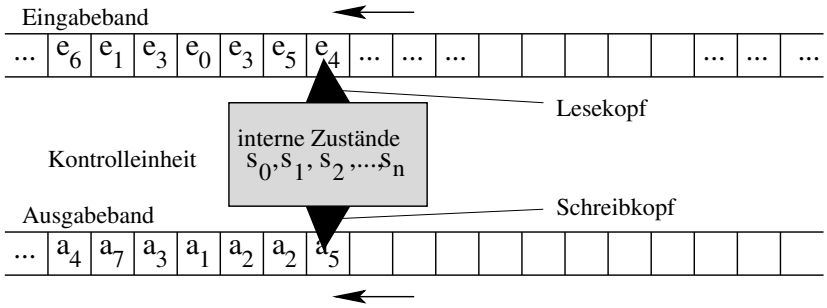


Abbildung 4.10: Endlicher Automat mit Ausgabe

Analog zum endlichen Automaten (EA) definieren wir die endliche Maschine (EM) als ein Sechstupel:

Definition 4.14 *Endlicher Automat mit Ausgabe*
Ein endlicher Automat mit Ausgabe, kurz endliche Maschine (EM) genannt, ist durch ein Sechstupel

$$EM = (S, E, Z, \delta, \gamma, s_0)$$

definiert, wobei

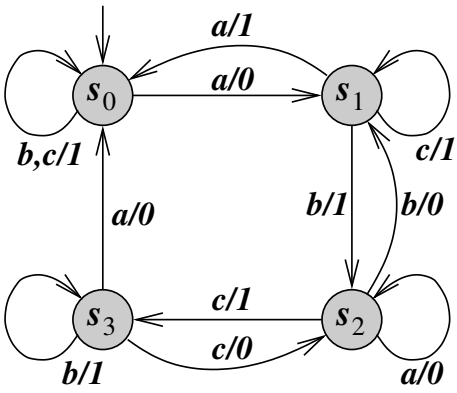
$$\begin{array}{rcll} S & = & \{s_0, s_1, \ldots, s_n\} & \textit{eine nichtleere Menge, die Zustandsmenge}, \\ E & = & \{e_0, e_1, \ldots, e_r\} & \textit{eine nichtleere Menge, das Eingangsalphabet}, \\ Z & = & \{z_0, z_1, \ldots, z_m\} & \textit{eine nichtleere Menge, das Ausgabealphabet}, \\ \delta & : & S \times E \longrightarrow S & \textit{die Übergangsfunktion}, \\ \gamma & : & S \times E \longrightarrow Z & \textit{die Ausgabefunktion und} \\ s_0 & \in & S & \textit{der Anfangszustand ist}. \end{array}$$

Die EM weist gegenüber einem EA folgende Unterschiede auf:

- Es gibt ein Ausgabealphabet und eine Ausgabefunktion.
- Nach der Eingabe eines Symbols wird neben der Übergangsfunktion δ zur Bestimmung des Folgezustandes die Ausgabefunktion γ angewendet um das Ausgabezeichen zu bestimmen.
- Es gibt bei endlichen Maschinen EM keine Endzustände. Die Ausgabe ersetzt die Endzustände.

Bei einer EA steht die Klassifizierung der Eingabewörter in nicht akzeptierte und akzeptierte Wörter im Vordergrund. Bei einer EM hingegen wird zu jedem verarbeiteten Eingabewort aus $e \in E^*$ ein Ausgabewort $z \in Z^*$ erzeugt. Das Ergebnis der Verarbeitung der Eingaben wird also nicht durch das Erreichen eines Endzustandes generiert, sondern durch explizite Ausgaben. Beispielsweise wurde beim Getränkeautomaten durch die Eingabe von Geldstücken die Ausgabe eines bestimmten Getränks und einer bestimmten Menge Wechselgeld erzeugt.

Beispiel 4.44 *Beschreibung einer EM*
Gegeben sei die endliche Automat EM$= (S, E, A, \delta, \gamma, s_0)$ durch $S = \{s_0, s_1, s_2, s_3\}$, $E = \{a, b, c\}$, $A = \{0, 1\}$, sowie der Übergangsfunktion δ und die Ausgabefunktion γ, die durch das folgende Zustandsdiagramm bzw. die Tabelle beschrieben ist.



Übergangsfunktion δ und Ausgabefunktion γ in Tabellenform

δ/γ	a	b	c
s_0	$s_1/0$	$s_0/1$	$s_0/1$
s_1	$s_0/1$	$s_2/1$	$s_1/1$
s_2	$s_2/0$	$s_1/0$	$s_3/1$
s_3	$s_0/0$	$s_3/1$	$s_2/0$

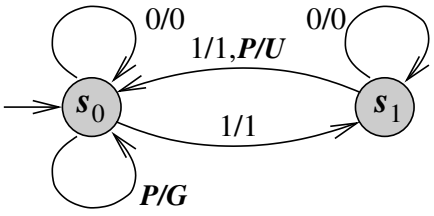
□

Die Überführungs- und die Ausgabefunktion wird bei einer EM wie auch bei einem EA durch ein Zustandsdiagramm oder durch eine Zustandstabelle beschrieben. Dabei wird jeder Eintrag um die Ausgabe erweitert. Im Zustandsdiagramm wird an jeder Kante durch einen Schrägstrich von der Eingabe getrennt die Ausgabe angegeben. In der Zustandstabelle wird die Ausgabe durch einen Schrägstrich getrennt vom Folgezustand angegeben.

Beispiel 4.45 *Entwurf einer endlichen Maschine*
Angenommen ein PC erzeuge serielle Bitfolgen aus 0 und 1, die über eine Leitung übertragen werden sollen. Ein Wort bestehe aus beliebig vielen 0 und 1 und werde durch das Zeichen P abgeschlossen. Die Übertragung der Worte soll durch ein Paritätsbit, das angibt ob die Anzahl 1 eines Wortes gerade oder ungerade ist, gesichert werden. Gesucht ist eine EM, die das Paritätsbit berechnet und das Zeichen P in der Eingabefolge durch ein entsprechendes Paritätszeichen G für gerade oder U für ungerade in der Ausgabefolge ersetzt.

Es ist klar, dass die EM das Eingabealphabet $E = \{0,1,P\}$ und das Ausgabealphabet $Z = \{0,1,G,U\}$ haben muss. Damit entwerfen wir ausgehend vom Ausgangszustand s_0 das Zustandsdiagramm dieser EM. Dem Ausgangszustand s_0 können wir natürlich nur die Bedeutung einer geraden Parität zuweisen, denn im Zustand s_0 ist möglicherweise noch kein Bit und damit keine 1 eingegeben worden.

Angenommen wir befinden uns im Zustand s_0 und das erste Bit des ersten Wortes wird gelesen. Mit einer gelesenen 0 verbleiben wir im Zustand s_0. Da wir aber auch eine 0 ausgeben müssen, lösen wir dies durch eine Schlinge mit der Beschriftung 0/0 am Zustand s_0. Mit einer gelesenen 1 verändert sich die Parität von gerade nach ungerade, damit kommen wir in einen neuen Zustand s_1. Da wir auch eine 1 ausgeben müssen, folgt hieraus eine Transition mit der Beschriftung 1/1 von Zustand s_0 nach Zustand s_1. Ein im Zustand s_0 gelesenes Zeichen P zeigt an, das ein Wort zu Ende ist. Weiter bedeutet dies, dass das bisherige Wort kein Bit enthalten (leeres Wort) hat oder dass es nur aus 0 bestanden hat. Es kann aber auch bedeuten, dass das Wort aus 0 und 1 bestand, wodurch der Zustand s_0 zwischenzeitlich verlassen wurde, aber die EM aufgrund einer ungeraden Anzahl 1 wieder in den Zustand s_0 zurückgekehrt ist. Aus diesen Möglichkeiten folgt, dass wir mit einem gelesenen Zeichen P im Zustand s_0 das Zeichen G für gerade Parität ausgeben und im Zustand s_0 verbleiben müssen, also eine Schlinge mit der Beschriftung P/G am Zustand s_0. Damit ist der Zustand s_0 vollständig festgelegt.



Übergangsfunktion δ und Ausgabefunktion γ in Tabellenform

δ	0	1	P
s_0	$s_0/0$	$s_1/1$	s_0/G
s_1	$s_1/0$	$s_0/1$	s_1/U

Wir betrachten den Zustand s_1, dem wir die Bedeutung der ungeraden Parität zuweisen. Mit einer gelesenen 0 verändert sich die Parität nicht, wir verbleiben im Zustand s_1 und geben eine 0 aus. Wir lösen dies durch eine Schlinge mit der Beschriftung 0/0 am Zustand s_1. Mit einer gelesenen 1 verän-

dert sich die Parität von ungerade nach gerade, wir kommen in den Zustand s_0. Da wir eine 1 ausgeben müssem, ergibt dies eine Transition mit der Beschriftung 1/1 von Zustand s_1 nach Zustand s_0. Mit einem gelesenen P ist das Wort zu Ende. Da das nächste Wort mit einer geraden Parität beginnen soll, müssen wir mit der Ausgabe der ungeraden Parität in den Zustand s_0 zurück. Da außer s_0 und s_1 keine weiteren Zustände benötigt werden, ist das Zustandsdiagramm fertig.

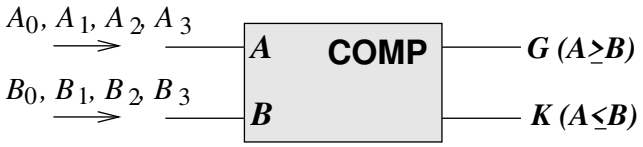
Offensichtlich hätten wir dieses Problem der fortlaufenden Paritätsberechnung mit einem EA nicht lösen können. Durch die Überführung in einen Endzustand können wir die Parität eines einzelnen Wortes zwar bestimmen, danach befindet sich der EA jedoch in einem Endzustand. Wird dieser verlassen, dann geht auch die dadurch bestimmte Information verloren. □

Reduzierung des Zustandsraumes einer endlichen Maschine

Wir wollen im Folgenden die Zustandsreduzierung von EA auf die EM erweitern. Dazu wollen wir jedoch nur den Algorithmus anpassen ohne die vollständige Herleitung wieder ausführlich zu betrachten. Der Zustandsraum einer EM unterscheidet sich von dem eines EA dadurch, dass es bei der EM einerseits mit der Ausgabe spezifierte Transitionen und andererseits keine Endzustände gibt. Bzgl. dieser beiden Punkte ist eine Anpassung des Algorithmus erforderlich.

Zwei EMs bzw. zwei Zustände einer EM sind nur dann äquivalent, wenn die Zustände neben dem äquivalenten Übergangsverhalten auch ein identisches Ausgabeverhalten aufweisen. Genauer gesagt, zwei Zustände einer EM sind dann äquivalent, wenn die aus den beiden Zuständen herausführenden Transitionen neben einem äquivalenten Folgezustand auch die gleichen Ausgaben aufweisen. Da es bei einer EM keine Endzustände gibt, die verglichen werden müssen, können wir im ersten Schritt einfach anstelle der Endzutände die Ausgaben überprüfen. Zwei Zustände können nicht äquivalent sein, wenn die Transitionen dieser Zustände unterschiedliche Ausgaben aufweisen. In diesem Fall erhält ein Zustandspaar im ersten Schritt eine Markierung X_0. Von da an müssen die Ausgaben auch nicht mehr weiter beachtet werden und das Verfahren läuft exakt genauso ab wie beim EA. Wir wollen ein weiteres Beispiel einer EM betrachten und die Zustandsrreduzierung an diesem Beispiel verdeutlichen.

Beispiel 4.46 *Entwurf und Reduzierung einer endlichen Maschine*
Zum Größenvergleich zweier vierstelliger Dualzahlen A (A_3,A_2,A_1,A_0) und B (B_3,B_2,B_1,B_0) werde ein Komparator benötigt. Die beiden Dualzahlen A und B werden bitseriell einlaufend über zwei Leitungen empfangen, so dass zur Realisierung des Komparators eine speicherbehaftete Anordnung in Form einer EM benötigt wird.

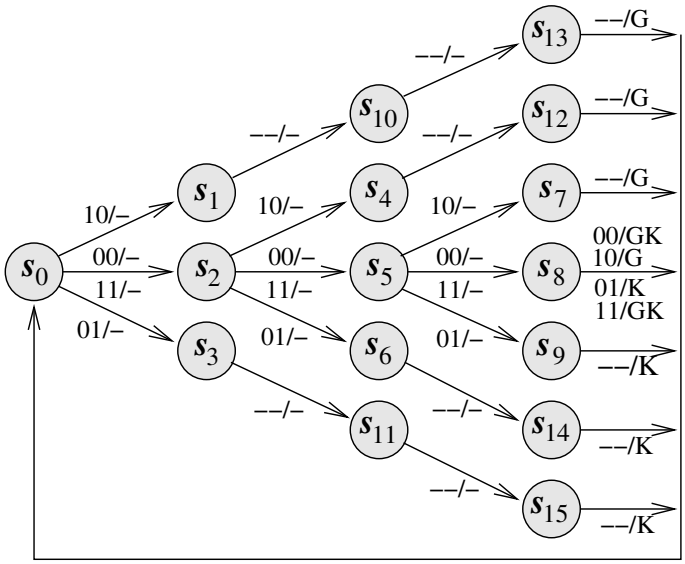


Die Ausgabe G des Komparators zeigt an, wenn $A \geq B$ ist, und die Ausgabe K zeigt an, wenn $A \leq B$ gilt. Zu entwerfen ist eine EM, welche die seriell einlaufenden Bitstellen von A und B schritthaltend vergleicht und das Ergebnis nach dem Einlaufen der jeweils letzten Bitstelle A_0, B_0 (das niederwertigste Bit läuft zuletzt ein) eines Wortes ausgibt.

Da die jeweils entsprechenden Bitsstellen (A_3 mit B_3, A_2 mit B_2, A_1 mit B_1 und A_0 mit B_0) der beiden Dualzahlen A und B von der EM parallel gelesen werden, und jede einzelne Bitstelle entweder 0 oder 1 sein kann, ergeben sich folgende Kombinationsmöglichkeiten: $A_i = 0$ mit $B_i = 0$, $A_i = 0$ mit $B_i = 1$, $A_i = 1$ mit $B_i = 0$ und $A_i = 1$ mit $B_i = 1$. Das Eingabealphabet der EM lautet somit: $E = \{00, 01, 10, 11\}$. Ausgegeben wird die Information $A \geq B$ oder $A \leq B$, das Ausgabealphabet lautet $Z = \{G, K\}$. Damit können wir ausgehend vom Ausgangszustand s_0 das Zustandsdiagramm dieser EM entwerfen.

Im Ausgangszustand s_0 werden die ersten beiden (höchstwertigsten) Bitstellen A_3 und B_3 gelesen. Da

das Eingabealphabet vier Zeichen aufweist, gibt es vier Transition, um den Zustand s_0 zu verlassen. Mit der Eingabe 10 kommen wir in einen Folgezustand s_1, in dem $A > B$ gilt, das das höchstwertigste Bit bereits eine Entscheidung ergeben hat. Entsprechend kommen wir mit der Eingabe 01 in einen Zustand s_3, in dem $A < B$ gilt. Mit der Eingabe 00 kommen wir in einen Zustand s_2, in dem weiterhin $A = B$ gilt. Ebenso kommen wir mit der Eingabe 11 in einen Zustand, in dem weiterhin $A = B$ gilt. Da es für die ausstehende Entscheidung $A \geq B$ oder $A \leq B$ nicht von Bedeutung ist, ob hier 00 oder 11 eingelaufen ist, können wir hier auch in den Zustand s_2 gehen. Dies wird das Diagramm nicht unötig vergrößern.



Als nächstes müssen wir die drei Folgezustände s_1, s_2 und s_3 betrachten, in denen die nächsten beiden Bitstellen A_2 und B_2 gelesen werden. Zu jedem dieser drei Zustände kann es vier Folgezustände geben. Wir wollen das Diagramm jedoch direkt vereinfachen. Der Zustand s_2 ist am einfachsten zu behandeln, dieser gestaltet sich analog zum Zustand s_0. Mit 10 kommen wir in einen Zustand s_4, mit 00 oder 11 kommen wir in einen Zustand s_5 und mit 01 kommen wir in einen Zustand s_6. Im Zustand s_1 ist die Entscheidung bereits gefallen, dennoch müssen natürlich die nächsten beiden Bitstellen gelesen werden. Wir müssen aber von Zustand s_1 aus nicht mehr verzweigen, da in allen Folgezuständen bis zum Einlaufen des nächsten Wortes $A > B$ gilt. Wir wählen also einen Folgezustand s_{10}, in den wir mit allen vier Eingabezeichen kommen. Gleiches gilt für den Zustand s_3, hier wählen wir den Folgezustand s_{11}.

δ	00	01	10	11
s_0	$s_2/-$	$s_3/-$	$s_1/-$	$s_2/-$
s_1	$s_{10}/-$	$s_{10}/-$	$s_{10}/-$	$s_{10}/-$
s_2	$s_5/-$	$s_6/-$	$s_4/-$	$s_5/-$
s_3	$s_{11}/-$	$s_{11}/-$	$s_{11}/-$	$s_{11}/-$
s_4	$s_{12}/-$	$s_{12}/-$	$s_{12}/-$	$s_{12}/-$
s_5	$s_8/-$	$s_9/-$	$s_7/-$	$s_8/-$
s_6	$s_{14}/-$	$s_{14}/-$	$s_{14}/-$	$s_{14}/-$
s_7	s_0/G	s_0/G	s_0/G	s_0/G
s_8	s_0/GK	s_0/K	s_0/G	s_0/GK
s_9	s_0/K	s_0/K	s_0/K	s_0/K
s_{10}	$s_{13}/-$	$s_{13}/-$	$s_{13}/-$	$s_{13}/-$
s_{11}	$s_{15}/-$	$s_{15}/-$	$s_{15}/-$	$s_{15}/-$
s_{12}	s_0/G	s_0/G	s_0/G	s_0/G
s_{13}	s_0/G	s_0/G	s_0/G	s_0/G
s_{14}	s_0/K	s_0/K	s_0/K	s_0/K
s_{15}	s_0/K	s_0/K	s_0/K	s_0/K

Würden wir für jeden Zustand jeweils die vier Folgezustände darstellen, dann würde sich ein sog. vollständiger Entscheidungsbaum ergeben. Da die Eingabeworte aus je vier Zeichen bestehen, besteht jede Transitionssequenz aus genau vier Transitionen. Mit der jeweils vierten Transition muss der EA aber unbedingt in den Ausgangszustand s_0 zurückkehren, um dort die ersten beiden Stellen des nächsten Wortes zu lesen. Ohne diese Schleife würden wir den Zustansdraum unendlich ausdehnen, wobei sich jeweils immer die gleichen Zustände wiederholen würden. Andererseits darf aber auch vor Ablauf von genau vier Transitionen der Zustand s_0 *nicht* erreicht werden, um die Synchronität mit den einlaufenden Worten sicherzustellen.

Damit gibt es noch drei weitere Folgezustände s_7, s_8 und s_9 zum Zustand s_5. Zum Zustand s_4 gibt es noch einen Folgezustand s_{12}, zum Zustand s_6 einen Folgezustand s_{14}, zum Zustand s_{10} einen Folgezustand s_{13} und zum Zustand s_{11} einen Folgezustand s_{15}, so dass schließlich das dargestellte Zustandsdiagramm entsteht. Gleichzeitig müssen wir mit den jeweils vierten Transitionen die berechnete Information $A \geq B$ oder $A \leq B$ ausgeben, die im Zustandsdiagramm durch G für größer und K für kleiner abgekürzt ist. Die Ausgabe erfolgt nach der jeweils vierten Transition wenn der Automat in den Ausgangszustand s_0 zurückgekehrt ist. Bei allen vorhergehenden Transitionen steht die Ausgabe noch nicht fest, sie kann aufgrund der durchlaufenden Zustände erst berechnet werden. Im Zustandsdiagramm haben wir die fehlenden Ausgaben durch "-" gekennzeichnet. □

Beispiel 4.47 *Äquivalente Zustände einer EM*
Gegeben sei die endliche Maschine aus Beispiel 4.46. Gesucht sind äquivalente Zustände und die reduzierte EM.

Im 1. Schritt bekommen alle Zustandspaare (s_i, s_j), bei denen sich die Ausgaben unterscheiden, eine Markierung mit X_0

(s_i,s_j)	$\delta(s_i,00),\delta(s_j,00)$	$\delta(s_i,01),\delta(s_j,01)$	$\delta(s_i,10),\delta(s_j,10)$	$\delta(s_i,11),\delta(s_j,11)$	
(s_0,s_1)	$(s_2/-,s_{10}/-)$	$(s_3/-,s_{10}/-)$	$(s_1/-,s_{10}/-)$	$(s_2/-,s_{10}/-)$	
(s_0,s_2)	$(s_2/-,s_5/-)$	$(s_3/-,s_6/-)$	$(s_1/-,s_4/-)$	$(s_2/-,s_5/-)$	
(s_0,s_3)	$(s_2/-,s_{11}/-)$	$(s_3/-,s_{11}/-)$	$(s_1/-,s_{11}/-)$	$(s_2/-,s_{11}/-)$	
(s_0,s_4)	$(s_2/-,s_{12}/-)$	$(s_3/-,s_{12}/-)$	$(s_1/-,s_{12}/-)$	$(s_2/-,s_{12}/-)$	
(s_0,s_5)	$(s_2/-,s_8/-)$	$(s_3/-,s_9/-)$	$(s_1/-,s_7/-)$	$(s_2/-,s_8/-)$	
(s_0,s_6)	$(s_2/-,s_{14}/-)$	$(s_3/-,s_{14}/-)$	$(s_1/-,s_{14}/-)$	$(s_2/-,s_{14}/-)$	
(s_0,s_7)	$(s_2/-,s_0/G)$	$(s_3/-,s_0/G)$	$(s_1/-,s_0/G)$	$(s_2/-,s_0/G)$	X_0
(s_0,s_8)	$(s_2/-,s_0/GK)$	$(s_3/-,s_0/K)$	$(s_1/-,s_0/G)$	$(s_2/-,s_0/GK)$	X_0
(s_0,s_9)	$(s_2/-,s_0/K)$	$(s_3/-,s_0/K)$	$(s_1/-,s_0/K)$	$(s_2/-,s_0/K)$	X_0
(s_0,s_{10})	$(s_2/-,s_{13}/-)$	$(s_3/-,s_{13}/-)$	$(s_1/-,s_{13}/-)$	$(s_2/-,s_{13}/-)$	
(s_0,s_{11})	$(s_2/-,s_{15}/-)$	$(s_3/-,s_{15}/-)$	$(s_1/-,s_{15}/-)$	$(s_2/-,s_{15}/-)$	
(s_0,s_{12})	$(s_2/-,s_0/G)$	$(s_3/-,s_0/G)$	$(s_1/-,s_0/G)$	$(s_2/-,s_0/G)$	X_0
(s_0,s_{13})	$(s_2/-,s_0/G)$	$(s_3/-,s_0/G)$	$(s_1/-,s_0/G)$	$(s_2/-,s_0/G)$	X_0
(s_0,s_{14})	$(s_2/-,s_0/K)$	$(s_3/-,s_0/K)$	$(s_1/-,s_0/K)$	$(s_2/-,s_0/K)$	X_0
(s_0,s_{15})	$(s_2/-,s_0/K)$	$(s_3/-,s_0/K)$	$(s_1/-,s_0/K)$	$(s_2/-,s_0/K)$	X_0
(s_1,s_2)	$(s_{10}/-,s_5/-)$	$(s_{10}/-,s_6/-)$	$(s_{10}/-,s_4/-)$	$(s_{10}/-,s_5/-)$	
(s_1,s_3)	$(s_{10}/-,s_{11}/-)$	$(s_{10}/-,s_{11}/-)$	$(s_{10}/-,s_{11}/-)$	$(s_{10}/-,s_{11}/-)$	
(s_1,s_4)	$(s_{10}/-,s_{12}/-)$	$(s_{10}/-,s_{12}/-)$	$(s_{10}/-,s_{12}/-)$	$(s_{10}/-,s_{12}/-)$	
(s_1,s_5)	$(s_{10}/-,s_8/-)$	$(s_{10}/-,s_9/-)$	$(s_{10}/-,s_7/-)$	$(s_{10}/-,s_8/-)$	
(s_1,s_6)	$(s_{10}/-,s_{14}/-)$	$(s_{10}/-,s_{14}/-)$	$(s_{10}/-,s_{14}/-)$	$(s_{10}/-,s_{14}/-)$	
(s_1,s_7)	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	X_0
(s_1,s_8)	$(s_{10}/-,s_0/GK)$	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/GK)$	X_0
(s_1,s_9)	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	X_0
(s_1,s_{10})	$(s_{10}/-,s_{13}/-)$	$(s_{10}/-,s_{13}/-)$	$(s_{10}/-,s_{13}/-)$	$(s_{10}/-,s_{13}/-)$	
(s_1,s_{11})	$(s_{10}/-,s_{15}/-)$	$(s_{10}/-,s_{15}/-)$	$(s_{10}/-,s_{15}/-)$	$(s_{10}/-,s_{15}/-)$	
(s_1,s_{12})	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	X_0
(s_1,s_{13})	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	$(s_{10}/-,s_0/G)$	X_0
(s_1,s_{14})	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	X_0

(s_i,s_j)	$\delta(s_i,00),\delta(s_j,00)$	$\delta(s_i,01),\delta(s_j,01)$	$\delta(s_i,10),\delta(s_j,10)$	$\delta(s_i,11),\delta(s_j,11)$	
(s_1,s_{15})	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	$(s_{10}/-,s_0/K)$	X_0
(s_2,s_3)	$(s_5/-,s_{11}/-)$	$(s_6/-,s_{11}/-)$	$(s_4/-,s_{11}/-)$	$(s_5/-,s_{11}/-)$	
(s_2,s_4)	$(s_5/-,s_{12}/-)$	$(s_6/-,s_{12}/-)$	$(s_4/-,s_{12}/-)$	$(s_5/-,s_{12}/-)$	
(s_2,s_5)	$(s_5/-,s_8/-)$	$(s_6/-,s_9/-)$	$(s_4/-,s_7/-)$	$(s_5/-,s_8/-)$	
(s_2,s_6)	$(s_5/-,s_{14}/-)$	$(s_6/-,s_{14}/-)$	$(s_4/-,s_{14}/-)$	$(s_5/-,s_{14}/-)$	
...					

Wir wollen die Bestimmung der Markierungen X_0 hier nicht weiter ausbreiten und tragen das Ergebnis direkt in das Schema ein.

s_1	X_2													
s_2	X_2	X_2												
s_3	X_2	X_2	X_2											
s_4	X_1	X_1	X_1	X_1										
s_5	X_1	X_1	X_1	X_1	X_1									
s_6	X_1	X_1	X_1	X_1	X_1	X_1								
s_7	X_0	X_0	X_0	X_0	X_0	X_0	X_0							
s_8	X_0	X_0	X_0	X_0	X_0	X_0	X_0	X_0						
s_9	X_0	X_0	X_0	X_0	X_0	X_0	X_0	X_0	X_0					
s_{10}	X_1	X_1	X_1	X_1		X_1	X_1	X_0	X_0	X_0				
s_{11}	X_1	X_1	X_1	X_1	X_1	X_1		X_0	X_0	X_0	X_1			
s_{12}	X_0	X_0	X_0	X_0	X_0	X_0	X_0		X_0	X_0	X_0	X_0		
s_{13}	X_0	X_0	X_0	X_0	X_0	X_0	X_0		X_0	X_0	X_0	X_0		
s_{14}	X_0	X_0	X_0	X_0	X_0	X_0	X_0	X_0	X_0		X_0	X_0	X_0	X_0
s_{15}	X_0	X_0	X_0	X_0	X_0	X_0	X_0	X_0	X_0		X_0	X_0	X_0	X_0
	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}

Im 2. Schritt bekommen alle Zustandspaare (s_i,s_j) eine Markierung X_1, bei denen das Folgezustandspaar für ein beliebiges $e \in E$ bereits eine Markierung X_0 besitzt. Dabei müssen nur die noch nicht mit X_0 markierten Zustandspaare (s_i,s_j) untersucht werden.

(s_i,s_j)	$\delta(s_i,00),\delta(s_j,00)$	$\delta(s_i,01),\delta(s_j,01)$	$\delta(s_i,10),\delta(s_j,10)$	$\delta(s_i,11),\delta(s_j,11)$	
(s_0,s_1)	$(s_2/-,s_{10}/-)$	$(s_3/-,s_{10}/-)$	$(s_1/-,s_{10}/-)$	$(s_2/-,s_{10}/-)$	
(s_0,s_2)	$(s_2/-,s_5/-)$	$(s_3/-,s_6/-)$	$(s_1/-,s_4/-)$	$(s_2/-,s_5/-)$	
(s_0,s_3)	$(s_2/-,s_{11}/-)$	$(s_3/-,s_{11}/-)$	$(s_1/-,s_{11}/-)$	$(s_2/-,s_{11}/-)$	
(s_0,s_4)	$(s_2/-,s_{12}/-)$	$(s_3/-,s_{12}/-)$	$(s_1/-,s_{12}/-)$	$(s_2/-,s_{12}/-)$	X_1
(s_0,s_5)	$(s_2/-,s_8/-)$	$(s_3/-,s_9/-)$	$(s_1/-,s_7/-)$	$(s_2/-,s_8/-)$	X_1
(s_0,s_6)	$(s_2/-,s_{14}/-)$	$(s_3/-,s_{14}/-)$	$(s_1/-,s_{14}/-)$	$(s_2/-,s_{14}/-)$	X_1
(s_0,s_{10})	$(s_2/-,s_{13}/-)$	$(s_3/-,s_{13}/-)$	$(s_1/-,s_{13}/-)$	$(s_2/-,s_{13}/-)$	X_1
(s_0,s_{11})	$(s_2/-,s_{15}/-)$	$(s_3/-,s_{15}/-)$	$(s_1/-,s_{15}/-)$	$(s_2/-,s_{15}/-)$	X_1
(s_1,s_2)	$(s_{10}/-,s_5/-)$	$(s_{10}/-,s_6/-)$	$(s_{10}/-,s_4/-)$	$(s_{10}/-,s_5/-)$	
(s_1,s_3)	$(s_{10}/-,s_{11}/-)$	$(s_{10}/-,s_{11}/-)$	$(s_{10}/-,s_{11}/-)$	$(s_{10}/-,s_{11}/-)$	
(s_1,s_4)	$(s_{10}/-,s_{12}/-)$	$(s_{10}/-,s_{12}/-)$	$(s_{10}/-,s_{12}/-)$	$(s_{10}/-,s_{12}/-)$	X_1
(s_1,s_5)	$(s_{10}/-,s_8/-)$	$(s_{10}/-,s_9/-)$	$(s_{10}/-,s_7/-)$	$(s_{10}/-,s_8/-)$	X_1
(s_1,s_6)	$(s_{10}/-,s_{14}/-)$	$(s_{10}/-,s_{14}/-)$	$(s_{10}/-,s_{14}/-)$	$(s_{10}/-,s_{14}/-)$	X_1
...					

Wir wollen an dieser Stelle wieder abbrechen und das Ergebnis direkt eintragen. Im 3. Schritt bekommen alle Zustandspaare (s_i, s_j) eine Markierung X_2, bei denen das Folgezustandspaar für ein beliebiges $e \in E$ bereits eine Markierung X_0 oder X_1 besitzt.

(s_i, s_j)	$\delta(s_i,00), \delta(s_j,00)$	$\delta(s_i,01), \delta(s_j,01)$	$\delta(s_i,10), \delta(s_j,10)$	$\delta(s_i,11), \delta(s_j,11)$	
(s_0,s_1)	$(s_2/-, s_{10}/-)$	$(s_3/-, s_{10}/-)$	$(s_1/-, s_{10}/-)$	$(s_2/-, s_{10}/-)$	X_2
(s_0,s_2)	$(s_2/-, s_5/-)$	$(s_3/-, s_6/-)$	$(s_1/-, s_4/-)$	$(s_2/-, s_5/-)$	X_2
(s_0,s_3)	$(s_2/-, s_{11}/-)$	$(s_3/-, s_{11}/-)$	$(s_1/-, s_{11}/-)$	$(s_2/-, s_{11}/-)$	X_2
(s_1,s_2)	$(s_{10}/-, s_5/-)$	$(s_{10}/-, s_6/-)$	$(s_{10}/-, s_4/-)$	$(s_{10}/-, s_5/-)$	X_2
...					

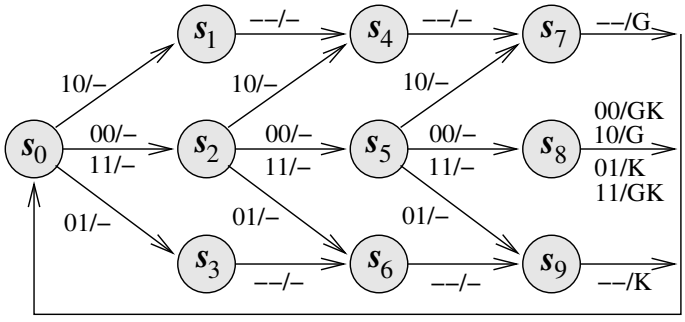
Es kann leicht überprüft werden, dass sich keine weiteren Markierungen mehr ergeben. Die äquivalenten Zustandspaare dieser Maschine sind

$$\begin{array}{cccc} (s_4,s_{10}) & (s_6,s_{11}) & (s_7,s_{12}) & (s_9,s_{14}) \\ & & (s_7,s_{13}) & (s_9,s_{15}) \\ & & (s_{12},s_{13}) & (s_{14},s_{15}) \end{array}$$

Damit folgt für die reduzierte EM die Zustandsmenge:

$$S_{red} = \{ s_0, s_1, s_2, s_3, [s_4], s_5, [s_6], [s_7], s_8, [s_9] \} = \{ s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9 \}$$

und die folgende Übergangsfunktion δ sowie die Ausgabefunktion γ:



□

Aufgabe 4.11 *Komparator als endliche Maschine*
Überlegen Sie, wie sich das Zustandsdiagramm des Komparators ändern muss, wenn die niederwertigste Bitstelle zuerst einläuft? Wieviele zusätzliche Zustände werden benötigt? Zeichnen Sie das Zustandsdiagramm. ◇

4.5.6 Vereinfachtes Minimierungsverfahren

Das betrachtete Verfahren zur Reduzierung der Zustände einer EM kann man auch wesentlich kompakter in zwei sog. Minimierungsregeln zusammenfassen, die wir im Folgenden kennenlernen wollen. Dabei können mit der ersten Regel, die auch sehr einfach und schnell anzuwenden ist, die meisten Zustände bereits zusammengefasst werden. Die Anwendung der zweiten Minimierungsregel ist schwieriger und aufwändiger, insbesondere bei größeren Zustandsräumen. Es ist daher zu empfehlen, zunächst mit der ersten Regel den Zustandsraum soweit wie möglich zu reduzieren, so dass die zweite Regel dann nur noch auf den bereits reduzierten Zustandsraum anzuwenden ist. Während die erste Regel unvollständig minimiert, schöpft die zweite Regel in jedem Fall alle Minimierungsmöglichkeiten aus. Um also sicher den minimalen Zustandsraum zu ermitteln, muss in jedem Fall die zweite Regel angewendet werden.

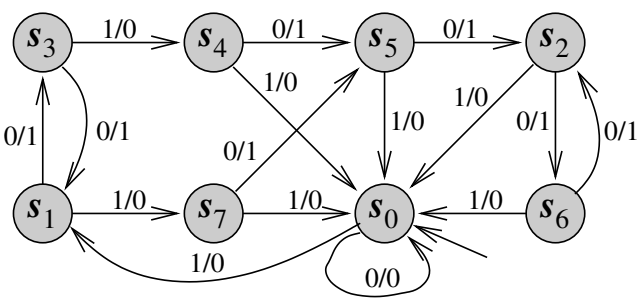
Wir werden öfters davon ausgehen, dass die Minimierung mit der ersten Regel ausreichend ist, so dass die zweite, erheblich aufwändigere nicht mehr angewendet werden muss. Wir wollen daher zum Abschluss die beiden Regeln noch kennenlernen und an einem Beispiel anwenden.

Mit der ersten Minimierungsregel werden diejenigen Reduktionsmöglichkeiten erfasst, die bei einer direkten Äquivalenz zweier Zustände auftritt. Dabei kann einer der äquivalenten Zustände eliminiert werden, der verbleibende Zustand ist Ersatz des eliminierten Zustandes. Mit der zweiten Minimierungsregel können auch Minimierungen erfasst werden, bei denen Zustände einer Klasse zu einem neuen Zustand zusammengefasst werden können. Diese beiden Fälle hatten wir bisher nicht unterschieden.

1. Minimierungsregel
Wenn zwei Zustände eines Automaten bei identischen Eingaben denselben Folgezustand s^{n+1} haben und dabei identische Ausgaben z^{n+1} erzeugen, dann sind diese zwei Zustände **äquivalent**. *Auf einen der beiden Zustände kann verzichtet werden. Dabei müssen alle Kanten, die vorher auf dem eliminierten Zustand endeten, jetzt auf dem dazu äquivalenten verbliebenen Zustand enden.*

Bei dieser Modifikation hat sich der Zustandsraum möglicherweise so verändert, dass eine weitere vorher nicht mögliche Anwendung dieser Regel möglich geworden ist. Die Regel kann fortlaufend wiederholt so oft angewendet werden, bis sich keine weiteren Vereinfachungsmöglichkeiten mehr ergeben.

Beispiel 4.48 *Anwendung der 1. Minimierungsregel*
Gegeben sei die endliche Maschine EM$= (S, E, \delta, \gamma, s_0,)$, die durch
$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$, $E = \{0,1\}$, $Z = \{0,1\}$, die folgende Übergangsfunktion δ und Ausgabefunktion γ beschrieben ist.



δ	0	1
s_0	$s_0/0$	$s_1/0$
s_1	$s_3/1$	$s_7/0$
s_2	$s_6/1$	$s_0/0$
s_3	$s_1/1$	$s_4/0$
s_4	$s_5/1$	$s_0/0$
s_5	$s_2/1$	$s_0/0$
s_6	$s_2/1$	$s_0/0$
s_7	$s_5/1$	$s_0/0$

Gesucht sind äquivalente Zustände in dieser EM.

Zur Anwendung der ersten Regel können wir jeden Zustand mit jedem anderen vergleichen und deren Äquivalenz prüfen. Dabei erkennen wir, dass die Zustände s_5 und s_6 äquivalent sind, denn bei einer Eingabe 0 ergibt sich jeweils der Folgezustand s_2 und die Ausgabe 1, bei einer Eingabe 1 ergibt sich jeweils der Folgezustand s_0 und die Ausgabe 0. Man kann dies kurz durch die folgende Symbolik ausdrücken:

$$s_5 \xrightarrow{0/1} s_2 \qquad s_5 \xrightarrow{1/0} s_0$$
$$s_6 \xrightarrow{0/1} s_2 \qquad s_6 \xrightarrow{1/0} s_0$$

Folglich kann der Zustand s_6 eliminiert werden. Die Transition von Zustand s_2 nach Zustand s_6 endet somit in Zustand s_5. Ebenso kann dieser Reduzierungsschritt auch in einer Tabelle durchgeführt werden. Man erkennt, dass die Zeilen für s_5 und s_6 identisch sind, und kann eine von beiden weglassen. In der Folgetabelle muss dann der eliminierte Zustand s_6 jeweils durch s_5 ersetzt werden.

δ	0	1
s_0	$s_0/0$	$s_1/0$
s_1	$s_3/1$	$s_7/0$
s_2	$s_6/1$	$s_0/0$
s_3	$s_1/1$	$s_4/0$
s_4	$s_5/1$	$s_0/0$
s_5	$s_2/1$	$s_0/0$
s_6	$s_2/1$	$s_0/0$
s_7	$s_5/1$	$s_0/0$

$\Longrightarrow$

δ	0	1
s_0	$s_0/0$	$s_1/0$
s_1	$s_3/1$	$s_7/0$
s_2	$s_5/1$	$s_0/0$
s_3	$s_1/1$	$s_4/0$
s_4	$s_5/1$	$s_0/0$
s_5	$s_2/1$	$s_0/0$
s_7	$s_5/1$	$s_0/0$

$\Longrightarrow$

δ	0	1
s_0	$s_0/0$	$s_1/0$
s_1	$s_3/1$	$s_2/0$
s_2	$s_5/1$	$s_0/0$
s_3	$s_1/1$	$s_2/0$
s_5	$s_2/1$	$s_0/0$

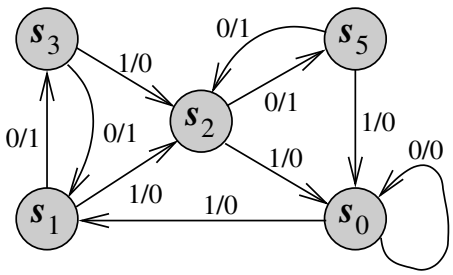
In der reduzierten Tabelle erkennen wir, dass wiederum die Zustände s_2, s_4 und s_7 äquivalent sind. Wir können die Zustände s_4 und s_7 eliminieren. Danach ist eine weitere Anwendung der 1. Minimierungsregel nicht mehr möglich. □

Die 2. Minimierungsregel prüft nun, welche Zustände bei Beibehaltung des Verhaltens durch Bildung von Klassen ersetzt werden können. Dabei muss im Gegensatz zur 1. Regel keiner der ursprünglichen Zustände erhalten bleiben.

2. Minimierungsregel
Zustände mit gleichen Ausgaben können zu Klassen zusammengefasst werden. Innerhalb einer Klasse können Zustände, deren Folgezustände in Abhängigkeit der Eingabe innerhalb der gleichen Klasse liegen, zu neuen Zuständen zusammengefasst werden.

Zur Anwendung der 2. Minimierungsregel werden in einer Tabelle Klassen von Zuständen mit identischen Ausgaben zusammengefasst. Diese Klassen werden fortlaufend nummeriert. Für jeden Zustand wird bestimmt, in welcher Klasse der Folgezustand in Abhängigkeit der Eingabe liegt. Ist für alle Zustände innerhalb einer Klasse die Folgeklasse in Abhängigkeit der Eingabe gleich, so kann diese Klasse zu einem Zustand zusammengefasst werden. Klassen mit Zuständen, die unterschiedliche Folgeklassen haben, werden weiter aufgeteilt und neu bezeichnet. Das Verfahren ist beendet, sobald alle Folgeklassen von Zuständen einer Klasse identisch sind.

Beispiel 4.49 *Anwendung der 2. Minimierungsregel*



Die vereinfachte EM aus Beispiel 4.48 wird mit der 2. Minimierungsregel weiter vereinfacht. Aus der Zustandstabelle des vereinfachten Graphen werden dazu Klassen von Zuständen mit gleichem Folgeausgangssignal y^{n+1} gebildet.

δ	0	1
s_0	$s_0/0$	$s_1/0$
s_1	$s_3/1$	$s_2/0$
s_2	$s_5/1$	$s_0/0$
s_3	$s_1/1$	$s_2/0$
s_5	$s_2/1$	$s_0/0$

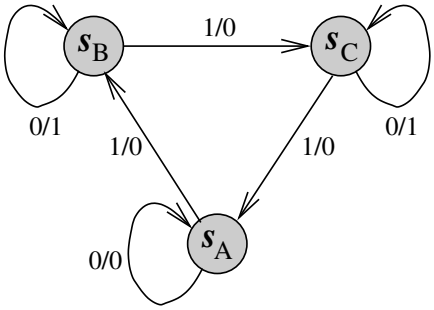
Klasse	[1]		[2]							
Zustand	s_0		s_1		s_2		s_3		s_5	
Eingabe	0	1	0	1	0	1	0	1	0	1
Ausgabe	0	0	1	0	1	0	1	0	1	0
Folgeklasse	[1]	[2]	[2]	[2]	[2]	[1]	[2]	[2]	[2]	[1]

Da zwei verschiedene Ausgangssignalfolgen $\gamma((s_0,0),(s_0,1)) = (0,0)$ und $\gamma((s_1,0),(s_1,1)) = (1,0)$ auftreten, entstehen die beiden dargestellten Klassen [1] und [2]. In der letzten Zeile der Tabelle ist angegeben, in welcher Klasse der Folgezustand liegt. Aus Zustand s_2 beispielsweise geht die Maschine mit der Eingabe 0 in den in Klasse [2] liegenden Zustand s_5, mit der Eingabe 1 in den in Klasse [1] liegenden Zustand s_0 über. Die Klasse [1] enthält nur eine Folgeklassenkombination, nämlich [1] [2]. Die Folgeklassen in der Klasse [2] sind unterschiedlich und entsprechen somit nicht der Forderung nach Gleichheit der Folgeklassen innerhalb einer Klasse. Die Maschine weist also auf jeden Fall mehr als zwei Zustände auf. Das Verfahren ist weiterzuführen.

Aus den bestehenden Klassen werden wieder neue Klassen von Zuständen, die für die gleiche Eingangskombination die gleiche Folgeklasse besitzen, gebildet. In der Tabelle bilden in Klasse [2] offensichtlich die Zustände s_1 und s_3 bzw. die Zustände s_2 und s_5 zwei neue Klassen. Es entsteht eine neue Tabelle:

Klasse	$[A]$		$[B]$				$[C]$			
Zustand	s_0		s_1		s_3		s_2		s_5	
Eingabe	0	1	0	1	0	1	0	1	0	1
Ausgabe	0	0	1	0	1	0	1	0	1	0
Folgeklasse	$[A]$	$[B]$	$[B]$	$[C]$	$[B]$	$[C]$	$[C]$	$[A]$	$[C]$	$[A]$

In dieser Tabelle haben sich wegen der Klassenneubildung natürlich auch die Folgeklassen geändert. Diese müssen bei jedem Schritt neu gebildet werden. Da in dieser Tabelle in allen Klassen die Folgeklassen identisch sind, ist das Verfahren beendet. Die Anzahl Zustände ist minimal. Die Klassen $[A]$, $[B]$ und $[C]$ bilden jeweils einen Zustand. Es ergibt sich ein reduziertes Diagramm mit 3 Zuständen.



□

Aufgabe 4.12 *Reduzierung der Anzahl Zustände*
Nehmen Sie das vollständige Zustandsdiagramm des zu Beginn der KE 4 betrachteten Getränkeautomaten in Abb. 4.4 auf Seite 11 und reduzieren Sie die Anzahl der Zustände schrittweise soweit wie möglich mit der 1. Minimierungsregel.
Reicht die 1. Minimierungsregel für eine vollständige Minimierung aus? ◇

4.6 Berechenbarkeit und Komplexität (von Jörg Keller)

In diesem Abschnitt wollen wir uns mit der Frage beschäftigen, ob wir jedes Problem, das sich mathematisch formulieren lässt, mit einem Computer lösen können. Wir werden an einem Beispiel sehen, dass wir das nicht können. Weiterhin wollen wir uns mit der Frage befassen, ob wir alle berechenbare Probleme effizient mit einem Computer lösen können. Dies können wir vermutlich nicht. Interessanterweise gehören viele interessante Problemstellungen der Wirtschaftsinformatik zu den Problemen, die wir vermutlich nicht effizient lösen können. Wir nennen diese Probleme NP-vollständig. Tröstlicherweise kennt man aber für viele der NP-vollständigen Probleme Algorithmen, die effizient sind und die Probleme immerhin näherungsweise lösen.

Diese Betrachtungen stellen, obwohl sie auch intellektuell sehr stimulierend sind, keinen Selbstzweck im akademischen Elfenbeinturm dar. Für den (Wirtschafts–)Informatiker ist es wichtig zu erkennen, wann sie oder er ein NP-vollständiges Problem vor sich hat. In diesem Fall macht es nämlich keinen Sinn, einen effizienten Algorithmus programmieren zu wollen der das Problem exakt löst. Einen solchen gibt es vermutlich nicht. Allerdings braucht man auch den Kopf nicht hängen zu lassen: man muss nicht Jahrhunderte darauf warten, dass ein exakter aber nicht effizienter Algorithmus terminiert: Approximationsalgorithmen und Heuristiken schaffen einen Ausweg.

In der Darstellung orientieren wir uns weitestgehend an der neuesten Ausgabe des Klassikers von Hopcroft und Ullman, mittlerweile verstärkt durch R. Motwani [2].

4.6.1 Berechenbarkeit

Um darüber zu reden, welche Probleme berechenbar sind, müssen wir uns zunächst auf ein Modell unserer Rechenmaschine einigen. Hierzu haben sich im Laufe der letzten Jahrzehnte mehrere Modelle etabliert: universelle Turing-Maschinen, unbeschränkte Grammatiken, partiell-rekursive Funktionen. Es hat sich allerdings gezeigt, dass alle diese Modelle die gleiche Mächtigkeit haben, so dass es egal ist, welches man benutzt. Die *Church-Turing-These*, benannt nach A. Church und A. Turing, besagt dass alle Probleme, die man überhaupt berechnen kann, die sind, die man mit obigen Modellen berechnen kann. Es scheint also auch nicht sinnvoll zu sein, weitere Modelle einzuführen.

Wir benutzen als Modell eines Rechners einen normalen PC mit Betriebssystem und Compiler. Der Rechner erhält ein Programm der Programmiersprache C und eine Eingabe für dieses Programm. Der Rechner übersetzt das C-Programm mit dem Compiler in ein ausführbares Programm und führt es mit der obigen Eingabe aus. Wir nehmen an, dass dieses Programm nur von der Standard-Eingabe liest und nur auf die Standard-Ausgabe schreibt.

Man mag nun einwenden, dass ein solcher Rechner ja nicht mit beliebigen ganzen Zahlen sondern nur mit Zahlen fester Länge (üblicherweise 32 Bit) arbeiten kann, auch nur einen endlichen Hauptspeicher besitzt, und somit ein endlicher Automat ist. Dem kann man entgegnen, dass die meisten Programmiersprachen mittlerweile beliebig lange Darstellungen von Zahlen unterstützen, die dann mehrere Speicherworte belegen, und dass man, wenn der Hauptspeicher nicht ausreicht, Daten auf die Festplatte auslagern kann. Sollte die nicht ausreichen, hängt man eine neue Festplatte an und archiviert die alte bis man sie wieder braucht. Jede weitere Erweiterung des Rechners kostet zwar zusätzlichen Aufwand, ist aber prinzipiell möglich, und entspricht den unendlich langen Bändern in Bandlaufwerken, mit denen die Turing-Maschine arbeitet.

Unser Beispiel für eine Problemstellung, die nicht in allen Fällen durch einen Computer entschieden werden kann, ist die trivial erscheinende Frage, ob ein Programm, das durch seinen Quelltext in der Programmiersprache C gegeben ist, bei einer ebenfalls gegebenen Eingabe als erstes „Hallo“ ausgeben wird oder nicht. Wir nehmen dabei zur Vereinfachung an, dass Ausgaben nur über die Funktion `printf` erfolgen. Offensichtlich ist diese Frage für das nun in der linken Spalte folgende Programm zu bejahen, und für das Programm in der rechten Spalte zu verneinen. Beide Programme sollen keine Eingabe (leeres Eingabewort) erhalten.

```
#include "stdio.h"
main(){
printf("Hallo");
}
```

```
#include "stdio.h"
main(){
printf("Bello");
}
```

Allerdings soll ja für ein beliebiges C-Programm entschieden werden, ob die Ausgabe mit „Hallo“ beginnt. Eine erste Überlegung führt dahin, dass man das zu testende C-Programm compiliert und ausführt bis entweder „Hallo“ ausgegeben ist, oder die Ausgabe sich von „Hallo“ unterscheidet, zum Beispiel weil als zweiter Buchstabe „e“ ausgegeben wird, oder bis das Programm terminiert. Je nachdem gibt man dann im ersten Fall „Ja“ aus, in den anderen beiden Fällen „Nein“. Das sollte auch ein Programm können, womit das Problem berechenbar wäre. Was macht man aber mit folgendem Programm, das wiederum keine Eingabe erhält?

```
#include "stdio.h"
main(){
while(1){;}
}
```

Dieses Programm führt eine Endlos-Schleife aus, ohne etwas auszugeben. Während man in diesem Fall noch argumentieren kann, dass ein Compiler dies durch Programmanalyse erkennen kann und somit „Nein“ auszugeben wäre, ist der folgende Fall deutlich schwieriger, bei dem das Programm eine natürliche Zahl n als Eingabe erhält.

```
#include "stdio.h"
main(){
 int n,a,b,c,t=3;
 scanf("%d",&n);
 while(1){
  for(a=1;a<t-1;a++) for(b=1;b<t-a;b++){
   c=t-a-b;
   if(ex(a,n)+ex(b,n)==ex(c,n)) printf("Hallo");
  }
  t++;
 }
}
```

Dieses Programm testet nach und nach, ob es für ein gegebenes n drei natürliche Zahlen a,b,c gibt so dass $a^n + b^n = c^n$ gilt. Für $n = 1$ findet das Programm sehr schnell $a = 1$, $b = 2$, $c = 3$ und gibt „Hallo“ aus. Für $n = 2$ findet das Programm $a = 3$, $b = 4$, $c = 5$ (es gilt $3^2 + 4^2 = 5^2$) und gibt „Hallo“ aus. Was passiert aber bei $n \geq 3$? Die Frage, ob es dann eine Lösung gibt (nur dann terminiert das Programm) kennt man als Fermats letztes Problem, und es hat etwa 300 Jahre gedauert, bis Mathematiker bewiesen, dass es bei $n \geq 3$ keine positiv-ganzzahlige Lösung obiger Gleichung gibt. Damit reicht also auch die Idee mit der Compileranalyse nicht.

Wir nehmen nun an, dass es ein Programm P gäbe, das das Hallo-Ausgabe-Problem bei jedem C-Programm entscheiden könne. Wir führen diese Annahme zu einem Widerspruch und zeigen so, dass das Hallo-Ausgabe-Problem nicht durch einen Computer entscheidbar ist.

Anmerkung: Meist benutzt man in diesem Beweis die ähnliche aber abstrakte Fragestellung, ob ein Programm terminert. Dieses Problem ist als das nicht-entscheidbare *Halteproblem* berühmt geworden.

Wenn es ein Programm P gäbe, das das Hallo-Ausgabe-Problem entscheidet, dann muss es irgendwo ein `printf` Kommando haben, das „Nein“ ausgibt, oder aber zumindest das „N“ davon. Dann könnten wir P sicher so zu einem Programm P' ändern, dass wir an dieser Stelle „Hallo“ ausgeben. Analysiert das hypothetische P' ein Programm Q mit einer gewissen Eingabe E, dann gibt P' also „Ja“ aus, wenn Q „Hallo“ ausgibt, und P' gibt „Hallo“ aus, wenn Q nicht „Hallo“ ausgibt.

Nun ändern wir das hypothetische Programm P' zu P'' auf folgende Weise: P'' liest nur noch den Quelltext von Q, aber nicht mehr die Eingabe E für Q. Stattdessen kopiert es das Programm Q und benutzt das stattdessen als Eingabe. Danach arbeitet P'' wie P', allerdings wird statt der Eingabe E nun Q selbst benutzt.

Als Programm Q, das von P'' analysiert wird, wählen wir nun P'' selbst.

Das hypothetische Programm P'' würde, wenn es denn existierte, nun entweder „Ja“ oder „Hallo“ ausgeben. Wenn P'' bei Eingabe $Q = P''$ „Ja“ ausgibt, dann hat das Programm Q bei Eingabe Q „Hallo“ ausgegeben. Das Programm Q ist aber gerade P'' selbst! Also gibt P'' bei Eingabe P'' „Hallo“ aus. Das kann aber nicht sein, da wir gerade damit gestartet sind, dass P'' bei Eingabe $Q = P''$ doch „Ja“ ausgibt. Hier findet sich also ein Widerspruch.

Die gleiche Überlegung mit dem gleichen Widerspruch kann man tätigen, wenn man davon ausgeht, dass P'' bei Eingabe $Q = P''$ „Hallo“ ausgibt.

Da das hypothetische Programm P'' nur zwei verschiedene Ausgaben machen kann, und jede von ihnen zu einem Widerspruch führt, kann es das Programm P'' und damit auch P' und P nicht geben. Wenn es aber kein Programm P geben kann, dann ist das Hallo-Ausgabe-Problem nicht durch einen Rechner entscheidbar.

Vermutlich müssen Sie diese Konstruktion mehrmals durcharbeiten, bis Sie sie verstehen. Zum Trost sei Ihnen gesagt, dass sich die meisten Menschen mit dieser Art Beweis sehr schwer tun.

Aufgabe 4.13 *Berechenbarkeit*
Leiten Sie Sie den Widerspruch für den Fall her, dass das hypothetische Programm P'' bei Eingabe $Q = P''$ „Hallo“ ausgibt. ◇

Es gibt noch weitere Fragestellungen die sich in der obigen Modellwelt sehr abstrakt formulieren lassen, die aber auch in der Praxis Auswirkungen haben. Ein Beispiel dafür ist der *Fixpunktsatz* von Kleene. Dieser besagt anschaulich formuliert, dass es zu jedem C-Programm, das als Eingabe ein Programm erhält, dieses Programm verändert und das veränderte Programm ausgibt, eine Eingabe gibt, die nach ihrer Veränderung das gleiche tut wie vorher. Eine Anwendung dieses Satzes in der Praxis ist die Garantie, dass es in jeder Programmiersprache ein Programm geben muss, dass seinen eigenen Quelltext ausgibt. Ein solches Programm nennt man *Quine*.

Aufgabe 4.14 *Berechenbarkeit*
Finden Sie einen Quine in der Programmiersprache C (im Zweifelsfall durch Googeln im Web) und beschreiben Sie weshalb er funktioniert. ◇

4.6.2 Komplexität

Wenn wir einen Algorithmus zur Lösung eines Problems haben, dann messen wir seine Laufzeit. In der theoretischen Analyse messen wir seine Laufzeit nicht in Sekunden, sondern in einer Anzahl von Schritten, wobei wir annehmen, dass ein Schritt eine feste Zeit dauert, deren Größe lediglich vom verwendeten Rechner abhängt. Wenn wir die Größe des Problems parametrisieren (typischerweise bezeichnet mit der Zahl n), dann messen wir die Anzahl der Schritte in Abhängigkeit von n. Nun gibt es aber vermutlich viele Probleme der Größe n. Wir geben nun die schlechteste Laufzeit über alle Probleme der Größe n an. Man nennt dies eine *worst case Analyse*.

Wollen wir beispielsweise n Zahlen aufsummieren, die in einem Feld im Speicher des Rechners vorliegen, dann benötigen wir dabei sicher höchstens $c \cdot n$ Schritte, wobei c eine Konstante ist, die von der genauen Arbeitsweise unseres Algorithmus abhängt. Wollen wir n Zahlen sortieren, so gibt es Algorithmen, die höchstens $c_1 \cdot n^2$ Schritte brauchen (z.B. Bubblesort), und welche, die $c_2 \cdot n \cdot \log_2 n$ Schritte brauchen (z.B. Heapsort). Für genügend großes n sind die letzteren Algorithmen sicher schneller als die ersten, egal wie die Konstanten aussehen. Um auf die Konstanten verzichten zu können, benutzen wir die asymptotische Notation mit den Landau'schen Symbolen. Wir schreiben $O(n^2)$ statt $c_1 \cdot n^2$ und $O(n \log n)$ statt $c_2 \cdot n \cdot \log_2 n$.

Aufgabe 4.15 *Komplexität*
Zeigen Sie, dass man das Produkt $A \cdot b$ aus einer $n \times n$-Matrix A und einem n-elementigen Vektor b in $O(n^2)$ Schritten berechnen kann. ◇

Wir definieren die Klasse $\mathcal{P}$ der effizient lösbaren Probleme als die Menge aller Probleme, so dass es für jedes Problem aus $\mathcal{P}$ einen Algorithmus gibt, der zur Lösung des betreffenden Problems der Größe n nicht mehr als $O(n^k)$ viele Schritte benötigt, wobei k eine vom Problem abhängige, feste natürliche Zahl ist. Die Klasse $\mathcal{P}$ enthält also alle Probleme die in einer Zeit gelöst werden können, die polynomiell in der Problemgröße ist. Damit gehören zum Beispiel die folgenden Probleme zur Klasse $\mathcal{P}$:
das Problem, n Zahlen aufzusummieren, das Problem n Zahlen zu sortieren, und das Problem eine $n \times n$-Matrix mit einem n-elementigen Vektor zu multiplizieren.

Nun gibt es allerdings Probleme, für die man noch keinen Algorithmus gefunden hat, dessen Laufzeit polynomiell in der Problemgröße ist. Ein Beispiel ist das Problem des Handlungsreisenden (Travelling Salesman Problem, TSP) mit Schranke. Gegeben ist eine Liste von n Städten, sowie für jedes Paar (a,b) von Städten a und b (wobei $a \neq b$ gelten soll) die Länge der Reise von a nach b, und schließlich eine maximale Reisedauer W. Gefragt ist nun, ob es eine Rundreise mit Reisedauer höchstens W gibt, d.h. eine Reise die ausgehend von einer Stadt a alle anderen Städte besucht, allerdings keine mehrfach, und schließlich wieder in a endet, und die nicht länger als W dauert. Dieses Problem ist ein typisches Modell für viele Fragestellungen der Tourenplanung in Unternehmen. Bei dem Problem TSP ohne Schranke sucht man übrigens die kürzeste Rundreise.

Ein einfacher (aber nicht effizienter) Algorithmus zur Lösung des Problems arbeitet wie folgt. Nacheinander erzeugen wir alle Reihenfolgen der n Städte, und für jede Reihenfolge berechnen wir die Dauer der Rundreise. Sobald wir eine Rundreise mit Länge $\leq W$ gefunden haben, geben wir diese aus. Ansonsten geben wir nach der letztmöglichen Reihenfolge aus, dass es keine Rundreise mit Länge $\leq W$ gibt. Für jede Reihenfolge dauert die Berechnung der Reisedauer $O(n)$ Schritte, da wir einfach die n Reisedauern von Stadt1 zu Stadt2, von Stadt2 zu Stadt3,..., von Stadt$n$ zu Stadt1 addieren müssen. Allerdings gibt es

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$$

Anordnungen von n Städten, da man für die Wahl der Stadt1 n Möglichkeiten hat, für die Stadt2 hat man noch $n-1$ Möglichkeiten usw. Das Symbol $n!$ nennt man übrigens n-Fakultät. Da jeder Faktor des obigen Produkts außer dem letzten mindestens die Größe 2 hat, gilt sicher $n! \geq 2^{n-1}$. Da für $n \geq 4$ der erste Faktor sogar größer als $2 \cdot 2$ ist, gilt

$$n! \geq 2^n \text{ für } n \geq 4$$

Damit ist die Laufzeit unseres Algorithmus exponentiell in der Problemgröße und nicht mehr polynomiell. Für dieses Problem ist bisher auch kein Algorithmus mit polynomieller Laufzeit bekannt.

Aufgabe 4.16 *Komplexität*
Zeigen Sie, dass Algorithmen mit einer exponentiellen Laufzeit von 2^n Schritten nur für winzige Problemgrößen n einsetzbar sind. Schätzen Sie dazu die Laufzeit des Algorithmus für $n = 60$ ab, wenn Sie annehmen, das $2^{10} \approx 10^3$, dass Sie einen Rechner haben, der 10^{10} Schritte in einer Sekunde machen kann, dass ein Tag knapp 10^5 Sekunden hat, und dass drei Jahre etwa 10^3 Tage haben. ◇

Wir definieren nun die Klasse $\mathcal{NP}$ als die Menge der Probleme, so dass es für jedes Problem aus $\mathcal{NP}$ einen Algorithmus gibt, der für eine vorgegebene potentielle Lösung dieses Problems in polynomieller Zeit entscheiden kann, ob diese potentielle Lösung auch wirklich eine Lösung des Problems darstellt. Das Problem TSP gehört sicher zur Klasse $\mathcal{NP}$, denn wenn eine Lösung, d.h. eine Reihung von Städten gegeben ist, kann man in $O(n)$ Schritten testen, ob die entsprechende Rundreise eine Dauer von $\leq W$ hat. Das $\mathcal{N}$ in $\mathcal{NP}$ steht übrigens für nicht-deterministisch. Man stellt sich dabei eine Maschine vor, die nicht-deterministisch, sozusagen in Nullzeit und mit göttlicher Eingebung, eine Lösung rät und diese Lösung nur noch in polynomieller Laufzeit validiert werden muss. Für Probleme in $\mathcal{NP}$ gibt es in der Regel einen Lösungsalgorithmus mit exponentieller Laufzeit.

Anmerkung: Wir tun hier so, als seien alle Funktionen, die nicht polynomiell in n sind, exponentiell in n. Dies ist nicht so, zum Beispiel ist $n^{\log n}$ nicht polynomiell aber kleiner als exponentiell. Wir vereinfachen uns das Leben aber an dieser Stelle ein bisschen.

Anmerkung: Es gilt offensichtlich $\mathcal{P} \subset \mathcal{NP}$, denn bei einem Problem für das wir in polynomieller Zeit eine Lösung berechnen können wir sicher auch in polynomieller Zeit prüfen, ob diese Lösung wirklich eine ist.

Eine der grundlegenden Fragen der theoretischen Informatik, die bis heute nicht beantwortet ist, ist die Frage ob $\mathcal{P} = \mathcal{NP}$. In diesem Fall gäbe es für Probleme wie TSP doch Algorithmen mit polynomieller Laufzeit, nur hätte man sie halt noch nicht gefunden. Man glaubt heute, dass diese Gleichheit nicht gilt. Allerdings gibt es immer wieder Überraschungen, d.h. ab und an wird auch für ein Problem, von dem man glaubt dass es zu $\mathcal{NP}$ aber nicht zu $\mathcal{P}$ gehört, doch ein Algorithmus mit polynomieller Laufzeit gefunden. Ein Beispiel hierfür stellt der erste deterministische Primzahltest dar, dessen Laufzeit polynomiell in der Anzahl der Stellen der zu testenden Zahl ist, und der 2002 unter dem Namen AKS-Test erstmalig veröffentlicht wurde [4].

Wenn wir glauben, dass es Probleme gibt, für die kein effizienter Algorithmus existiert, dann sollten wir in der Lage sein, solche Probleme zu erkennen, wenn sie uns begegnen, denn in diesem Fall wäre das weitere Suchen nach einem solchen Algorithmus vergebliche Mühe. Damit beschäftigen wir uns im Rest dieses Abschnitts. Allerdings brauchen wir auch, wenn wir ein solches Problem identifiziert haben, nicht einfach aufzugeben, denn es gibt Approximationsalgorithmen und Heuristiken, auf die wir kurz im nächsten Abschnitt 4.6.3 eingehen wollen.

Wenn man den Verdacht hat, dass ein vorliegendes Problem A zur Menge $\mathcal{NP}$ gehören könnte[1] dann benutzt man zum Nachweis die Methode der polynomiellen Reduktion, kurz p-Reduktion genannt. Man sucht sich ein Problem B, von dem bekannt ist, dass es zu $\mathcal{NP}$ gehört, und überlegt sich einen Algorithmus, mit dem man die Eingabe von B so transformieren kann, dass sie zu einer Eingabe für A wird, und wie man eine Lösung von A für die transformierte Eingabe so zurück transformieren kann, dass sie zu einer Lösung für die ursprüngliche Eingabe von B wird. Wenn man einen solchen Algorithmus mit polynomieller Laufzeit findet, dann muss A ebenfalls zu $\mathcal{NP}$ gehören. Denn würde A trotzdem zu $\mathcal{P}$ gehören, d.h. gäbe es einen Algorithmus mit polynomieller Laufzeit zur Lösung von A, dann könnte man mittels der Transformation auch B in polynomieller Zeit lösen, und auch B würde zu $\mathcal{P}$ gehören. B war aber gerade schon als zu $\mathcal{NP}$ zugehörig bekannt, und nach gängiger Meinung gilt $\mathcal{P} \neq \mathcal{NP}$.

Als Beispiel für einen Kandidaten für $\mathcal{NP}$ dient das folgende Problem der unabhängigen Mengen. Gegeben sei ein ungerichteter Graph G mit n Knoten und eine Zahl $1 \leq k \leq n$. Eine unabhängige Menge des Graphen ist eine Teilmenge der Knoten, wobei keine Knoten der Teilmenge durch Kanten verbunden sind. Es ist ein Algorithmus gesucht, der feststellt, ob der Graph G eine unabhängige Menge mit mindestens k Knoten enthält. Zur Lösung des Problems können wir alle k-elementigen Teilmengen der Knotenmenge aufzählen und für jede prüfen, ob sie eine unabhängige Menge ist, indem wir testen, ob es für irgendein Knotenpaar der Teilmenge eine Kante im Graphen gibt. Der Test geht in polynomieller Zeit, also ist das Problem in $\mathcal{NP}$, und „riecht“ schon nach einem Problem das nicht zu $\mathcal{P}$ gehört. Zum Nachweis muss man ein bekanntes Problem aus $\mathcal{NP}$ darauf reduzieren. Hierzu nimmt man in der Regel das 3SAT-Problem. Da die Konstruktion aber aufwändig ist werden wir sie hier nicht durchführen. Wichtig ist, dass man das Problem, von dem man schon weiß dass es zu $\mathcal{NP}$ gehört, auf das neue Problem reduziert und nicht umgekehrt. Denn wenn man das neue Problem auf das bekannte reduziert, zeigt man gar nichts. Es könnte ja zusätzlich einen ganz anderen Algorithmus mit polynomieller Laufzeit zur Lösung des neuen Problems geben.

Innerhalb der Probleme in der Menge $\mathcal{NP}$ gibt es einige Probleme, die zentral sind, die sogenannten $\mathcal{NP}$-vollständigen Probleme. Ein $\mathcal{NP}$-vollständiges Problem ist ein Problem, das zur Menge $\mathcal{NP}$ gehört und für jedes Problem aus $\mathcal{NP}$ gibt es eine p-Reduktion auf dieses Problem. Die $\mathcal{NP}$-vollständigen Probleme sind gerade die Probleme, die in $\mathcal{NP}$, aber nicht in $\mathcal{P}$ sind, denn wenn ein $\mathcal{NP}$-vollständiges Problem A in $\mathcal{P}$ enthalten wäre, dann würde $\mathcal{P} = \mathcal{NP}$ gelten, da jedes Problem in $\mathcal{NP}$ zunächst in polynomieller Zeit auf A reduziert werden könnte (denn A ist $\mathcal{NP}$-vollständig),

[1] Einen solchen Verdacht kann man haben, wenn man trotz Mühe und Recherche keinen effizienten Algorithmus für das Problem findet, und es vielleicht bekannten Problemen aus $\mathcal{NP}$ ähnelt. Zugegebenermaßen erfordert dies aber typischerweise Erfahrung.

und A in polynomieller Zeit gelöst werden kann (denn A ist in $\mathcal{P}$). Wir wollen hier nicht darauf eingehen wie man zeigt dass ein Problem $\mathcal{NP}$-vollständig ist, allerdings sollte man den Begriff und die Bedeutung kennen.

Will man zeigen, dass ein Problem A in $\mathcal{NP}$ ist, dann führt man in der Regel die oben beschriebene p-Reduktion nicht mit einem beliebigen Problem B aus $\mathcal{NP}$ durch, sondern von einem $\mathcal{NP}$-vollständigen Problem B nach A durch.

4.6.3 Approximationsalgorithmen und Heuristiken

Für ein Problem aus $\mathcal{NP}$ gibt es vermutlich keinen effizienten Algorithmus der das Problem löst. Allerdings gibt es durchaus Algorithmen, die das Problem vielleicht nicht ganz aber fast lösen. Zum Beispiel ist ein Tourenplaner auch froh, wenn er für das TSP Problem ohne Schranke einen Algorithmus hat, der zwar nicht die kürzeste Rundreise berechnet, der aber in polynomieller Zeit eine Rundreise berechnet, die nie mehr als einen gewissen Prozentsatz länger als die kürzeste Rundreise ist, oder aber wenn er einen Algorithmus hat, der in polynomieller Zeit eine Rundreise berechnet, die in der Regel nicht mehr als einen gewissen Prozentsatz länger als die kürzeste Rundreise ist. Mit „in der Regel“ bezeichnen wir die Situation, dass man zwar Städtekonstellationen angeben kann, bei denen die Rundreise sehr viel länger als die kürzeste Reise ist, dass aber für die Touren, die in der Praxis vorkommen, solche Fälle eigentlich nicht vorkommen.

Algorithmen, bei denen man beweisen kann, dass sie nie mehr als ein gewisses Quantum von der optimalen Lösung abweichen, nennt man *Approximationsalgorithmen*. Ist dieses Quantum ein Prozentsatz der optimalen Lösung, spricht man von einer *relativen Gütegarantie*. Algorithmen, bei denen man einen solchen Beweis nicht führen kann, nennt man *Heuristiken*. Es gibt auch Heuristiken dergestalt, dass die Heuristik zwar stets die optimale Lösung berechnet, allerdings in manchen Fällen dazu eine exponentielle Laufzeit benötigt, obwohl sie in der Regel mit einer polynomiellen Laufzeit auskommt.

Ein Beispiel für einen Approximationsalgorithmus ist der Algorithmus von Christofides für das metrische TSP, das ebenfalls in $\mathcal{NP}$ liegt. Beim metrischen TSP gilt für drei unterschiedliche Städte a, b, c immer die Dreiecksungleichung

$$\text{dist}(a,b) \leq \text{dist}(a,c) + \text{dist}(c,b) \ .$$

Hierbei ist $\text{dist}(a,b)$ die Reisedauer von a nach b. Die Dreiecksungleichung formalisiert die Erfahrung, dass es eigentlich nicht schneller gehen kann von a erst nach c und dann nach b zu reisen als von a direkt nach b zu reisen. Ausnahmen (von a nach c und c nach b gibt es eine ICE-Strecke, aber auf der direkten Strecke von a nach b gibt es nur eine Bimmelbahn) bestätigen die Regel. Der Algorithmus von Christofides erzeugt mit einer polynomiellen Laufzeit eine Tour die höchstens 1,5 mal so lang ist wie die optimale Tour.

Ein Beispiel für eine Heuristik der ersten Art ist der First-Fit-Decreasing-Algorithmus für das Bin-Packing Problem. Er ist ein Repräsentant der sogenannten *greedy* Algorithmen. Dies sind Algorithmen, die gar nicht erst mehrere Möglichkeiten ausprobieren sondern sich gleich auf eine Vorgehensweise festlegen. Das macht sie sehr kompakt und schnell, allerdings ist das Ergebnis manchmal nicht gut. Das Bin-Packing Problem besteht darin, dass man n Gegenstände verschiedener Größe in eine Reihe von Containern fester Größe packen muss, so dass möglichst wenige Container benötigt werden. Jeder einzelne Gegenstand muss dabei natürlich in einen Container passen. Der First-Fit-Decreasing-Algorithmus sortiert die Gegenstände der Größe nach, wobei die größten Gegenstände zuerst kommen, und packt jeden Gegenstand in den Container, der gerade an der Reihe ist, falls er noch dazupasst, und ansonsten in den nächsten Container.

Ein Beispiel für eine Heuristik der letzten Art ist der Simplex Algorithmus zur Minimierung einer linearen Funktion mit n reellwertigen Variablen, wobei die Variablen eine Reihe von linearen Ungleichungen erfüllen müssen. Obwohl es für das Problem der linearen Programmierung sogar exakte Algorithmen mit polynomieller Laufzeit gibt, ist der Simplex Algorithmus trotzdem weit verbreitet,

weil er in der Praxis am schnellsten ist, auch wenn er bei ungünstigen Eingaben zu einer exponentiellen Laufzeit führen kann.

Approximationsalgorithmen und Heuristiken werden im C-Modul *Approximationsalgorithmen*, bestehend aus den Kursen 01688 und 01689, behandelt.

Aufgabe 4.17 *Laufzeit First-Fit-Decreasing-Algorithmus*
Welche Laufzeit hat der First-Fit-Decreasing-Algorithmus für n Gegenstände, wenn die Gegenstände bereits nach absteigender Größe sortiert sind? ◇

Aufgabe 4.18 *First-Fit-Decreasing-Algorithmus*
Wieviele Container braucht der First-Fit-Decreasing-Algorithmus, wenn man k Gegenstände hat, die jeweils 75% eines Containers füllen, und k Gegenstände, die jeweils 25% eines Containers füllen? Hierbei soll k ein Vielfaches von 4 sein. Wieviele Container würden Sie brauchen? ◇

4.7 Anhang I: Mathematische Definitionen

In dieser Kurseinheit werden einige Begriffe aus der diskreten Mathematik verwendet, die im Folgenden zusammengestellt sind. Dabei werden bei Aussagen und Definitionen zur präzisen mathematischen Formulierung häufig die abkürzenden Zeichen $\neg$ für die Verneinung einer Aussage, $\wedge$ für die UND-Verknüpfung zweier Aussagen, $\vee$ für die ODER-Verknüpfung zweier Aussagen, $\Longrightarrow$ für die Implikation (daraus folgt) sowie der Allquantor $\forall$ (für alle) und der Existenzquantor $\exists$ (es gibt) verwendet.

4.7.1 Mengen

Eine *Menge* ist die Zusammenfassung von irgendwie zusammengehörigen Elementen zu einem Ganzen. Man schreibt $a \in A$, falls a ein Element der Menge A ist. Die Anzahl der Elemente einer Menge kann sowohl endlich als auch unendlich sein. Ist die Menge A endlich, dann schreibt man dafür $A = \{a_1, a_2, ..., a_n\}$, ist die Menge A unendlich, dann schreibt man $A = \{a_1, a_2, a_3, ...\}$.

Eine Menge A kann auch dadurch angegeben werden, dass man die Eigenschaften ihrer Elemente angibt. Ist G eine Grundmenge und E eine bestimmte Eigenschaft, dann gehören zu A alle diejenigen Elemente a aus G, welche diese Eigenschaft besitzen. Abgekürzt:

$$A = \{a \mid a \in G \text{ und } E(a)\} = \{a \in G \mid E(a)\}$$

Weiter verwenden wir für die *Vereinigung* und den *Durchschnitt* zweier Mengen A_1 und A_2 die Schreibweisen:

$$\begin{aligned} A_1 \cup A_2 &= \{a \mid a \in A_1 \vee a \in A_2\}, \\ A_1 \cap A_2 &= \{a \mid a \in A_1 \wedge a \in A_2\}. \end{aligned}$$

Daneben wird insbesondere die leere Menge durch $\emptyset$ und die Menge der natürlichen Zahlen ohne die Null durch $\mathbb{N}$ und mit der Null durch $\mathbb{N}_0$ symbolisiert.

Die Anzahl der Elemente einer Menge A wird mit $|A|$ bezeichnet. T heisst *Teilmenge* von A, abgekürzt durch $T \subseteq A$, wenn jedes Element von T auch zu A gehört. Die Menge aller Teilmengen von A heisst *Potenzmenge* von A, abgekürzt $\mathrm{PM}(A)$. Insbesondere ist $\emptyset \in \mathrm{PM}(A)$ und $A \in \mathrm{PM}(A)$. Ist $|A| = n$, so enthält $\mathrm{PM}(A)$ insgesamt 2^n Teilmengen.
Beispiel: Es sei $A = \{1,2,3\}$, dann ist $\mathrm{PM}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}\{2,3\}, \{1,2,3\}\}$.

Wir betrachten eine endliche Menge A und bilden die Potenzmenge $\mathrm{PM}(A)$ von A. Eine Menge A_M von Mengen $A_M = \{M_1, M_2, ..., M_k\} \subseteq \mathrm{PM}(A)$ nennt man eine *Zerlegung* von A, falls gilt:

$$\bigcup_{i=1}^{k} M_i = A \quad \text{und für alle } i, j \in \{1,2,...,k\} \text{ mit } i \neq j \text{ gilt: } M_i \cap M_j = \emptyset.$$

Eine Zerlegung ist die Aufteilung einer Menge A in paarweise disjunkte, d.h. durchschnittsleere, Teilmengen.

Es seien P_1 und P_2 Zerlegungen von A. Dann sagt man: P_2 ist *feiner* als P_1, abgekürzt $P_2 \prec P_1$, falls für alle $M_2 \in P_2$ ein $M_1 \in P_1$ existiert, für das $M_2 \subseteq M_1$ gilt. Entsprechend sagt man in diesem Fall das P_1 gröber ist als P_2. Das folgende Beispiel zeigt zwei Zerlegungen P_1 und P_2 von A.

A :	P_{11}			P_{12}	P_{13}		P_{14}		P_{15}	P_{16}		P_1
A :	P_{21}	P_{22}	P_{23}	P_{24}	P_{25}	P_{26}	P_{27}	P_{28}	P_{29}	P_{210}	P_{211}	P_2

Die Zerlegung P_2 der Menge A ist feiner als die Zerlegung P_1, es gilt $P_2 \prec P_1$.

4.7.2 Binäre Relation

Eine binäre Relation R ist allgemein eine Menge von Paaren (a,b), bei denen die Elemente a und b in einer bestimmten Beziehung zueinander stehen. Sind $a \in A$ und $b \in B$ zwei Elemente aus den Mengen A bzw. B, dann heißt (a,b) geordnetes Tupel über A und B. Jede Teilmenge R der Menge

$A \times B$ heißt binäre Relation über A und B ($R \subseteq A \times B$). Für uns ist dabei im Folgenden nur die spezielle binäre Relation der Form $R \subseteq A \times A$ von Interesse. Man bezeichnet R dann kurz als Relation auf einer Menge A.

Beispielsweise ist die Relation *hatMatrNr* zwischen Studenten und ihren Matrikelnummern eine Teilmenge von $A \times N$, wobei A die Menge der Studenten und N die Matrikelnummern darstellen.

4.7.3 Eigenschaften binärer Relationen

Im Zusammenhang mit der binären Relation

$$R \subseteq A \times A \equiv \{(a_1, a_2) \mid a_1, a_2 \in A\}$$

sind die folgenden Eigenschaften, die eine solche Relation aufweisen kann, von Bedeutung:

1. *symmetrisch*:
Eine binäre Relation R auf der Menge A heisst symmetrisch, wenn jedes geordnete Paar (a_1, a_2) der Relation R auch mit vertauschten Elementen als (a_2, a_1) in der Relation R enthalten ist. Mathematisch ausgedrückt:

$$\text{für alle } a_1, a_2 \in A \text{ gilt: } (a_1, a_2) \in R \implies (a_2, a_1) \in R$$

2. *asymmetrisch*:
Eine binäre Relation R auf der Menge A heisst asymmetrisch, wenn jedes geordnete Paar (a_1, a_2) der Relation R mit vertauschten Elementen als (a_2, a_1) nicht in der Relation R enthalten ist. Mathematisch ausgedrückt:

$$\text{für alle } a_1, a_2 \in A \text{ gilt: } (a_1, a_2) \in R \implies \neg(a_2, a_1) \in R$$

3. *reflexiv*:
Eine binäre Relation R auf der Menge A heisst reflexiv, wenn jedes Element a der Menge A zu sich selbst in Relation steht. Mathematisch ausgedrückt:

$$\text{für alle } a \in A \text{ gilt: } (a, a) \in R$$

4. *irreflexiv*:
Eine binäre Relation R auf der Menge A heisst irreflexiv, wenn jedes Element a der Menge A nicht zu sich selbst in Relation steht. Mathematisch ausgedrückt:

$$\text{für alle } a \in A \text{ gilt: } \neg(a, a) \in R$$

5. *transitiv*:
Eine binäre Relation R auf der Menge A heisst transitiv, wenn für je drei Elemente a_1, a_2, a_3 der Menge A gilt: Sind die Paare (a_1, a_2) und (a_2, a_3) in der Relation R, dann ist auch das Paar (a_1, a_3) in der Relation R. Mathematisch ausgedrückt:

$$\text{für alle } a_1, a_2, a_3 \in A \text{ gilt: } (a_1, a_2) \in R \wedge (a_2, a_3) \in R \implies (a_1, a_3) \in R.$$

Ein Beispiel für eine reflexive und transitive Relation ist die Relation $\prec$ auf der Menge aller Zerlegungen einer Menge A. Für je drei Zerlegungen P_1, P_2, P_3 gilt:
$(P_1 \prec P_2) \wedge (P_2 \prec P_3) \implies (P_1 \prec P_3)$.

Mit Hilfe von Mengen und Relationen wollen wir eine bestimmte häufig auftretende Relation, die Äquivalenzrelation *gleich* definieren.

Definition 4.15 *Äquivalenzrelation gleich (=)*
Eine binäre Relation R, welche die Eigenschaften symmetrisch, reflexiv und transitiv aufweist, heisst Äquivalenzrelation.

Die Äquivalenzrelation (=) ist uns sicherlich schon an vielen Stellen begegnet. Berechnen wir beispielsweise das Ergebnis einer arithmetischen Operation, dann treten wir die arithmetischen Operation vom Ergebnis dazu durch ein Gleichheitszeichen (Beispiel $A + B = C$). Die Operation $A + B$ steht in Relation zu C, dabei handelt es sich um eine Äquivalenzrelation.

Die Äquivalenzrelation zerlegt eine zugrundeliegende Menge A in natürlicher Weise. Über die Äquivalenzrelation wird die sog. Äquivalenzklasse $[x]$ definiert, welche die zu einem bestimmten Element x der Menge A äquivalenten Elemente enthält. Wir wollen diese Äquivalenzklasse $[x]$ in einer Definition festlegen:

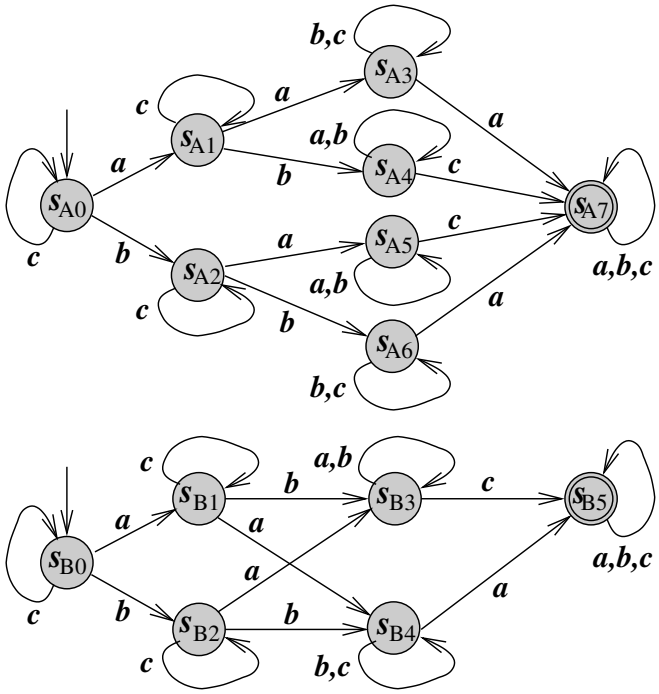
Definition 4.16 *Äquivalenzklasse* $[x]$
Es sei $\sim$ *eine Äquivalenzrelation auf der Menge A. Dann ist die Äquivalenzklasse* $[x]$ *von x gegeben durch:*

$$[x] = \{x' \in A \mid x' \sim x\}$$

Ferner gilt, dass die Äquivalenzklasse $[y]$ von y gleich der Äquivalenzklasse $[x]$ von x ist ($[y] = [x]$), falls $x \sim y$ gilt. $P = \{[x] \mid x \in A\}$ ist eine Zerlegung von A.

4.8 Anhang II: Ergänzendes Beispiel zur Äquivalenz

Beispiel 4.50 *Äquivalente Automaten (Anhang)*
Gegeben seien die folgenden beiden endlichen Automaten EA_A und EA_B mit dem Eingabealphabet $E_A = E_B = \{a,b,c\}$, den Endzuständen $F_A = \{s_{A7}\}$, $F_B = \{s_{B5}\}$ und den folgenden Übergangsfunktionen:



a) Welche Zustände innerhalb des Automaten EA_A sind k-äquivalent und welche sind äquivalent?

b) Welche Zustände von EA_A sind k-äquivalent zu welchen Zuständen in EA_B?

c) Welche Zustände von EA_A sind äquivalent zu welchen Zuständen in EA_B?

a) k-äquivalente Zustände innerhalb EA_A
Wir beginnen mit der trivialen 0-Äquivalenz $\overset{0}{\sim}$. Da nur s_{A7} ein Endzustand ist, sind bis auf den Zustand s_{A7} alle anderen Zustände s_{A0}, s_{A1}, s_{A2}, s_{A3}, s_{A4}, s_{A5} und s_{A6} paarweise zueinander 0-äquivalent.

Für die Bestimmung der 1-Äquivalenz $\overset{1}{\sim}$ müssen nun für alle möglichen Zustandspaare (s_i, s_j) deren Folgezustände $s_i^1 = \delta(s_i, w)$ und $s_j^1 = \delta(s_j, w)$ für alle $w \in E^*$ mit $|w| = 1$ untereinander bzgl. der Eigenschaft *Endzustand* $s \in F$ oder *kein Endzustand* verglichen werden. Dazu stellen wir uns alle Folgezustände tabellarisch dar:

$\delta(s_{A0},a) = s_{A1}$	$\delta(s_{A0},b) = s_{A2}$	$\delta(s_{A0},c) = s_{A0}$
$\delta(s_{A1},a) = s_{A3}$	$\delta(s_{A1},b) = s_{A4}$	$\delta(s_{A1},c) = s_{A1}$
$\delta(s_{A2},a) = s_{A5}$	$\delta(s_{A2},b) = s_{A6}$	$\delta(s_{A2},c) = s_{A2}$
$\delta(s_{A3},a) = s_{A7} \in F$	$\delta(s_{A3},b) = s_{A3}$	$\delta(s_{A3},c) = s_{A3}$
$\delta(s_{A4},a) = s_{A4}$	$\delta(s_{A4},b) = s_{A4}$	$\delta(s_{A4},c) = s_{A7} \in F$
$\delta(s_{A5},a) = s_{A5}$	$\delta(s_{A5},b) = s_{A5}$	$\delta(s_{A5},c) = s_{A7} \in F$
$\delta(s_{A6},a) = s_{A7} \in F$	$\delta(s_{A6},b) = s_{A6}$	$\delta(s_{A6},c) = s_{A6}$
$\delta(s_{A7},a) = s_{A7} \in F$	$\delta(s_{A7},b) = s_{A7} \in F$	$\delta(s_{A7},c) = s_{A7} \in F$

1-äquivalent sind die Zustände, die bei den drei Transitionen $\delta(s_i, w)$ mit $w = a$, $w = b$ und $w = c$ das gleiche Folgezustandsmuster mit *Endzustand* $s \in F$ oder *kein Endzustand* aufweisen und 0-äquivalent sind. Dies ist bei den Zuständen s_{A0}, s_{A1} und s_{A2} der Fall, die Zustände s_{A0}, s_{A1} und s_{A2} sind 1-äquivalent. Ebenso sind die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} 1-äquivalent.

Bei der 2-Äquivalenz $\overset{2}{\sim}$ müssen alle möglichen Zustandspaare (s_i, s_j) hinsichtlich $s_i^2 = \delta(s_i, w)$ und $s_j^2 = \delta(s_j, w)$ für alle $w \in E^*$ mit $|w| = 2$ untereinander bzgl. der Eigenschaft *Endzustand* oder *kein Endzustand* verglichen werden.

$\delta(s_{A0},aa)=s_{A3}$	$\delta(s_{A0},ab)=s_{A4}$	$\delta(s_{A0},ac)=s_{A1}$
$\delta(s_{A0},ba)=s_{A5}$	$\delta(s_{A0},bb)=s_{A6}$	$\delta(s_{A0},bc)=s_{A2}$
$\delta(s_{A0},ca)=s_{A1}$	$\delta(s_{A0},cb)=s_{A2}$	$\delta(s_{A0},cc)=s_{A0}$
$\delta(s_{A1},aa)=s_{A7}\in F$	$\delta(s_{A1},ab)=s_{A3}$	$\delta(s_{A1},ac)=s_{A3}$
$\delta(s_{A1},ba)=s_{A4}$	$\delta(s_{A1},bb)=s_{A4}$	$\delta(s_{A1},bc)=s_{A7}\in F$
$\delta(s_{A1},ca)=s_{A3}$	$\delta(s_{A1},cb)=s_{A4}$	$\delta(s_{A1},cc)=s_{A1}$
$\delta(s_{A2},aa)=s_{A5}$	$\delta(s_{A2},ab)=s_{A5}$	$\delta(s_{A2},ac)=s_{A7}\in F$
$\delta(s_{A2},ba)=s_{A7}\in F$	$\delta(s_{A2},bb)=s_{A6}$	$\delta(s_{A2},bc)=s_{A6}$
$\delta(s_{A2},ca)=s_{A5}$	$\delta(s_{A2},cb)=s_{A6}$	$\delta(s_{A2},cc)=s_{A2}$
$\delta(s_{A3},aa)=s_{A7}\in F$	$\delta(s_{A3},ab)=s_{A7}\in F$	$\delta(s_{A3},ac)=s_{A7}\in F$
$\delta(s_{A3},ba)=s_{A7}\in F$	$\delta(s_{A3},bb)=s_{A3}$	$\delta(s_{A3},bc)=s_{A3}$
$\delta(s_{A3},ca)=s_{A7}\in F$	$\delta(s_{A3},cb)=s_{A3}$	$\delta(s_{A3},cc)=s_{A3}$
$\delta(s_{A4},aa)=s_{A4}$	$\delta(s_{A4},ab)=s_{A4}$	$\delta(s_{A4},ac)=s_{A7}\in F$
$\delta(s_{A4},ba)=s_{A4}$	$\delta(s_{A4},bb)=s_{A4}$	$\delta(s_{A4},bc)=s_{A7}\in F$
$\delta(s_{A4},ca)=s_{A7}\in F$	$\delta(s_{A4},cb)=s_{A7}\in F$	$\delta(s_{A4},cc)=s_{A7}\in F$
$\delta(s_{A5},aa)=s_{A5}$	$\delta(s_{A5},ab)=s_{A5}$	$\delta(s_{A5},ac)=s_{A7}\in F$
$\delta(s_{A5},ba)=s_{A5}$	$\delta(s_{A5},bb)=s_{A5}$	$\delta(s_{A5},bc)=s_{A7}\in F$
$\delta(s_{A5},ca)=s_{A7}\in F$	$\delta(s_{A5},cb)=s_{A7}\in F$	$\delta(s_{A5},cc)=s_{A7}\in F$
$\delta(s_{A6},aa)=s_{A7}\in F$	$\delta(s_{A6},ab)=s_{A7}\in F$	$\delta(s_{A6},ac)=s_{A7}\in F$
$\delta(s_{A6},ba)=s_{A7}\in F$	$\delta(s_{A6},bb)=s_{A6}$	$\delta(s_{A6},bc)=s_{A6}$
$\delta(s_{A6},ca)=s_{A7}\in F$	$\delta(s_{A6},cb)=s_{A6}$	$\delta(s_{A6},cc)=s_{A6}$
$\delta(s_{A7},aa)=s_{A7}\in F$	$\delta(s_{A7},ab)=s_{A7}\in F$	$\delta(s_{A7},ac)=s_{A7}\in F$
$\delta(s_{A7},ba)=s_{A7}\in F$	$\delta(s_{A7},bb)=s_{A7}\in F$	$\delta(s_{A7},bc)=s_{A7}\in F$
$\delta(s_{A7},ca)=s_{A7}\in F$	$\delta(s_{A7},cb)=s_{A7}\in F$	$\delta(s_{A7},cc)=s_{A7}\in F$

2-äquivalent sind die Zustände, die bei den acht Transitionen $\delta(s_i,w)$ mit $w=aa$, $w=ab$, $w=ac$, $w=ba$, $w=bb$, $w=bc$, $w=ca$, $w=cb$ und $w=cc$ das gleiche Folgezustandsmuster mit *Endzustand* oder *kein Endzustand* aufweisen und 1-äquivalent sind. Damit sind die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} 2-äquivalent. Dagegen sind die Zustände s_{A0}, s_{A1} und s_{A2} jedoch nicht 2-äquivalent.

Bei der 3-Äquivalenz $\overset{3}{\sim}$ müssen alle möglichen Zustandspaare (s_i,s_j) hinsichtlich $s_i^3=\delta(s_i,w)$ und $s_j^3=\delta(s_j,w)$ für alle $w\in E^*$ mit $|w|=3$ untereinander bzgl. der Eigenschaft *Endzustand* oder *kein Endzustand* verglichen werden.

$\delta(s_{A0},aaa)=s_{A7}\in F$	$\delta(s_{A0},aab)=s_{A3}$	$\delta(s_{A0},aac)=s_{A3}$
$\delta(s_{A0},aba)=s_{A4}$	$\delta(s_{A0},abb)=s_{A4}$	$\delta(s_{A0},abc)=s_{A7}\in F$
$\delta(s_{A0},aca)=s_{A3}$	$\delta(s_{A0},acb)=s_{A4}$	$\delta(s_{A0},acc)=s_{A1}$
$\delta(s_{A0},baa)=s_{A5}$	$\delta(s_{A0},bab)=s_{A5}$	$\delta(s_{A0},bac)=s_{A7}\in F$
$\delta(s_{A0},bba)=s_{A7}\in F$	$\delta(s_{A0},bbb)=s_{A6}$	$\delta(s_{A0},bbc)=s_{A6}$
$\delta(s_{A0},bca)=s_{A5}$	$\delta(s_{A0},bcb)=s_{A6}$	$\delta(s_{A0},bcc)=s_{A2}$
$\delta(s_{A0},caa)=s_{A3}$	$\delta(s_{A0},cab)=s_{A4}$	$\delta(s_{A0},cac)=s_{A1}$
$\delta(s_{A0},cba)=s_{A5}$	$\delta(s_{A0},cbb)=s_{A6}$	$\delta(s_{A0},cbc)=s_{A2}$
$\delta(s_{A0},cca)=s_{A1}$	$\delta(s_{A0},ccb)=s_{A2}$	$\delta(s_{A0},ccc)=s_{A0}$
$\delta(s_{A1},aaa)=s_{A7}\in F$	$\delta(s_{A1},aab)=s_{A7}\in F$	$\delta(s_{A1},aac)=s_{A7}\in F$
$\delta(s_{A1},aba)=s_{A7}\in F$	$\delta(s_{A1},abb)=s_{A3}$	$\delta(s_{A1},abc)=s_{A3}$
$\delta(s_{A1},aca)=s_{A7}\in F$	$\delta(s_{A1},acb)=s_{A3}$	$\delta(s_{A1},acc)=s_{A3}$
$\delta(s_{A1},baa)=s_{A4}$	$\delta(s_{A1},bab)=s_{A4}$	$\delta(s_{A1},bac)=s_{A7}\in F$
$\delta(s_{A1},bba)=s_{A4}$	$\delta(s_{A1},bbb)=s_{A4}$	$\delta(s_{A1},bbc)=s_{A7}\in F$
$\delta(s_{A1},bca)=s_{A7}\in F$	$\delta(s_{A1},bcb)=s_{A7}\in F$	$\delta(s_{A1},bcc)=s_{A7}\in F$
$\delta(s_{A1},caa)=s_{A7}\in F$	$\delta(s_{A1},cab)=s_{A3}$	$\delta(s_{A1},cac)=s_{A3}$
$\delta(s_{A1},cba)=s_{A4}$	$\delta(s_{A1},cbb)=s_{A4}$	$\delta(s_{A1},cbc)=s_{A7}\in F$
$\delta(s_{A1},cca)=s_{A3}$	$\delta(s_{A1},ccb)=s_{A4}$	$\delta(s_{A1},ccc)=s_{A1}$
$\delta(s_{A2},aaa)=s_{A5}$	$\delta(s_{A2},aab)=s_{A5}$	$\delta(s_{A2},aac)=s_{A7}\in F$
$\delta(s_{A2},aba)=s_{A5}$	$\delta(s_{A2},abb)=s_{A5}$	$\delta(s_{A2},abc)=s_{A7}\in F$
$\delta(s_{A2},aca)=s_{A7}\in F$	$\delta(s_{A2},acb)=s_{A7}\in F$	$\delta(s_{A2},acc)=s_{A7}\in F$
$\delta(s_{A2},baa)=s_{A7}\in F$	$\delta(s_{A2},bab)=s_{A7}\in F$	$\delta(s_{A2},bac)=s_{A7}\in F$
$\delta(s_{A2},bba)=s_{A7}\in F$	$\delta(s_{A2},bbb)=s_{A6}$	$\delta(s_{A2},bbc)=s_{A6}$

$\delta(s_{A2}, bca) = s_{A7} \in F$	$\delta(s_{A2}, bcb) = s_{A6}$	$\delta(s_{A2}, bcc) = s_{A6}$
$\delta(s_{A2}, caa) = s_{A5}$	$\delta(s_{A2}, cab) = s_{A5}$	$\delta(s_{A2}, cac) = s_{A7} \in F$
$\delta(s_{A2}, cba) = s_{A7} \in F$	$\delta(s_{A2}, cbb) = s_{A6}$	$\delta(s_{A2}, cbc) = s_{A6} \in F$
$\delta(s_{A2}, cca) = s_{A5}$	$\delta(s_{A2}, ccb) = s_{A6}$	$\delta(s_{A2}, ccc) = s_{A2}$
$\delta(s_{A3}, aaa) = s_{A7} \in F$	$\delta(s_{A3}, aab) = s_{A7} \in F$	$\delta(s_{A3}, aac) = s_{A7} \in F$
$\delta(s_{A3}, aba) = s_{A7} \in F$	$\delta(s_{A3}, abb) = s_{A7} \in F$	$\delta(s_{A3}, abc) = s_{A7} \in F$
$\delta(s_{A3}, aca) = s_{A7} \in F$	$\delta(s_{A3}, acb) = s_{A3}$	$\delta(s_{A3}, acc) = s_{A7} \in F$
$\delta(s_{A3}, baa) = s_{A7} \in F$	$\delta(s_{A3}, bab) = s_{A7} \in F$	$\delta(s_{A3}, bac) = s_{A7} \in F$
$\delta(s_{A3}, bba) = s_{A7} \in F$	$\delta(s_{A3}, bbb) = s_{A3}$	$\delta(s_{A3}, bbc) = s_{A3}$
$\delta(s_{A3}, bca) = s_{A7} \in F$	$\delta(s_{A3}, bcb) = s_{A3}$	$\delta(s_{A3}, bcc) = s_{A3}$
$\delta(s_{A3}, caa) = s_{A7} \in F$	$\delta(s_{A3}, cab) = s_{A7} \in F$	$\delta(s_{A3}, cac) = s_{A7} \in F$
$\delta(s_{A3}, cba) = s_{A7} \in F$	$\delta(s_{A3}, cbb) = s_{A3}$	$\delta(s_{A3}, cbc) = s_{A3}$
$\delta(s_{A3}, cca) = s_{A7} \in F$	$\delta(s_{A3}, ccb) = s_{A3}$	$\delta(s_{A3}, ccc) = s_{A3}$
$\delta(s_{A4}, aaa) = s_{A4}$	$\delta(s_{A4}, aab) = s_{A4}$	$\delta(s_{A4}, aac) = s_{A7} \in F$
$\delta(s_{A4}, aba) = s_{A4}$	$\delta(s_{A4}, abb) = s_{A4}$	$\delta(s_{A4}, abc) = s_{A7} \in F$
$\delta(s_{A4}, aca) = s_{A7} \in F$	$\delta(s_{A4}, acb) = s_{A7} \in F$	$\delta(s_{A4}, acc) = s_{A7} \in F$
$\delta(s_{A4}, baa) = s_{A4}$	$\delta(s_{A4}, bab) = s_{A4}$	$\delta(s_{A4}, bac) = s_{A7} \in F$
$\delta(s_{A4}, bba) = s_{A4}$	$\delta(s_{A4}, bbb) = s_{A4}$	$\delta(s_{A4}, bbc) = s_{A7} \in F$
$\delta(s_{A4}, bca) = s_{A7} \in F$	$\delta(s_{A4}, bcb) = s_{A7} \in F$	$\delta(s_{A4}, bcc) = s_{A7} \in F$
$\delta(s_{A4}, caa) = s_{A7} \in F$	$\delta(s_{A4}, cab) = s_{A7} \in F$	$\delta(s_{A4}, cac) = s_{A7} \in F$
$\delta(s_{A4}, cba) = s_{A7} \in F$	$\delta(s_{A4}, cbb) = s_{A7} \in F$	$\delta(s_{A4}, cbc) = s_{A7} \in F$
$\delta(s_{A4}, cca) = s_{A7} \in F$	$\delta(s_{A4}, ccb) = s_{A7} \in F$	$\delta(s_{A4}, ccc) = s_{A7} \in F$
$\delta(s_{A5}, aaa) = s_{A5}$	$\delta(s_{A5}, aab) = s_{A5}$	$\delta(s_{A5}, aac) = s_{A7} \in F$
$\delta(s_{A5}, aba) = s_{A5}$	$\delta(s_{A5}, abb) = s_{A5}$	$\delta(s_{A5}, abc) = s_{A7} \in F$
$\delta(s_{A5}, aca) = s_{A7} \in F$	$\delta(s_{A5}, acb) = s_{A7} \in F$	$\delta(s_{A5}, acc) = s_{A7} \in F$
$\delta(s_{A5}, baa) = s_{A5}$	$\delta(s_{A5}, bab) = s_{A5}$	$\delta(s_{A5}, bac) = s_{A7} \in F$
$\delta(s_{A5}, bba) = s_{A5}$	$\delta(s_{A5}, bbb) = s_{A5}$	$\delta(s_{A5}, bbc) = s_{A7} \in F$
$\delta(s_{A5}, bca) = s_{A7} \in F$	$\delta(s_{A5}, bcb) = s_{A7} \in F$	$\delta(s_{A5}, bcc) = s_{A7} \in F$
$\delta(s_{A5}, caa) = s_{A7} \in F$	$\delta(s_{A5}, cab) = s_{A7} \in F$	$\delta(s_{A5}, cac) = s_{A7} \in F$
$\delta(s_{A5}, cba) = s_{A7} \in F$	$\delta(s_{A5}, cbb) = s_{A7} \in F$	$\delta(s_{A5}, cbc) = s_{A7} \in F$
$\delta(s_{A5}, cca) = s_{A7} \in F$	$\delta(s_{A5}, ccb) = s_{A7} \in F$	$\delta(s_{A5}, ccc) = s_{A7} \in F$
$\delta(s_{A6}, aaa) = s_{A7} \in F$	$\delta(s_{A6}, aab) = s_{A7} \in F$	$\delta(s_{A6}, aac) = s_{A7} \in F$
$\delta(s_{A6}, aba) = s_{A7} \in F$	$\delta(s_{A6}, abb) = s_{A7} \in F$	$\delta(s_{A6}, abc) = s_{A7} \in F$
$\delta(s_{A6}, aca) = s_{A7} \in F$	$\delta(s_{A6}, acb) = s_{A7} \in F$	$\delta(s_{A6}, acc) = s_{A7} \in F$
$\delta(s_{A6}, baa) = s_{A7} \in F$	$\delta(s_{A6}, bab) = s_{A7} \in F$	$\delta(s_{A6}, bac) = s_{A7} \in F$
$\delta(s_{A6}, bba) = s_{A7} \in F$	$\delta(s_{A6}, bbb) = s_{A6}$	$\delta(s_{A6}, bbc) = s_{A6}$
$\delta(s_{A6}, bca) = s_{A7} \in F$	$\delta(s_{A6}, bcb) = s_{A6}$	$\delta(s_{A6}, bcc) = s_{A6}$
$\delta(s_{A6}, caa) = s_{A7} \in F$	$\delta(s_{A6}, cab) = s_{A7} \in F$	$\delta(s_{A6}, cac) = s_{A7} \in F$
$\delta(s_{A6}, cba) = s_{A7} \in F$	$\delta(s_{A6}, cbb) = s_{A6}$	$\delta(s_{A6}, cbc) = s_{A6}$
$\delta(s_{A6}, cca) = s_{A7} \in F$	$\delta(s_{A6}, ccb) = s_{A6}$	$\delta(s_{A6}, ccc) = s_{A6}$
$\delta(s_{A7}, aaa) = s_{A7} \in F$	$\delta(s_{A7}, aab) = s_{A7} \in F$	$\delta(s_{A7}, aac) = s_{A7} \in F$
$\delta(s_{A7}, aba) = s_{A7} \in F$	$\delta(s_{A7}, abb) = s_{A7} \in F$	$\delta(s_{A7}, abc) = s_{A7} \in F$
$\delta(s_{A7}, aca) = s_{A7} \in F$	$\delta(s_{A7}, acb) = s_{A7} \in F$	$\delta(s_{A7}, acc) = s_{A7} \in F$
$\delta(s_{A7}, baa) = s_{A7} \in F$	$\delta(s_{A7}, bab) = s_{A7} \in F$	$\delta(s_{A7}, bac) = s_{A7} \in F$
$\delta(s_{A7}, bba) = s_{A7} \in F$	$\delta(s_{A7}, bbb) = s_{A7} \in F$	$\delta(s_{A7}, bbc) = s_{A7} \in F$
$\delta(s_{A7}, bca) = s_{A7} \in F$	$\delta(s_{A7}, bcb) = s_{A7} \in F$	$\delta(s_{A7}, bcc) = s_{A7} \in F$
$\delta(s_{A7}, caa) = s_{A7} \in F$	$\delta(s_{A7}, cab) = s_{A7} \in F$	$\delta(s_{A7}, cac) = s_{A7} \in F$
$\delta(s_{A7}, cba) = s_{A7} \in F$	$\delta(s_{A7}, cbb) = s_{A7} \in F$	$\delta(s_{A7}, cbc) = s_{A7} \in F$
$\delta(s_{A7}, cca) = s_{A7} \in F$	$\delta(s_{A7}, ccb) = s_{A7} \in F$	$\delta(s_{A7}, ccc) = s_{A7} \in F$

3-äquivalent sind die Zustände, die bei den 27 Transitionen $\delta(s_i, w)$ das gleiche Folgezustandsmuster mit *Endzustand* oder *kein Endzustand* aufweisen und 2-äquivalent sind. Damit sind die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} 3-äquivalent.

Wenn wir nun noch die 4-Äquivalenz betrachten würden, dann würden sich dabei das gleiche Ergebnis wie für die 3-Äquivalenz ergeben. Diesen Aufwand mit je 81 Transitionen pro Zustand wollen wir hier nicht mehr vorführen. Die gleiche Argumentation lässt sich auf alle weiteren k-Äquivalenzen übertragen.

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A1}		$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A2}			$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A3}				$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A4}					$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A5}						$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{A6}							$\overset{0}{\sim}$	
s_{A7}								$\overset{0}{\sim}$

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{A1}		$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{A2}			$\overset{1}{\sim}$					
s_{A3}				$\overset{1}{\sim}$			$\overset{1}{\sim}$	
s_{A4}					$\overset{1}{\sim}$	$\overset{1}{\sim}$		
s_{A5}						$\overset{1}{\sim}$		
s_{A6}							$\overset{1}{\sim}$	
s_{A7}								$\overset{1}{\sim}$

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{2}{\sim}$							
s_{A1}		$\overset{2}{\sim}$						
s_{A2}			$\overset{2}{\sim}$					
s_{A3}				$\overset{2}{\sim}$			$\overset{2}{\sim}$	
s_{A4}					$\overset{2}{\sim}$	$\overset{2}{\sim}$		
s_{A5}						$\overset{2}{\sim}$		
s_{A6}							$\overset{2}{\sim}$	
s_{A7}								$\overset{2}{\sim}$

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{A0}	$\overset{3}{\sim}$							
s_{A1}		$\overset{3}{\sim}$						
s_{A2}			$\overset{3}{\sim}$					
s_{A3}				$\overset{3}{\sim}$			$\overset{3}{\sim}$	
s_{A4}					$\overset{3}{\sim}$	$\overset{3}{\sim}$		
s_{A5}						$\overset{3}{\sim}$		
s_{A6}							$\overset{3}{\sim}$	
s_{A7}								$\overset{3}{\sim}$

Bleibt noch die Frage welche Zustände *äquivalent* sind bzgl. der Definition 4.11. Diese besagt, dass zwei Zustände $s_i, s_j \in S$ äquivalent ($s_i \sim s_j$) zueinander sind wenn für alle $k \in \mathbb{N}_0$ gilt: $s_i \overset{k}{\sim} s_j$. Da wir mit der obigen Argumentation für alle $k \geq 3$ die gleiche Äquivalenzrelation (Tabelle der Äquivalenzen) erhalten würden wie für $k=3$, folgt daraus: Für alle Zustandspaare, für die $s_i \overset{3}{\sim} s_j$ gilt, für die gilt auch $s_i \sim s_j$. Damit sind in EA_A die Zustände s_{A3} und s_{A6} sowie die Zustände s_{A4} und s_{A5} äquivalent.

b) k-äquivalente Zustände von EA_A und EA_B
Da nur die Zustände s_{A7} in EA_A und s_{B5} in EA_B Endzustände sind und alle anderen nicht, sind diese zueinander 0-äquivalent. Andererseits sind alle Nicht-Endzustände der beiden Automaten paarweise zueinander 0-äquivalent.

Für die Bestimmung der 1-Äquivalenz $\overset{1}{\sim}$ müssen für alle möglichen Zustandspaare (s_i, s_j) deren Folgezustände $s^1_{Ai} = \delta_A(s_{Ai}, w)$ und $s^1_{Bi} = \delta_B(s_{Bi}, w)$ für alle $w \in E^*$ mit $|w| = 1$ untereinander bzgl. der Eigenschaft *Endzustand* $s \in F$ oder *kein Endzustand* verglichen werden.

Entsprechend müssen dann weiter alle Äquivalenztabellen für $\overset{k}{\sim}$ mit $k = 1, 2, ...$ ermittelt werden. Dies soll hier nicht mehr vorgeführt werden. Wir geben daher nur noch das Ergebnis an:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B1}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B2}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B3}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B4}	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	$\overset{0}{\sim}$	
s_{B5}								$\overset{0}{\sim}$

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{B1}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{B2}	$\overset{1}{\sim}$	$\overset{1}{\sim}$	$\overset{1}{\sim}$					
s_{B3}					$\overset{1}{\sim}$	$\overset{1}{\sim}$		$\overset{1}{\sim}$
s_{B4}				$\overset{1}{\sim}$			$\overset{1}{\sim}$	
s_{B5}					$\overset{1}{\sim}$	$\overset{1}{\sim}$		$\overset{1}{\sim}$

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{2}{\sim}$							
s_{B1}		$\overset{2}{\sim}$						
s_{B2}			$\overset{2}{\sim}$					
s_{B3}				$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$
s_{B4}				$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$
s_{B5}				$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$	$\overset{2}{\sim}$

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\overset{3}{\sim}$							
s_{B1}		$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$
s_{B2}		$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$
s_{B3}		$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$
s_{B4}		$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$
s_{B5}		$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$	$\overset{3}{\sim}$

Für die 4- und alle weiteren k-Äquivalenz ergibt sich das gleiche Ergebnis wie für die 3-Äquivalenz.

c) Äquivalente Zustände von EA_A und EA_B
Laut der Definition 4.11 sind zwei Zustände $s_i \in S_1$ und $s_j \in S_2$ äquivalent $s_i \sim s_j$ zueinander wenn für alle $k \in \mathbb{N}_0$ gilt: $s_i \overset{k}{\sim} s_j$. Aus den Tabellen für die k-Äquivalenzen folgt:

	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}
s_{B0}	$\sim$							
s_{B1}		$\sim$						
s_{B2}			$\sim$					
s_{B3}					$\sim$	$\sim$		$\sim$
s_{B4}				$\sim$			$\sim$	
s_{B5}					$\sim$	$\sim$		$\sim$

Die folgenden Zustände sind also äquivalent:

$$
\begin{aligned}
&s_{A0} \sim s_{B0}\\
&s_{A1} \sim s_{B1}\\
&s_{A2} \sim s_{B2}\\
&s_{A3} \sim s_{A6} \sim s_{B4}\\
&s_{A4} \sim s_{A5} \sim s_{B3}\\
&s_{A7} \sim s_{B5}
\end{aligned}
$$

□

4.9 Lösungen zu den Selbsttestaufgaben

Aufgabe 4.1

Eingabeaktionen für einen erfolgreichen Verkaufsvorgang
Ein Verkaufsvorgang wird erfolgreich abgeschlossen wenn der Betrag S in Form von Folgen von Geldstücken eingegeben wurde. Mögliche erfolgreiche Folgen von Eingabeaktionen sind also:

- Ein Geldstück G1
- Ein Geldstück G2 gefolgt vom einem weiteren Geldstück G2
- Ein Geldstück G2 gefolgt vom einem Geldstück G3 und einem weiteren Geldstück G3
- Aufeinanderfolgender Einwurf von vier Geldstücken G3.

Aufgabe 4.2

Zeichenreihen der Länge $k=3$ über dem Alphabet $E = \{a,b,x\}$.

$$\begin{aligned} E^0 &= \{\varepsilon\} \qquad E^1 = \{a,b,x\} \\ E^2 &= \{aa,ab,ax,ba,bb,bx,xa,xb,xx\} \\ E^3 &= \{aaa,aab,aax,aba,abb,abx,axa,axb,axx,baa,bab,bax,bba,bbb,bbx,bxa,bxb,bxx, \\ &\quad xaa,xab,xax,xba,xbb,xbx,xxa,xxb,xxx\} \end{aligned}$$

Aufgabe 4.3

Wir betrachten zuerst den Zustand 3. Mit der Eingabe G1,A befindet sich der Betrag 6G im Automat und da die 6G für das gewählte Getränk ausreichend sind, kommen wir mit der Ausgabe von A zurück nach Zustand 0. Mit der Eingabe G2,A befindet sich der Betrag 5G im Automat und da 5G für das gewählte Getränk nicht ausreichend kommen wir in einen neuen Zustand 7. Interessanter wird hingegen schon die Eingabe G1,B. Da sich im Zustand 3 bereits 4G im betrachteten Verkaufsprozess befinden, ist das gewählte Getränk bereits bezahlt.
Da das Verhalten des Getränkeautomaten nicht genauer festlegt ist, nehmen wir der Einfachheit halber für die Modellierung an, dass der Automat auf die Veränderung des Wahlschalters nur nach der nächsten Eingabe reagiert. Mit dem Einwurf von G1 befinden sich 6G im Verkaufsprozess, das Getränk B wird ausgegeben und gleichzeitig auch 2G Wechselgeld wodurch der Verkaufsvorgang mit dem Übergang nach Zustand 0 abgeschlossen ist.

Betrachten wir den Zustand 4, dann kommen wir mit der Transition G1,A/- in den neuen Zustand 8 und mit G2,A/- in den neuen Zustand 9. Mit den Transitionen G1,B/B,1W und G2,B/B kommen wir in den Ausgangszustand 0 zurück.

Aufgabe 4.4

Die folgende Tabelle zeigt die Verarbeitung der einzelnen Wörter.

	Eingabewort	erreichter Zustand
(1)	$G1;A\ G1;A\ G1;A$	$\delta(s_0, G1;A\ G1;A\ G1;A) = s_0$
(2)	$G1;B\ G1;B\ G1;B$	$\delta(s_0, G1;B\ G1;B\ G1;B) = s_1$
(3)	$G2;B\ G2;B\ G1;B$	$\delta(s_0, G2;B\ G2;B\ G1;B) = s_0$
(4)	$G2;B\ G2;A\ G2;A\ G1;B$	$\delta(s_0, G2;B\ G2;A\ G2;A\ G1;B) = s_0$
(5)	$G1;B\ G1;A\ G2;A\ G2;A$	$\delta(s_0, G1;B\ G1;A\ G2;A\ G2;A) = s_0$
(6)	$G2;A\ G2;B\ G2;B\ G1;A\ G1;A$	$\delta(s_0, G2;A\ G2;B\ G2;B\ G1;A\ G1;A) = s_0$
(7)	$G2;A\ G2;B\ G2;A\ G2;A\ G2;A$	$\delta(s_0, G2;A\ G2;B\ G2;A\ G2;A\ G2;A) = s_{19}$
(8)	$G2;B\ G2;B\ G1;A\ G2;B\ G2;B$	$\delta(s_0, G2;B\ G2;B\ G1;A\ G2;B\ G2;B) = s_2$
(9)	$G1;A\ G2;A\ G2;A\ G2;A\ G2;A$	$\delta(s_0, G1;A\ G2;A\ G2;A\ G2;A\ G2;A) = s_0$

Aufgabe 4.5

Mit den Eingabeworten $E^2 = \{\varepsilon, a, b, aa, bb, ab, ba\}$ bilden wir die möglichen Konfigurationen $K_2 = S \times E^2$:

$$\begin{array}{ccccccc} (s_0,\varepsilon) & (s_0,a) & (s_0,aa) & (s_0,ab) & (s_0,b) & (s_0,bb) & (s_0,ba) \\ (s_1,\varepsilon) & (s_1,a) & (s_1,aa) & (s_1,ab) & (s_1,b) & (s_1,bb) & (s_1,ba) \end{array}$$

Die Übergangsrelation $\models_{EA}$ enthält alle möglichen Übergänge von einer Konfiguration (s, ew) zu einer Folgekonfiguration (s, w), die aufgrund der Übergangsfunktion δ möglich sind:

$$\begin{array}{lll} (s_0,a) \models_{EA} (s_1,\varepsilon) & (s_0,aa) \models_{EA} (s_1,a) & (s_0,ab) \models_{EA} (s_1,b) \\ (s_0,b) \models_{EA} (s_0,\varepsilon) & (s_0,bb) \models_{EA} (s_0,b) & (s_0,ba) \models_{EA} (s_0,a) \\ (s_1,a) \models_{EA} (s_1,\varepsilon) & (s_1,aa) \models_{EA} (s_1,a) & (s_1,ab) \models_{EA} (s_1,b) \\ (s_1,b) \models_{EA} (s_0,\varepsilon) & (s_1,bb) \models_{EA} (s_0,b) & (s_1,ba) \models_{EA} (s_0,a) \end{array}$$

Dargestellt als Relation $R_{\models} \subseteq ((S \times E^*) \times (S \times E^*))$ erhalten wir somit:

$$\begin{array}{rcl} R_{\models} & = & \{((s_0,a),(s_1,\varepsilon)), ((s_0,aa),(s_1,a)), ((s_0,ab)(s_1,b)), \\ & & ((s_0,b),(s_0,\varepsilon)), ((s_0,bb),(s_0,b)), ((s_0,ba),(s_0,a)), \\ & & ((s_1,a),(s_1,\varepsilon)), ((s_1,aa),(s_1,a)), ((s_1,ab),(s_1,b)), \\ & & ((s_1,b),(s_0,\varepsilon)), ((s_1,bb),(s_0,b)), ((s_1,ba),(s_0,a))\} \end{array}$$

Aufgabe 4.6

Die Sprache $L(EA)$ dieses Automaten ist gegeben durch die Zeichenreihen, die diesen Automaten vom Anfangszustand s_0 in den Endzustand s_2 überführen:

	Eingabewort	erreichter Zustand
(1)	a	$\delta(s_0,a) = s_1$
(2)	b	$\delta(s_0,b) = s_2$
(3)	aa	$\delta(s_0,aa) = s_2$
(4)	bb	$\delta(s_0,bb) = s_2$
(5)	ab	$\delta(s_0,ab) = s_0$
(6)	ba	$\delta(s_0,ba) = s_0$
(7)	aaa	$\delta(s_0,aaa) = s_0$
(8)	bbb	$\delta(s_0,bbb) = s_2$
(9)	aab	$\delta(s_0,aab) = s_2$
(10)	bba	$\delta(s_0,bba) = s_0$
(11)	abb	$\delta(s_0,abb) = s_2$
(12)	baa	$\delta(s_0,baa) = s_0$
(13)	aba	$\delta(s_0,aba) = s_1$
(14)	bab	$\delta(s_0,baa) = s_2$
(15)	$aaaa$	$\delta(s_0,aaaa) = s_1$
(16)	$bbbb$	$\delta(s_0,bbbb) = s_2$
...	...	...

Die Sprache ergibt sich demnach zu $L(EA) = \{b, aa, bb, bbb, aab, abb, bab, bbbb, ...\}$.

Aufgabe 4.7

Iteration L^* der Sprache $L = \{0, 11\}$.

$$\begin{array}{rcl} L^0 & = & \{\varepsilon\} \qquad L^1 = \{0, 11\} \\ L^2 & = & \{00, 011, 110, 1111\} \\ L^3 & = & \{000, 0011, 0110, 01111, 1100, 11011, 11110, 111111\} \\ L^4 & = & \{0000, 00011, 00110, 001111, 01100, 011011, 011110, 0111111, 11000, 110011, \\ & & 110110, 1101111, 111100, 1111011, 1111110, 11111111\} \\ L^5 & = & \{00000, 000011, 000110, 0001111, ...\} \\ & & \ldots \\ L^* & = & L^0 \cup L^1 \cup L^2 \cup L^3 \cup L^4 \cup ... \end{array}$$

Aufgabe 4.8

Sprache $L(RA)$ von Ausdruck $A_1 = (\boldsymbol{ab})^* + (\boldsymbol{ab})^*$

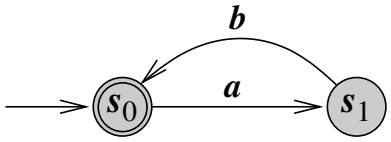
$$\begin{aligned} L[A_1] &= L[(\boldsymbol{ab})^* + (\boldsymbol{ab})^*] = L[(\boldsymbol{ab})^*] \cup L[\boldsymbol{ab})^*] = (L[\boldsymbol{ab}])^* \cup (L[\boldsymbol{ab}])^* \\ &= (L[\boldsymbol{a}]L[\boldsymbol{b}])^* \cup (L[\boldsymbol{b}]L[\boldsymbol{b}])^* \\ &= (\{a\}\{b\})^* \cup (\{a\}\{b\})^* = (\{ab\})^* \cup (\{ab\})^* = (\{ab\})^* \\ &= \{\varepsilon, ab, abab, ababab, abababab, ...\} \end{aligned}$$

Sprache $L(RA)$ von Ausdruck $A_2 = (\boldsymbol{a}+\varepsilon)(\boldsymbol{ab})^*$

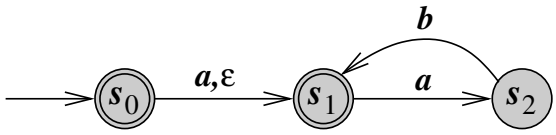
$$\begin{aligned} L[A_2] &= L[(\boldsymbol{a}+\varepsilon)(\boldsymbol{ab})^*] = L[(\boldsymbol{a}+\varepsilon)]L[(\boldsymbol{ab})^*] = (L[\boldsymbol{a}] \cup L[\varepsilon])(L[\boldsymbol{ab}])^* \\ &= (\{a\} \cup \{\varepsilon\})\{ab\}^* = \{\varepsilon, a\}\{\varepsilon, ab, abab, ababab, ...\} \\ &= \{\varepsilon, a, ab, aab, abab, aabab, ababab, aababab, ...\} \end{aligned}$$

Aufgabe 4.9

Automat EA zum Ausdruck $A_1 = (\boldsymbol{ab})^* + (\boldsymbol{ab})^*$, $L(A_1) = \{\varepsilon, ab, abab, ababab, abababab, ...\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 ausschließlich ein a. Nach einem a kann er ausschließlich nur ein b verarbeiten. Danach wieder nur ein a und wieder ein b. Das Verhalten ist äquivalent zum EA in Ausdruck 2) von Beispiel 4.30.

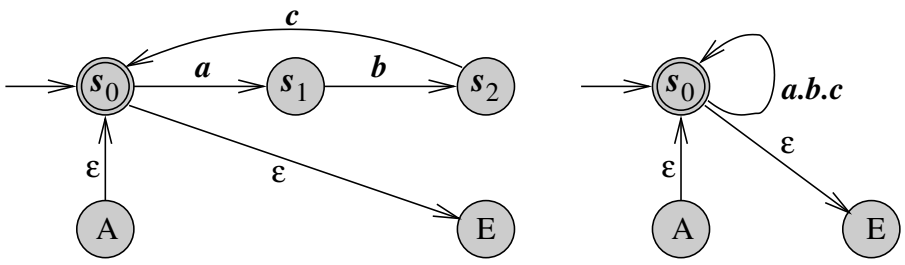


Automat EA zum Ausdruck $A_2 = (\boldsymbol{a}+\varepsilon)(\boldsymbol{ab})^*$, $L(A_2) = \{\varepsilon, a, ab, aab, abab, aabab, ababab, aababab, ...\}$
Der Automat zu diesem Ausdruck akzeptiert im Anfangszustand s_0 ein ε oder ein a. Nach dem ε als auch nach dem a kann er wiederholt immer wieder ab verarbeiten.

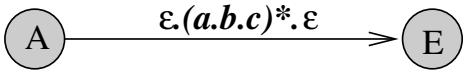


Aufgabe 4.10

Automat (1): Wir führen zuerst die Initialisierung durch.
Mit der Knotenelimination können in einem Schritt die beiden Knoten s_1 und s_2 eliminiert werden. Dies resultiert in einer Schlinge $a.b.c$ am Zustand s_0.

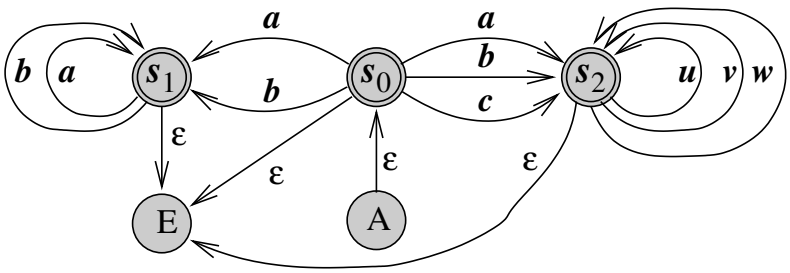


Entfernung der Schlinge resultiert in einer Konkatenation von $(a.b.c)^*$ mit der Transition ε von A nach s_0 und mit der Transition ε von s_0 nach E. Mit der Knotenelimination kann der Zustand s_0 entfernt werden, so dass folgt:

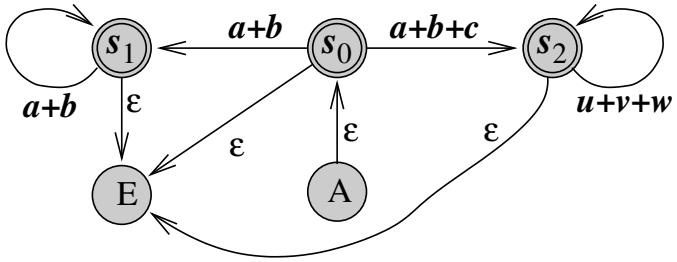


$$\mathrm{RA}_1 = \varepsilon.(\boldsymbol{a}.\boldsymbol{b}.\boldsymbol{c})^*.\varepsilon = (\boldsymbol{a}.\boldsymbol{b}.\boldsymbol{c})^*$$

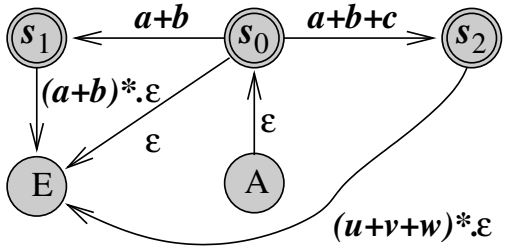
Automat (2): Wir führen zuerst die Initialisierung durch.



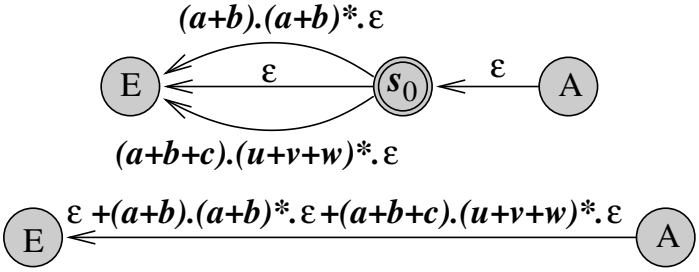
Mit der Kantenelimination werden die Schlingen an den Zuständen s_1 und s_2 zusammengelegt. Ferner werden die Kanten von s_0 nach s_1 und von s_0 nach s_2 zusammengelegt.



Dann werden mit einer Schleifenelimination die Schlingen an den Zuständen s_1 und s_2 entfernt. Dies hat zur Folge, dass das ε an der Transition von s_1 nach E mit $(a+b)^*$ konkateniert wird. Entsprechend wird ε an der Transition von s_2 nach E mit $(u+v+w)^*$ konkateniert.



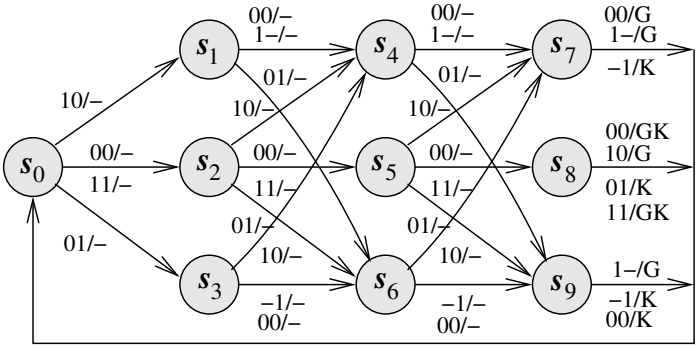
Als nächstes können die Zustände s_1 und s_2 eliminiert werden. Schließlich kann der Zustand s_0 elimiert werden und es folgt:



$$\begin{aligned} \mathrm{RA}_2 &= \varepsilon + (a+b).(a+b)^*.\varepsilon + (a+b+c).(u+v+w)^*.\varepsilon \\ &= \varepsilon + (a+b)^* + (a+b+c).(u+v+w)^* \\ &= (a+b)^* + (a+b+c).(u+v+w)^* \end{aligned}$$

Aufgabe 4.11

Annahme: Niederwertigste Bitstelle läuft zuerst ein
Weitere Zustände kann es nicht geben, da alle möglichen Pfade spezifiziert wurden und jede Sequenz auch hier immer nur 4 Zustände enthalten darf. Es gibt aber zusätzliche Transitionen, da die Entscheidung darüber, ob $A > B$, $A < B$ oder $A = B$ gilt, mit jeder neu einlaufenden Bitstelle rückgängig gemacht werden kann.
Erst die höchstwertigste Bitstelle bringt die endgültige Entscheidung. So gibt es Transitionen von Zustand 1 nach Zustand 6 und von Zustand 4 nach Zustand 9 sowie von Zustand 3 nach Zustand 4 und von Zustand 6 nach Zustand 7. Eine Transitionen von einer der beiden äußeren Sequenzen in die mittlere Sequenz ist nicht möglich, den wenn für je zwei Worte einmal $A > B$ oder $A < B$ gilt, dann können diese beide Worte nicht mehr gleich sein.
Dies führt auf das folgende Zustandsdiagramm:



Aufgabe 4.12

Das Verhalten des Getränkeautomaten wird neben dem Zustandsdiagramm auch durch die Zustandstabelle vollständig beschrieben. Der Übersichtlichkeit wegen bietet es sich daher an, die Reduzierung in der Zustandstabelle durchzuführen.

Zustand	**Folgezustand**			
S^n	S^{n+1}			
Eingabe	$G1;A$	$G1;B$	$G2;A$	$G2;B$
s_0	$s_1/-$	$s_1/-$	$s_2/-$	$s_2/-$
s_1	$s_3/-$	s_0/B	$s_4/-$	$s_4/-$
s_2	$s_5/-$	$s_5/-$	$s_6/-$	$s_6/-$
s_3	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_4	$s_8/-$	$s_0/B,1W$	$s_9/-$	s_0/B
s_5	$s_{10}/-$	$s_0/B,1W$	$s_{11}/-$	s_0/B
s_6	$s_{12}/-$	s_0/B	$s_{13}/-$	$s_{13}/-$
s_7	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_8	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_9	s_0/A	$s_0/B,2W$	$s_{14}/-$	$s_0/B,1W$
s_{10}	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_{11}	s_0/A	$s_0/B,2W$	$s_{15}/-$	$s_0/B,1W$
s_{12}	s_0/A	$s_0/B,2W$	$s_{16}/-$	$s_0/B,1W$
s_{13}	$s_{17}/-$	$s_0/B,1W$	$s_{18}/-$	s_0/B
s_{14}	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_{15}	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_{16}	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_{17}	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_{18}	s_0/A	$s_0/B,2W$	$s_{19}/-$	$s_0/B,1W$
s_{19}	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$

Zur Anwendung der 1. Minimierungsregel können wir jeden Zustand mit jedem anderen vergleichen und deren Äquivalenz prüfen. Dabei erkennen wir, dass die Zustände s_7, s_8, s_{10}, s_{14}, s_{15}, s_{16}, s_{17} und s_{19} äquivalent sind. Bei allen vier Eingaben $G1;A$, $G1;B$, $G2;A$ und $G2;B$ ergibt sich für alle diese Zustände jeweils der gleiche Folgezustand s_0 und die gleiche Ausgabe, für $G1;A$ folgt $A,2W$, für $G1;B$ folgt $B,3W$, für $G2;A$ folgt A und für $G2;B$ folgt $B,2W$. Symbolisch ausgedrückt:

$$
\begin{array}{cccc}
s_7 \xrightarrow{G1;A/A,2W} s_0 & s_7 \xrightarrow{G1;B/B,3W} s_0 & s_7 \xrightarrow{G2;A/A} s_0 & s_7 \xrightarrow{G2;B/B,2W} s_0 \\
s_8 \xrightarrow{G1;A/A,2W} s_0 & s_8 \xrightarrow{G1;B/B,3W} s_0 & s_8 \xrightarrow{G2;A/A} s_0 & s_8 \xrightarrow{G2;B/B,2W} s_0 \\
s_{10} \xrightarrow{G1;A/A,2W} s_0 & s_{10} \xrightarrow{G1;B/B,3W} s_0 & s_{10} \xrightarrow{G2;A/A} s_0 & s_{10} \xrightarrow{G2;B/B,2W} s_0 \\
s_{14} \xrightarrow{G1;A/A,2W} s_0 & s_{14} \xrightarrow{G1;B/B,3W} s_0 & s_{14} \xrightarrow{G2;A/A} s_0 & s_{14} \xrightarrow{G2;B/B,2W} s_0 \\
s_{15} \xrightarrow{G1;A/A,2W} s_0 & s_{14} \xrightarrow{G1;B/B,3W} s_0 & s_{15} \xrightarrow{G2;A/A} s_0 & s_{14} \xrightarrow{G2;B/B,2W} s_0 \\
s_{16} \xrightarrow{G1;A/A,2W} s_0 & s_{16} \xrightarrow{G1;B/B,3W} s_0 & s_{16} \xrightarrow{G2;A/A} s_0 & s_{16} \xrightarrow{G2;B/B,2W} s_0 \\
s_{17} \xrightarrow{G1;A/A,2W} s_0 & s_{17} \xrightarrow{G1;B/B,3W} s_0 & s_{17} \xrightarrow{G2;A/A} s_0 & s_{17} \xrightarrow{G2;B/B,2W} s_0 \\
s_{19} \xrightarrow{G1;A/A,2W} s_0 & s_{19} \xrightarrow{G1;B/B,3W} s_0 & s_{19} \xrightarrow{G2;A/A} s_0 & s_{19} \xrightarrow{G2;B/B,2W} s_0
\end{array}
$$

Folglich können die Zustände s_8, s_{10}, s_{14}, s_{15}, s_{16}, s_{17} und s_{19} eliminiert werden. Alle auf diesen sieben Zuständen endenden Transition müssen damit in Zustand s_7 enden. Es ergibt sich die folgende Zustandstabelle:

Zustand	**Folgezustand**			
S^n	S^{n+1}			
Eingabe	$G1;A$	$G1;B$	$G2;A$	$G2;B$
s_0	$s_1/-$	$s_1/-$	$s_2/-$	$s_2/-$
s_1	$s_3/-$	s_0/B	$s_4/-$	$s_4/-$
s_2	$s_5/-$	$s_5/-$	$s_6/-$	$s_6/-$
s_3	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_4	$s_7/-$	$s_0/B,1W$	$s_9/-$	s_0/B
s_5	$s_7/-$	$s_0/B,1W$	$s_{11}/-$	s_0/B
s_6	$s_{12}/-$	s_0/B	$s_{13}/-$	$s_{13}/-$
s_7	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_9	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_{11}	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_{12}	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_{13}	$s_7/-$	$s_0/B,1W$	$s_{18}/-$	s_0/B
s_{18}	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$

In dieser Zustandstabelle erkennen wir nun, dass durch die Zusammenlegung von Zuständen neue äquivalente Zustände entstanden sind. Es gilt:

$$
\begin{array}{cccc}
s_3 \xrightarrow{G1;A/A} s_0 & s_3 \xrightarrow{G1;B/B,2W} s_0 & s_3 \xrightarrow{G2;A/-} s_7 & s_3 \xrightarrow{G2;B/B,1W} s_0 \\
s_9 \xrightarrow{G1;A/A} s_0 & s_9 \xrightarrow{G1;B/B,2W} s_0 & s_9 \xrightarrow{G2;A/-} s_7 & s_9 \xrightarrow{G2;B/B,1W} s_0 \\
s_{11} \xrightarrow{G1;A/A} s_0 & s_{11} \xrightarrow{G1;B/B,2W} s_0 & s_{11} \xrightarrow{G2;A/-} s_7 & s_{11} \xrightarrow{G2;B/B,1W} s_0 \\
s_{12} \xrightarrow{G1;A/A} s_0 & s_{12} \xrightarrow{G1;B/B,2W} s_0 & s_{12} \xrightarrow{G2;A/-} s_7 & s_{12} \xrightarrow{G2;B/B,1W} s_0 \\
s_{18} \xrightarrow{G1;A/A} s_0 & s_{18} \xrightarrow{G1;B/B,2W} s_0 & s_{18} \xrightarrow{G2;A/-} s_7 & s_{18} \xrightarrow{G2;B/B,1W} s_0
\end{array}
$$

Folglich können die Zustände s_9, s_{11}, s_{12} und s_{18} eliminiert werden. Alle auf diesen vier Zuständen

endenden Transition müssen damit in Zustand s_3 enden:

Zustand	**Folgezustand**			
S^n	S^{n+1}			
Eingabe	$G1;A$	$G1;B$	$G2;A$	$G2;B$
s_0	$s_1/-$	$s_1/-$	$s_2/-$	$s_2/-$
s_1	$s_3/-$	s_0/B	$s_4/-$	$s_4/-$
s_2	$s_5/-$	$s_5/-$	$s_6/-$	$s_6/-$
s_3	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_4	$s_7/-$	$s_0/B,1W$	$s_3/-$	s_0/B
s_5	$s_7/-$	$s_0/B,1W$	$s_3/-$	s_0/B
s_6	$s_3/-$	s_0/B	$s_{13}/-$	$s_{13}/-$
s_7	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$
s_{13}	$s_7/-$	$s_0/B,1W$	$s_3/-$	s_0/B

Auch in dieser Zustandstabelle sind neue äquivalente Zustände entstanden:

$$s_4 \xrightarrow{G1;A/-} s_7 \qquad s_4 \xrightarrow{G1;B/B,1W} s_0 \qquad s_4 \xrightarrow{G2;A/-} s_3 \qquad s_4 \xrightarrow{G2;B/B} s_0$$
$$s_5 \xrightarrow{G1;A/-} s_7 \qquad s_5 \xrightarrow{G1;B/B,1W} s_0 \qquad s_5 \xrightarrow{G2;A/-} s_3 \qquad s_5 \xrightarrow{G2;B/B} s_0$$
$$s_{13} \xrightarrow{G1;A/-} s_7 \qquad s_{13} \xrightarrow{G1;B/B,1W} s_0 \qquad s_{13} \xrightarrow{G2;A/-} s_3 \qquad s_{13} \xrightarrow{G2;B/B} s_0$$

Folglich können die Zustände s_5 und s_{13} eliminiert werden. Alle auf diesen beiden Zuständen endenden Transition müssen damit in Zustand s_4 enden:

Zustand	**Folgezustand**			
S^n	S^{n+1}			
Eingabe	$G1;A$	$G1;B$	$G2;A$	$G2;B$
s_0	$s_1/-$	$s_1/-$	$s_2/-$	$s_2/-$
s_1	$s_3/-$	s_0/B	$s_4/-$	$s_4/-$
s_2	$s_4/-$	$s_4/-$	$s_6/-$	$s_6/-$
s_3	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_4	$s_7/-$	$s_0/B,1W$	$s_3/-$	s_0/B
s_6	$s_3/-$	s_0/B	$s_4/-$	$s_4/-$
s_7	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$

Auch in dieser Zustandstabelle gibt es noch zwei äquivalente Zustände:

$$s_1 \xrightarrow{G1;A/-} s_3 \qquad s_1 \xrightarrow{G1;B/B} s_0 \qquad s_1 \xrightarrow{G2;A/-} s_4 \qquad s_1 \xrightarrow{G2;B/-} s_4$$
$$s_6 \xrightarrow{G1;A/-} s_3 \qquad s_6 \xrightarrow{G1;B/B} s_0 \qquad s_6 \xrightarrow{G2;A/-} s_4 \qquad s_6 \xrightarrow{G2;B/-} s_4$$

Damit können wir den Zustand s_6 eliminieren.

Zustand	**Folgezustand**			
S^n	S^{n+1}			
Eingabe	$G1;A$	$G1;B$	$G2;A$	$G2;B$
s_0	$s_1/-$	$s_1/-$	$s_2/-$	$s_2/-$
s_1	$s_3/-$	s_0/B	$s_4/-$	$s_4/-$
s_2	$s_4/-$	$s_4/-$	$s_1/-$	$s_1/-$
s_3	s_0/A	$s_0/B,2W$	$s_7/-$	$s_0/B,1W$
s_4	$s_7/-$	$s_0/B,1W$	$s_3/-$	s_0/B
s_7	$s_0/A,2W$	$s_0/B,3W$	s_0/A	$s_0/B,2W$

In dieser Tabelle gibt es keine mit der 1. Regel eliminierbare Zustände mehr. Zur Anwendung der 2. Minimierungsregel werden Zuständen mit identischen Ausgaben in einer Klasse zusammengefasst.

Wir erkennen sofort, dass nur die Zustände s_0 und s_2 eine Klasse bilden können. Alle anderen Zustände unterscheiden sich bezgl. der Ausgaben. Da aber s_0 und s_3 damit Folgezustände haben, die sich in unterschiedlichen Klassen befinden, können auch diese beiden Zustände nicht zusammengelegt werden. Die Anwendung der 2. Regel bringt nichts mehr.

Aufgabe 4.13

Wenn das hypothetische Programm P'' bei Eingabe $Q = P''$ „Hallo“ ausgibt, dann hat Q nicht „Hallo“ ausgegeben, sondern „Ja“. Da das Programm Q aber gerade P'' selbst ist, gibt $Q = P''$ bei Eingabe $Q = P''$ „Ja“ aus, was aber nicht sein kann, da wir gerade damit gestartet sind, dass in genau dieser Situation „Hallo“ ausgegeben wird. Also haben wir auch hier einen Widerspruch erzeugt.

Aufgabe 4.14

Ein Beispiel für einen Quine in C ist das folgende Programm

```
char*f="char*f=%c%s%c;main() {printf(f,34,f,34,10);}%c";main() {printf(f,34,f,34,10);}
```

Der Trick ist hier, dass man einen String definiert, der im Programm zweimal verwendet wird: einmal um im Programmlisting den String auszugeben, und einmal um den restlichen Programmtext auszugeben.

Aufgabe 4.15

Das Produkt bildet wiederum einen n-elementigen Vektor, und jedes Element dieses Vektors berechnet sich aus dem Skalarprodukt einer Matrixzeile mit dem Vektor b, wobei n Elementmultiplikationen und $n-1$ Additionen vorliegen. Insgesamt haben wir also n^2 Multiplikationen und $n(n-1)$ Additionen, also $O(n^2)$ Schritte.

Aufgabe 4.16

Die Laufzeit beträgt $2^{60} \approx 10^{18}$ Schritte. Dies entspricht 10^8 Sekunden und damit etwa 10^3 Tagen. Damit erhalten wir eine Laufzeit von etwa 3 Jahren. Es sei erwähnt, dass Tourenplanungen oft mehr als 100 Orte umfassen.

Aufgabe 4.17

Wenn man sich vom gerade befüllten Container merkt, wieviel er schon enthält, d.h. wieviel Platz noch in ihm frei ist, dann kann man in einer konstanten Anzahl von Schritten entscheiden, ob der nächste Gegenstand noch in diesen Container passt, oder ob man den nächsten Container nehmen muss, der bisher noch nichts enthält. Damit erhält man eine Laufzeit von $O(n)$ für n Gegenstände.

Aufgabe 4.18

Der First-Fit-Decreasing-Algorithmus betrachtet zuerst die k Gegenstände, die jeweils 75% eines Containers füllen, und packt jeden von diesen Gegenständen in einen eigenen Container, da zwei solcher Gegenstände nie zusammenpassen. Dazu braucht er k Container. Nun betrachtet er die Gegenstände, die jeweils 25% eines Containers füllen, und packt jeweils vier von ihnen in einen Container. Dazu braucht er $k/4$ Container, also insgesamt $5k/4$ Container. Würde man jeweils einen großen und einen kleinen Gegenstand zusammenpacken, bräuchte man nur k Container.