HANSER

Rolf Socher

Theoretische Grundlagen der Informatik

ISBN-10: 3-446-41260-3
ISBN-13: 978-3-446-41260-6

2

2 Endliche Automaten

2.1 Einführung

Viele Dinge des täglichen Lebens lassen sich als Automaten auffassen, die verschiedene Zustände einnehmen können. Betrachten wir etwa eine U-Bahn-Eintrittskontrolle mit drei stählernen Armen. Dieses Gerät befindet sich stets in einem der zwei Zustände „verriegelt“ und „entriegelt“. Im verriegelten Zustand lassen sich die Arme nicht drehen. Wirft man nun einen Chip ein, so gibt ein Mechanismus die Arme frei. Die Eingabe eines Chips bewirkt also den Übergang vom verriegelten in den entriegelten Zustand. Durch Drehung der Arme geht das Gerät in den verriegelten Zustand zurück. Die beiden möglichen Aktionen sind das Einwerfen eines Chips und das Drehen der Arme.

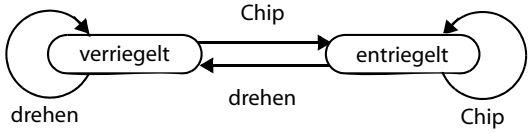

Abbildung 2.1: Der U-Bahn-Automat

Das Verhalten des Gerätes hängt offenbar sowohl von der Aktion (Chip einwerfen oder Arme drehen) als auch von seinem jeweiligen Zustand ab. Abbildung 2.1 zeigt die möglichen Zustandsübergänge: Im verriegelten Zustand bewirkt das Drehen der Arme nichts, während das Einwerfen eines Chips die Sperre entriegelt. Werden im entriegelten Zustand die Arme gedreht, so geht das Gerät in den verriegelten Zustand über. Im entriegelten Zustand ist das Einwerfen eines weiteren Chips dagegen sinnlos, die Sperre bleibt entriegelt. Man bezeichnet ein solches Gerät, das in Abhängigkeit vom aktuellen Zustand und von der jeweiligen Aktion bzw. Eingabe in einen anderen Zustand übergeht, als Automaten. Aus dem Alltag sind Getränke- oder Zigarettenautomaten bestens bekannt, also Geräte, die bei Einwurf eines bestimmten Betrags eine Ware ausgeben.

In der Informatik werden Automaten hauptsächlich verwendet, um Zeichenfolgen auf ihre Plausibilität zu prüfen. Der in Abbildung 2.2 gezeigte Automat A_P kann zur Paritätsprüfung verwendet werden. Die Eingabe besteht aus einer Reihe von Binärziffern. Der Automat prüft, ob diese eine gerade oder ungerade Anzahl von Einsen

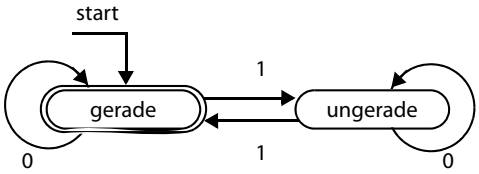

Abbildung 2.2: Der Automat AP zur Paritätsprüfung

enthält. Der Ablauf beginnt im Zustand „gerade“, weil am Anfang noch keine Einsen gelesen wurden und 0 eine gerade Zahl ist. Dies wird durch den Pfeil mit der Beschriftung „start“ angedeutet. Liest der Automat eine Null, so bleibt er im jeweiligen Zustand („gerade“ bzw. „ungerade“), liest er eine Eins, so wechselt er von „gerade“ nach „ungerade“ bzw. von „ungerade“ nach „gerade“.

Hat der Automat das Eingabewort vollständig gelesen, so zeigt der Zustand, in dem er sich dann befindet, ob das Eingabewort gerade oder ungerade Parität hat.

Im Allgemeinen prüft man mit Hilfe eines Automaten, ob die Eingabe einer bestimmten Bedingung genügt (beispielsweise die Bedingung gerader Parität). Zu diesem Zweck führt man so genannte *akzeptierende Zustände* ein (auch *Endezustände* genannt), in unserem Beispiel den Zustand „gerade“. In der graphischen Darstellung werden akzeptierende Zustände mit doppelt umrandeten Knoten gekennzeichnet. Befindet sich der Automat nach der Verarbeitung der Eingabe in einem akzeptierenden Zustand, so bedeutet dies, dass die Eingabe akzeptiert wird.

In den beiden genannten Beispielen kann die Eingabe als ein Wort, also als Folge von Zeichen aufgefasst werden. Im folgenden Abschnitt sollen zunächst die daraus resultierenden Grundbegriffe wie Zeichen, Wörter und Sprachen definiert werden.

2.2 Grundlegende Begriffe

Ein *Alphabet* ist eine endliche, nicht leere Menge, deren Elemente *Zeichen* genannt werden. Ein Zeichen ist eine formale Einheit, die nicht näher definiert wird. Üblicherweise sind dies Ziffern, Buchstaben oder Sonderzeichen.

Die Einschränkung, dass ein Alphabet nicht leer sein darf, wird getroffen, um von vornherein Sonderfälle, die sich aus einem leeren Alphabet ergeben könnten, auszuschließen. Alphabete werden grundsätzlich mit dem Buchstaben Σ bezeichnet.

Ein Wort (über einem Alphabet Σ) ist eine endliche Folge von Zeichen aus Σ. Sind zum Beispiel a, b, c Zeichen, so lassen sich daraus Wörter wie cba oder aaab bilden. Das *leere Wort* wird mit ε bezeichnet. Die Menge aller Wörter, die mit Zeichen aus Σ gebildet werden können, wird mit Σ^* bezeichnet.

Die *Länge eines Wortes* w ist die Anzahl der in w vorkommenden Zeichen. Sie wird mit $|w|$ bezeichnet. Ist etwa w = aabac, so ist $|w| = 5$. Formal lässt sich $|w|$ rekursiv definieren durch die folgende Vorschrift:

$|\varepsilon| = 0$

$|aw| = 1 + |w|$, falls $a \in \Sigma$, $w \in \Sigma^*$.

Diese rekursive Definition zeigt beispielhaft das allgemeine Muster der rekursiven Definition einer Funktion *f*, die auf der Menge Σ^* aller Wörter definiert ist. Die erste Zeile der Definition legt den Funktionswert $f(\varepsilon)$ fest. Die zweite Zeile definiert den Funktionswert für ein Wort aw, dessen erstes Zeichen a ist. In der Definition dieses Funktionswertes dürfen sowohl a und w als auch der Wert $f(w)$ *rekursiv* verwendet werden.

Ist w ein Wort über dem Alphabet Σ und a ein Zeichen aus Σ, so bezeichnet $|w|_a$ die Anzahl der Vorkommen des Zeichens a in w. Ist etwa w = *element*, so ist $|w|_e = 3$, $|w|_m = 1$ und $|w|_x = 0$.

Die *Konkatenation* zweier Wörter ist das Wort, das sich ergibt, wenn man die beiden Wörter direkt hintereinander schreibt. So ergibt die Konkatenation der Wörter *Auto* und *mat* das Wort *Automat*. Manchmal benutzt man zur Verdeutlichung einen expliziten Konkatenationsoperator in Form eines Malpunktes. Man kann also auch $v \cdot w$ statt vw schreiben. Das leere Wort ist das neutrale Element der Konkatenation, d.h. für jedes Wort w gilt $\varepsilon w = w\varepsilon = w$.

Ähnlich wie bei Zahlen verwendet man eine Potenzschreibweise für das mehrfache Konkatenieren von Wörtern. So schreibt man etwa vereinfachend a^4 für das Wort *aaaa* oder $(ab)^3$ für das Wort *ababab*. Allgemein definiert man für ein beliebiges Wort w:

$w^0 = \varepsilon$

$w^n = w \cdot w^{n-1}$, falls $n > 0$.

Ist w ein Wort über einem Alphabet Σ, so bezeichnet w^T die *Spiegelung* von w. Ist etwa w = *automat*, so ist w^T = *tamotua*.

Eine *Sprache* über einem Alphabet Σ ist eine Menge von Wörtern über Σ. Sprachen werden grundsätzlich mit dem Buchstaben L (engl. *language*), gegebenenfalls mit Indizes, bezeichnet. Spezielle Sprachen sind die leere Menge, die Menge Σ^* sowie die Menge $\{\varepsilon\}$, die nur das leere Wort enthält.

Beispiel 2.1:

a) Beim U-Bahn-Automaten wird das Alphabet

Σ = {„Chip“, „drehen“}

verwendet. Man beachte, dass das ganze Wort „Chip“ als Eingabezeichen definiert ist und nicht die einzelnen Zeichen C, h, i und p. Ein Zeichen stellt somit die kleinste unabhängige und unteilbare Einheit der Eingabe dar.

b) Der Automat A_P zur Paritätsprüfung verwendet das Alphabet $\Sigma = \{0, 1\}$. Sprachen über Σ sind etwa die Mengen {0, 10, 0011} oder {ε, 01, 0011, 000111,…} = $\{0^n 1^n \mid n \geq 0\}$. ■

Da Sprachen Mengen sind, sind auf ihnen die Mengenoperationen $\cup$, $\cap$ usw. definiert. Darüber hinaus wird die Konkatenation zweier Sprachen L_1 und L_2 folgendermaßen definiert:

$$L_1 L_2 = \{vw | v \in L_1, w \in L_2\}.$$

Ist etwa $\Sigma = \{0, 1\}$, $L_1 = \{0, 00\}$ und $L_2 = \{1, 11\}$, so ist $L_1L_2 = \{01, 011, 001, 0011\}$.

Ist L eine Sprache, so ist L^n definiert als die n-fache Konkatenation von L mit sich selbst, also

$$L^0 = \{\varepsilon\}$$

$$L^n = L \cdot L^{n-1}, \text{ falls } n > 0.$$

Die Kleenesche Hülle L^* der Sprache L ist folgendermaßen definiert:

$$L^* = \bigcup_{n \geq 0} L^n$$

2.3 Deterministische endliche Automaten

Definition des DEA

Der U-Bahn-Automat und der Automat zur Paritätsprüfung lassen sich jeweils auch in Tabellenform darstellen.

Tabelle 2.1: Automat zur Paritätsprüfung

δ	**0**	**1**
gerade/$_E$	gerade	ungerade
ungerade	ungerade	gerade

Tabelle 2.1 zeigt, dass der Automat vom Zustand „gerade“ bei Eingabe einer 1 in den Zustand „ungerade“ übergeht. Die Tabelle lässt sich auch durch eine Funktion δ beschreiben, die einem Zustand und einem Zeichen jeweils einen Folgezustand zuordnet, also δ(gerade, 1) = ungerade usw. Der Zustand „gerade“ ist mit /$_E$ als Endezustand gekennzeichnet.

Ein *deterministischer endlicher Automat* (im Folgenden schreiben wir meist kurz „Automat“ statt „deterministischer endlicher Automat“) besteht also aus einem Alphabet Σ, das die zulässigen Eingabezeichen auflistet, einer endlichen Menge Z von Zuständen und einer Vorschrift δ, die die Zustandsübergänge beschreibt.

Zusätzlich verfügt ein Automat über einen speziellen Startzustand sowie einen oder mehrere akzeptierende Zustände (auch Endezustände genannt).

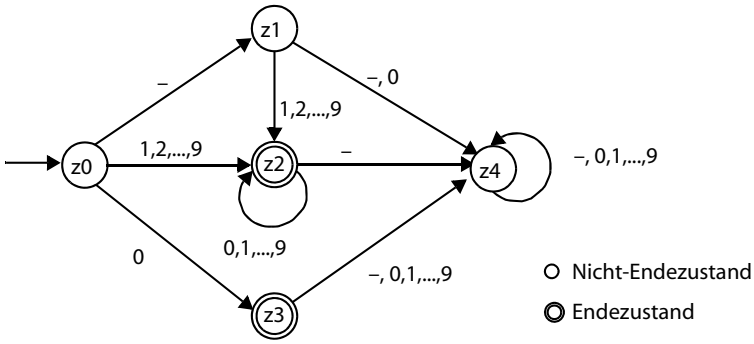

Abbildung 2.3: Ein Automat zur lexikalischen Analyse

Formal definiert man einen Automaten folgendermaßen:

Definition 2.1: Deterministischer endlicher Automat, DEA

Ein *deterministischer endlicher Automat* besteht aus den fünf Komponenten Z, Σ, δ, z_0, E.

- Z ist eine endliche Menge von *Zuständen*.
- Σ ist ein Alphabet, das sog. *Eingabealphabet*.
- $\delta : Z \times \Sigma \rightarrow Z$ ist die *Übergangsfunktion*.
- $z_0 \in Z$ ist der *Startzustand*.
- $E \subseteq Z$ ist die Menge der *Endezustände*.

Im Folgenden werden die Endezustände in der graphischen Darstellung doppelt umrandet und in der Tabellenform mit $/_E$ gekennzeichnet.

Beispiel 2.2: Ein Automat zur lexikalischen Analyse

Der Automat $A_{LA} = (Z, \Sigma, \delta, z_0, E)$ prüft, ob das Eingabewort eine nach der Konvention der Programmiersprache Pascal korrekt geschriebene ganze Zahl ohne führende Nullen darstellt. Als gültige Eingabezeichen kommen die Ziffern 0 bis 9 sowie das Vorzeichen – in Frage, also

$\Sigma = \{–, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Abbildung 2.3 zeigt die graphische Darstellung des Automaten.

Zu Beginn (das heißt im Startzustand z_0) können wir je nach erstem Eingabezeichen drei Fälle unterscheiden:

a) Das erste Zeichen ist eine Null. Der Automat springt in den Zustand z_3. Dieser muss ein akzeptierender Zustand sein, denn 0 ist ein gültiges Eingabewort. Anschließend darf jedoch kein weiteres Zeichen mehr kommen, denn führende Nullen sind nicht erlaubt.

b) Das erste Zeichen ist eine Ziffer ungleich 0. Der Automat springt nach z_2, der auch ein akzeptierender Zustand sein muss. Anschließend dürfen beliebig viele Ziffern, jedoch kein Vorzeichen kommen.

c) Das erste Zeichen ist das Vorzeichen. Der Automat springt nach z_1. Kommt nun eine weitere Ziffer ungleich 0, so springt der Automat in den Zustand z_2. ■

Ein Eingabewort für einen endlichen Automaten A ist ein Wort über dem Alphabet Σ. Kommen in einem Wort Zeichen vor, die nicht dem Eingabealphabet angehören, so kann der Automat dieses Wort nicht verarbeiten.

In Anlehnung an die Pfeile in der graphischen Darstellung schreiben wir auch $z \rightarrow_a z'$, falls $z' = \delta(z, a)$ gilt, sowie $z \rightarrow_\Sigma z'$, falls es ein Zeichen $a \in \Sigma$ gibt mit $z \rightarrow_a z'$.

Verarbeitung eines Eingabewortes

Die Verarbeitung eines Eingabewortes lässt sich mit Hilfe so genannter Konfigurationen beschreiben.

Betrachten wir als Beispiel den Automaten A_{LA} zur lexikalischen Analyse. Das Eingabewort sei $w = -304$. Zu Beginn der Verarbeitung befindet sich der Automat im Zustand z_0, und das Eingabewort ist –304. Aktueller Zustand und verbleibendes Eingabewort bilden zusammen die wesentliche Information, die zur Verarbeitung einer Eingabe nötig ist. Im ersten Schritt wird das Minuszeichen gelesen und der Automat geht in den Zustand z_1 über. Das aktuelle Eingabewort ist nun 304. Dies wird durch den folgenden Übergang beschrieben:

$$(z_0, -304) \rightarrow (z_1, 304).$$

Definition 2.2: Konfiguration

Eine *Konfiguration* eines endlichen Automaten A ist ein Paar (z, w) mit $z \in Z$, $w \in \Sigma^*$. Dabei bezeichnet z den aktuellen Zustand, in dem sich der Automat zu einem Zeitpunkt der Berechnung befindet, und w das verbleibende Eingabewort.

Die *Übergangsrelation* $\rightarrow$ auf der Menge der Konfigurationen von A ist definiert durch

$$(z, aw) \rightarrow (z', w), \text{ falls } z \rightarrow_a z'.$$

Betrachten wir nun die weitere Verarbeitung im obigen Beispiel.

Die komplette Verarbeitung des Wortes $w = -304$ lässt sich beschreiben durch die Folge

$$(z_0, -304) \rightarrow (z_1, 304) \rightarrow (z_2, 04) \rightarrow (z_2, 4) \rightarrow (z_2, \varepsilon).$$

Dies lässt sich unter Verwendung der transitiven, reflexiven Hülle $\rightarrow^*$ (siehe Abschnitt 8.3) der Relation $\rightarrow$ auch folgendermaßen beschreiben:

$(z_0, -304) \rightarrow^* (z_2, \varepsilon)$.

Die Verarbeitung endet in dem Zustand z_2, der ein akzeptierender Zustand ist. Dem entspricht die Tatsache, dass das Eingabewort –304 eine korrekt geschriebene ganze Zahl ist.

Die Verarbeitung des Wortes w = –3–0, welches keine korrekte ganze Zahl darstellt, lässt sich beschreiben durch die Folge

$(z_0, -3-0) \rightarrow (z_1, 3-0) \rightarrow (z_2, -0) \rightarrow (z_4, 0) \rightarrow (z_4, \varepsilon)$

bzw. in Kurzform:

$(z_0, -3-0) \rightarrow^* (z_4, \varepsilon)$.

Der Zustand z_4 ist kein akzeptierender Zustand, also wird das Eingabewort –3–0 nicht akzeptiert.

Ein Eingabewort w wird also genau dann akzeptiert, wenn die Verarbeitung von w in einem akzeptierenden Zustand z_e endet. Die Menge aller Eingabewörter, die vom Automaten A akzeptiert werden, heißt die von A akzeptierte Sprache.

Definition 2.3: Akzeptierung durch einen DEA, reguläre Sprache

Der DEA A *akzeptiert* das Eingabewort w genau dann, wenn es ein $z_e \in E$ gibt mit

$(z_0, w) \rightarrow^* (z_e, \varepsilon)$.

Die vom Automaten A *akzeptierte Sprache* $\mathcal{L}(A)$ ist definiert durch:

$\mathcal{L}(A) = \{w \in \Sigma^* | A \text{ akzeptiert } w\}$.

Eine Sprache L heißt *regulär*, falls es einen DEA A gibt, der L akzeptiert, das heißt, wenn $L = \mathcal{L}(A)$ gilt.

Ist A ein DEA, der die Sprache L akzeptiert, so schreiben wir auch *A ist ein Automat für L*.

Wir verwenden im Folgenden die erweiterte Übergangsfunktion δ: Wir erweitern δ zu einer Funktion $\delta : Z \times \Sigma^* \rightarrow Z$ mit $\delta(z, w) = z'$, falls $(z, w) \rightarrow^* (z', \varepsilon)$. Mit dieser Notation lässt sich die Akzeptierung des Wortes w folgendermaßen beschreiben: A akzeptiert w, wenn $\delta(z_0, w) \in E$ gilt.

Die Sprache des Automaten A_P zur Paritätsprüfung ist die Menge aller Binärwörter mit gerader Parität:

$\mathcal{L}(A_P) = \{w \in \Sigma^* | |w|_1 \text{ ist gerade}\}$.

Die Sprache des Automaten A_{LA} zur lexikalischen Analyse ist die Menge aller korrekt geschriebener Ganzzahlen. Diese beiden Sprachen sind also regulär.

Die beiden in Abbildung 2.4 dargestellten Automaten A_1 und A_2 akzeptieren offenbar beide genau diejenigen Binärwörter, die mit einer Eins enden:

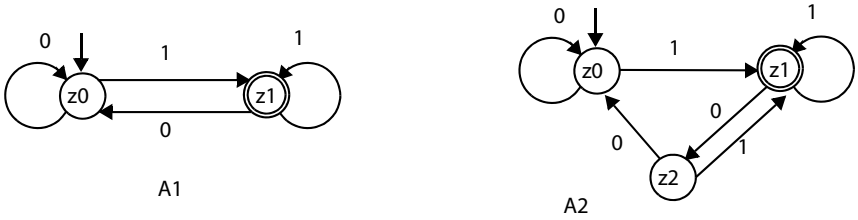

Abbildung 2.4: Zwei äquivalente Automaten

$\mathcal{L}(A_1) = \mathcal{L}(A_2) = \{w1 | w \in \Sigma^*\}$.

Da die Funktion eines Automaten ausschießlich im Akzeptieren von Wörtern besteht, können Automaten, die dieselbe Sprache akzeptieren, als *äquivalent* angesehen werden.

Definition 2.4: Äquivalenz von Automaten

Zwei Automaten A_1 und A_2 heißen *äquivalent*, geschrieben $A_1 \equiv A_2$, wenn sie dieselbe Sprache akzeptieren, das heißt, wenn $\mathcal{L}(A_1) = \mathcal{L}(A_2)$ gilt.

Die Relation ≡ ist eine Äquivalenzrelation (siehe Abschnitt 8.3).

Gerät der Automat A_{LA} zur lexikalischen Analyse (siehe Abbildung 2.3) während der Verarbeitung eines Eingabewortes im Zustand z_4, so bleibt er anschließend in diesem Zustand, insbesondere kann er nie mehr einen akzeptierenden Zustand erreichen. Einen solchen Zustand, von dem aus kein akzeptierender Zustand erreichbar ist, nennt man einen *Fehlerzustand*.

Definition 2.5: Erreichbarkeit, Fehlerzustand

Der Zustand z' heißt *erreichbar* vom Zustand z, falls es ein Wort w gibt, so dass $(z, w) \rightarrow^* (z', \varepsilon)$ gilt. $[z]^*$ bezeichnet die Menge aller von z erreichbaren Zustände.

Ein *Fehlerzustand* eines Automaten A ist ein Zustand, von dem aus kein Endezustand erreichbar ist, also ein Zustand z, für den $[z]^* \cap E = \varnothing$ gilt.

Der Automat A_{LA} zur lexikalischen Analyse befindet sich nach Eingabe eines Minuszeichens im Zustand z_1. Anschließend darf noch eine beliebige korrekt geschriebene ganze Zahl *ohne Vorzeichen* kommen. Man sagt, der Automat *akzeptiert im Zustand* z_1 jede beliebige korrekt geschriebene ganze Zahl ohne Vorzeichen.

Definition 2.6: Akzeptierung im Zustand z

Der DEA A *akzeptiert* das Eingabewort w *im Zustand* z wenn es ein $z_e \in E$ gibt mit

$(z, w) \rightarrow^* (z_e, \varepsilon)$.

Die von *A im Zustand z akzeptierte Sprache* $\mathcal{L}(A, z)$ ist definiert durch:

$\mathcal{L}(A, z) = \{w \in \Sigma^* | A \text{ akzeptiert } w \text{ im Zustand } z\}$.

Mit der erweiterten δ-Notation lässt sich auch schreiben: *A akzeptiert* das Eingabewort w im Zustand z, wenn $\delta(z, w) \in E$ gilt. Aus der Definition folgt: $z \in E$ genau dann, wenn $\varepsilon \in \mathcal{L}(A, z)$.

Wird im Startzustand des Automaten A_{LA} eine Null eingegeben, so kann anschließend keine weitere Zeichenfolge eine gültige ganze Zahl ohne führende Nullen ergeben. Da allerdings das Zeichen Null selbst eine gültige Zahl darstellt, ist in diesem Fall $\mathcal{L}(A_{LA}, z_1)$ nicht leer, sondern es gilt $\mathcal{L}(A_{LA}, z_1) = \{\varepsilon\}$.

Für den Fehlerzustand z_4 dagegen gilt $\mathcal{L}(A_{LA}, z_4) = \emptyset$. Offensichtlich ist ein Zustand z eines DEA A genau dann ein Fehlerzustand, wenn $\mathcal{L}(A, z) = \emptyset$ gilt (siehe Aufgabe 2.13).

2.4 Minimierung endlicher Automaten

Der Minimalautomat

Die beiden in Abbildung 2.4 dargestellten Automaten A_1 und A_2 zeigen, dass es im Allgemeinen für eine Sprache L mehrere Automaten mit unterschiedlich vielen Zuständen gibt, die L akzeptieren. Für die Implementierung eines endlichen Automaten, der eine Sprache L erkennen soll, ist es sicherlich sinnvoll, die Anzahl der erforderlichen Zustände zu minimieren.

Definition 2.7: Minimalautomat

Sei L eine reguläre Sprache über dem Alphabet Σ und sei A ein Automat, der L akzeptiert, das heißt $L = \mathcal{L}(A)$.

A heißt *Minimalautomat* für L, falls es keinen Automaten A' für L gibt, der weniger Zustände als A hat.

Isomorphie endlicher Automaten

Der Minimalautomat für L ist sogar eindeutig bestimmt, das heißt, sind A_1 und A_2 zwei Minimalautomaten für L, so unterscheiden sie sich lediglich in der Benennung der Zustände. Die beiden in Tabelle 2.2 gezeigten Automaten A_1 und A_2 sind offenbar strukturgleich, sie haben dasselbe Eingabealphabet, gleich viele Zustände, gleich viele Endezustände und auch die Verbindungsstruktur ist die gleiche. Man

kann A_1 aus A_2 erhalten, indem man die Zustände folgendermaßen umbenennt: $z_0 \to w_0$, $z_1 \to w_2$, $z_2 \to w_3$, $z_3 \to w_4$, $z_4 \to w_1$.

Tabelle 2.2: Zwei isomorphe Automaten

A_1	0	1
z_0	z_1	z_3
z_1	z_0	z_4
$z_2/_E$	z_4	z_2
$z_3/_E$	z_4	z_3
z_4	z_2	z_4

A_2	0	1
w_0	w_2	w_4
w_1	w_3	w_1
w_2	w_0	w_1
$w_3/_E$	w_1	w_3
$w_4/_E$	w_1	w_4

Formal wird die Strukturgleichheit von Automaten folgendermaßen formuliert:

Definition 2.8: Isomorphie von Automaten

Die beiden Automaten $A_1 = (Z_1, \Sigma, \delta_1, z_{01}, E_1)$ und $A_2 = (Z_2, \Sigma, \delta_2, z_{02}, E_2)$ heißen *isomorph*, wenn es eine bijektive Abbildung $f : Z_1 \to Z_2$ gibt mit:

$f(z_{01}) = z_{02}$
$f(E_1) = E_2$
$z \to_a z'$ gdw.[1] $f(z) \to_a f(z')$.

Im Folgenden soll ein algorithmisches Verfahren entwickelt werden, um einen gegebenen DEA A zu minimieren. Ziel der Minimierung ist es, einen DEA $\overline{A}$ mit folgenden Eigenschaften zu konstruieren:

- $\overline{A}$ ist äquivalent zu A und
- es gibt keinen Automaten $A' \equiv A$, der weniger Zustände als $\overline{A}$ hat.

Entfernen nicht erreichbarer Zustände

Beispiel 2.3:

Als Beispiel betrachten wir den in Abbildung 2.5 dargestellten Automaten A. Es ist leicht zu erkennen, dass der Zustand z_3 überflüssig ist, da er vom Startzustand z_0 aus niemals erreicht werden kann, das heißt, es gilt $z_3 \notin [z_0]^*$. Man kann also einen Automaten A' konstruieren, indem aus A der Zustand z_3 entfernt wird. Dieser Automat A' ist sicherlich äquivalent zu A, denn ist $w \in \mathcal{L}(A)$, so gibt es eine Berechnung $(z_0, w) \to_A^* (z_e, \varepsilon)$. Darin kommt der Zustand z_3 nicht vor, da er vom Startzustand

[1] „gdw." heißt „genau dann, wenn"