W.-M. Lippe

Grundlagen der theoretischen Informatik

Vorwort

1 Mathematische Grundlagen

Die Mathematik bietet mit ihrer formalen Ausdrucksweise eine präzise Grundlage zur Beschreibung von Informatik – Sachverhalten. Daher werden in diesem Abschnitt – ohne in die Details zu gehen – die wichtigsten mathematischen Grundbegriffe zusammen – gefaßt, die im weiteren Verlauf verwendet werden

1.1 Mengen

Eine Menge M ist die Zusammenfassung von Elementen zu einem Ganzen. In der klassischen Mathematik und Logik ist eine Menge M auf einer Grundmenge G durch ihre Elemente bestimmt. Für jedes Element x der Grundmenge G wird festgelegt, ob es zur Menge M gehören soll („ $x \in M$ “) oder nicht zu M gehören soll („ $x \notin M$ “).

Die Anzahl der Elemente einer Menge M (Mächtigkeit von M) wird mit |M| bezeichnet und kann endlich oder durch unendlich sein. Es gibt verschiedene Darstellungsformen für Mengen: eine Menge M kann in aufzählender Form, in beschreibender Form oder auch graphisch mit Hilfe von Diagrammen angegeben werden.

Die aufzählende Form, auch Listenform genannt, wird meist für endliche Mengen benutzt, z. B.

$$M = \{a_1, a_2, \cdots, a_n\}$$

für eine n-elementige Menge M. Manchmal wird diese Form auch für unendliche Mengen benutzt, wenn deren Elemente als Glieder einer Folge mit leicht erkennbarem Bildungsgesetzt gegeben sind, z. B.

$$M = \{2, 4, 6, 8, \cdots\}$$

für die Menge der geraden natürlichen Zahlen.

Beliebige, insbesondere unendliche Mengen, werden meist in der beschreibenden Form durch charakteristische Prädikate (Eigenschaften) erklärt. Man schreibt:

$$M = \{x \in G \mid P(x)\}$$

und sagt, *M* besteht aus den Elementen *x* der Grundmenge *G*, welche das Prädikat *P* erfüllen). Die oben aufgeführte Menge der geraden natürlichen Zahlen läßt sich mittels

$$M = \{n \in IN \mid n \bmod 2 = 0\} = \{n \in IN \mid n \; ist \; gerade\}$$

auch in der beschreibenden Form angeben.

Eine weitere Möglichkeit, eine Menge *M* auf einer Grundmenge *G* zu beschreiben, besteht in der Angabe einer charakteristischen Funktion χ_M für $x \in G$. Die charakteristische Funktion hat die Grundmenge *G* als Definitionsbereich und die zweiwertige Menge $\{0,1\}$ als Wertevorrat. Gehört ein $x \in G$ zu *M,* so nimmt die charakteristische Funktion χ_M den Wert 1 an, anderenfalls gilt $\chi_M = 0$.

Definition 1.1 (Charakteristische Funktion)

Sei *G* eine Grundmenge und *M* eine Teilmenge von *G*. Dann heißt die Funktion

$$\chi_M : G \to \{0,1\}$$
$$\chi_M(x) = \begin{cases} 1 : x \in M \\ 0 : x \notin M \end{cases}$$

die Identifikator- oder charakteristische Funktion der Menge *M.*

Zwischen der Definition einer Menge in der beschreibenden Form mittels eine Prädikates und der Definition einer Menge durch ihrs charakteristische Funktion besteht kein relevanter Unterschied. Das charakteristische Prädikat kann für ein $x \in G$ entweder wahr oder falsch sein, je nachdem, ob x die Eigenschaft *P* besitzt oder nicht besitzt. Die charakteristische Funktion nimmt in Analogie den Wert 1 an, wenn $x \in G$ zu *M* gehört bzw. den Wert 0 für diejenigen $x \in G$ an, welche nicht zu *M* gehören.

Beispiel 1.1

Die oben angegebene Menge *M* der geraden zahlen auf der Grundmenge der natürlichen Zahlen läßt sich alternativ beschreiben durch

Aufzählende Form
$M = \{2,4,6,8,\ldots\}$

Charakteristische Prädikate
$M = \{n \in IN \mid n \bmod 2 = 0\} = \{n \in IN \mid n \; ist \; gerade\}$

Charakteristische Funktion
$\chi_M(x) = 1 - (x \bmod 2)$

In der klassischen Mengenlehre sind zwei Mengen von besonderer Bedeutung, die Universalmenge und die leere Menge:

Definition 1.2 *(Universalmenge, leere Menge)*

Es sei *G* eine Grundmenge und *A* eine Menge über *G*. *A* heißt Universalmenge, wenn sie gleich der Grundmenge *G* ist, also jedes Element der Grundmenge schon zu *A* gehört. *A* heißt leere Menge, wenn kein Element der Grundmenge in *A* enthalten ist und wird mit ϕ bezeichnet.

Es ist *A* eine Universalmenge, wenn das *A* definierende Prädikat *P* allgemeingültig ist, oder anderes ausgedrückt, die charakteristische Funktion der Menge A, χ_A, für alle Elemente der Grundmenge *G* den Wert 1 annimmt. Ist *A* eine leere Menge, so ist charakteristische Funktion von A, χ_A, für alle $x \in G$ gleich 0. Das charakteristische Prädikat *P* der Menge *A* heißt in diesem Fall unerfüllbar (inkonsistent), es erfüllt kein Element der Grundmenge die Eigenschaft *P*.

Von besonderer Bedeutung sind ferner insbesondere noch die Mengen:

$\mathbb{N}$	:	Menge der natürlichen Zahlen ohne die Null
$\mathbb{N}_0$	:	Menge der natürlichen Zahlen einschließlich der Null
$\mathbb{Z}$	:	Menge der ganzen Zahlen
$\mathbb{Q}$	:	Menge der rationalen Zahlen
$\mathbb{R}$	:	Menge der reellen Zahlen

In Definition 1.1 wurde bereits der Begriff der Teilmenge verwendet. Dieser Begriff muß noch genauer definiert werden. Anschaulich kann dies folgendermaßen erfolgen:

Seien *M, N* zwei klassische Mengen über der Grundmenge *G*. *M* heißt eine Teilmenge von $("M \subseteq N")$, wenn jedes Element der Menge *M* auch in $N("M \subset N")$ enthalten ist. M heißt eine echte Teilmenge von $N("M \subset N")$, falls $M \subseteq N$ gilt und N noch weitere Elemente der Grundmenge *G* enthält, die nicht in *M* enthalten sind. Die beiden Mengen heißen gleich $("M = N")$, falls beide Mengen genau die gleichen Elementen enthalten, oder anders ausgedrückt, wenn sowohl $M \subseteq N$ als auch $N \subseteq M$ gilt. Die Teilmengenbeziehung und die Gleichheit zweier klassischer Mengen kann aber auch über die charakteristischen Funktionen definiert werden:

Definition 1.1 (Teilmengen und Gleichheit klassischer Mengen)

Es seien M und N zwei klassische Mengen über der Grundmenge G und $\chi_M(x), \chi_N(x): G \to \{0,1\}$ die charakteristischen Funktionen von M und N. Es heißt dann M eine Teilmenge von N (in Zeichen „$M \subseteq N$ "oder auch „$N \supseteq M$ "), wenn

$$\chi_M(x) \le \chi_N(x) \qquad \forall\, x \in G$$

gilt. Es heißt M eine echte Teilmenge von N (in Zeichen „$M \subset N$ " oder auch „$N \supset M$ "), falls

$$\left(\forall x \in G: \quad \chi_M(x) \le \chi_N(x)\right) \quad \wedge \quad \left(\exists\, x \in G: \left(\chi_M(x) < \chi_N(x)\right)\right)$$

gilt. Weiter heißen M und N gleich (in Zeichen: („$M = N$ "), falls gilt:

$$\chi_M(x) = \chi_N(x) \qquad \forall\, x \in G$$

Definition 1.4 (Potenzmenge)

Die Menge aller Teilmengen von *M* heißt Potenzmenge von *M* (in Zeichen $\wp(M)$ oder 2^M).

Ist die Mächtigkeit von *M* gleich $n(|M| = n)$ so enthält $\wp^n$ insgesamt 2^n Teilmengen, wovon 2^{n-1} echte Teilmengen sind. Ferner gilt

$$\phi \in \wp(M) \quad \text{und} \quad M \in \wp(M)$$

Hierbei steht „$\forall x \in G$“ für „für alle Elemente x aus G“ und „$\exists x_G G$“ für „es existiert ein Element x in G“.

Neben diesen Beziehungen zwischen Mengen existieren weiter Mengenoperationen wie Komplementbildung, Schnitt und Vereinigung:

Definition 1.5 (Mengenoperationen auf klassischen Mengen)

Für zwei klassische Mengen A, B auf einer Grundmenge G definiert man folgende Mengenoperationen:

1. das Komplement von A, $A^c = \{ x \in G \,|\, x \notin A \}$
2. den Schnitt von A und B, $A \cap B = \{ x \in G \mid x \in A \wedge x \in B \}$
3. die Vereinigung von A und B, $A \cup B = \{ x \in G \mid x \in A \vee x \in B \}$
4. die Differenzmenge A ohne B, $A - B = \{ x \in G \mid x \in A \wedge x \notin B \}$
5. die symmetrische Differenz von A und B, $A * B = (A - B) \cup (B - A)$
6. die Potenzmenge von A, $\wp(A) = \{ X \subseteq G \mid X \subseteq A \}$
7. das kartesische Produkt von A und B, $A \times B = \{ (x, y) \,|\, x \in A \wedge y \in B \}$

Die Definition 1.5 kann auch auf der Basis der charakteristischen Funktion erfolgen, man erhält z.B.

$$\chi_{A^c}(x) = (1 - \chi_A(x))$$
$$\chi_{A \cap B}(x) = \min(\chi_A(x), \chi_B(x))$$
$$\chi_{A \cup B}(x) = \max(\chi_A(x), \chi_B(x))$$

Aus den hier definierten Mengenoperationen für klassische Mengen lassen sich die in Tabelle 1.1 dargestellten Eigenschaften der Mengenoperationen auf crispen Mengen beweisen.

$\left(A^c\right)^c = A$	Involution
$A \cup B = B \cup A$ $A \cap B = B \cap A$	Kommutativität
$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$	Assoziativität
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	Distributivität
$A \cap A = A$ $A \cup A = A$	Idempotenz
$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$	Absorption
$A \cup \left(A^c \cap B\right) = A \cup B$ $A \cap \left(A^c \cup B\right) = A \cap B$	Absorption des Komplements
$A \cup G = G$ $A \cap \phi = \phi$	Identität
$(A \cap B)^c = A^c \cup B^c$ $(A \cup B)^c = A^c \cap B^c$	De Morgansche Gesetze
$A \cap A^c = \phi$	Gesetz vom Widerspruch
$A \cup A^c = G$	Gesetz vom ausgeschlossenen Dritten

Tabelle 1.1 Eigenschaften der Mengenoperationen für klassische Mengen

Statt A^c ist auch die Notation $\overline{A}$ gebräuchlich. Die Korrektheit der Eigenschaften läßt sich unmittelbar aus den Definitionen ableiten. Unter Verwendung der charakteristischen Funktion erhält man z. B.

Beispiele 1.2

1. Die Korrektheit von $\left(A^c\right)^c \equiv \bar{\bar{A}} = A$ ergibt sich aus

$$\chi_{\left(A^c\right)^c}(x) = 1 - \chi_{A^c}(x) = 1 - \left(1 - \chi_A(x)\right) = \chi_A(x)$$

2. Die Korrektheit von $A \cup A = A$ ergibt sich aus

$$\chi_{A \cup A}(x) = \max\left(\chi_A(x), \chi_A(x)\right) = \chi_A(x)$$

3. Die Korrektheit von $\chi_{(A \cap B)^c}(x) = \chi_{A^c \cup B^c}(x)$ ergibt sich aus

$$\chi_{\left(A \cap B^c\right)}(x) = 1 - \chi_{A \cap B}(x) = 1 - \min\left(\chi_A(x), \chi_B(x)\right)$$

o.B.d.A sei $\chi_A(x) \geq \chi_B(x)$

$\Rightarrow \min\left(\chi_A(x), \chi_B(x)\right) = \chi_B(x)$ und

$\max\left(1 - \chi_A(x), 1 - \chi_B(x)\right) = 1 - \chi_B(x)$

$\Rightarrow$

$$\begin{aligned} 1 - \min\left(\chi_A(x), \chi_B(x)\right) &= 1 - \chi_B(x) \\ &= \max\left(1 - \chi_A(x), 1 - \chi_B(x)\right) \\ &= \chi_{A^c \cup B^c}(x) \end{aligned}$$

Definition 1.6 (geordnetes Tupel, Kartesisches Produkt)

1. Seien $A_1, \ldots, x_n)$ Mengen und $x_i \in A_1, \ldots, x_n \in A_n$, dann heißt $(x_1, \ldots, x_n)$ ein geordnetes Tupel von Elementen über $A_1, \ldots, A_n$.

2. Die Menge aller geordneten Tupel $(x_1, \ldots, x_n)$ mit $x_1 \in A_1, \ldots, x_n \in A_n$ heißt kartesisches Produkt Kreuzprodukt oder Mengen $A_1, \ldots, A_n (A_1 x \ldots x A_n)$, d. h. $(A_1 x \ldots x A_n) := \left\{(x_1, \ldots x_n) \mid x_1 \in A_1, \ldots, x_n \in A_n\right\}$.

Beispiel 1.3

Seien $A=\{1,2\}$ *und* $B=\{2,3\}$ dann gilt:

$A\cup B=\{1,2,3\}$, $A\cap B=\{2\}$, $A-B-\{1\}$

$A\times B=\{(1,2),(1,3),(2,2),(2,3)\}$

$\wp(A)=\{\phi,\{1\},\{2\},\{1,2\}\}$

1.2 Relationen

Mit Hilfe des Kartesischen Produktes lassen sich Relationen definieren gemäß

Definition 1.7 (Relation)

Eine n-stellige Relation R über den Mengen $A_1,\ldots,A_n$ ist eine Teilmenge der Menge $A_1x\ldots xA_n$

$(R\subseteq A_1x\ldots xA_n)$

Für n=2 spricht man auch von einer binären Relation. Für $R\subseteq M\times M$ spricht man auch von „Relation auf M“. An Stelle von $(x,y)\in R$ schreibt man in diesem Fall auch xRy (Infix-Notatun).

Die nachfolgenden Eigenschaften für Relationen sind von besonderer Bedeutung:

Definition 1.8 (Eigenschaften von Relationen)

Eine binäre Relation R auf einer Menge M heißt

symmetrisch genau dann, wenn	$\forall x,y\in M:(xRy\Rightarrow yRx)$
antisymmetrisch genau dann, wenn	$\forall x,y\in M:(xRy\wedge yRx\Rightarrow x=y)$
asymmetrisch genau dann, wenn	$\forall x,y\in M:(xRy\Rightarrow\neg(yRx))$
reflexiv genau dann, wenn	$\forall x\in M:xRx$
irreflexiv genau dann, wenn	$\forall x\in M:\neg(xRx)$
transitiv genau dann, wenn	$\forall x,y,z\in M:(xRy1yR_2\Rightarrow xR_2$

Hierbei bedeuten $"\Rightarrow","\wedge"$ *und* $"\neg"$ die logischen Operatoren „es folgt“, „und“ und „Negation“, die in den nächsten Abschnitten genauer definiert werden.

Definition 1.9 (Äquivalenzrelation, Ordnungsrelation)

Eine Relation R auf einer Menge M heißt

1. Äquivalenzrelation, wenn sie symmetrisch, reflexiv und transitiv ist
2. Ordnungsrelation, wenn sie antisymmetrisch, reflexiv und transitiv ist.

Tabelle 1.2

Die Grundmenge sei die Menge der ganzen Zahlen, also $M = \mathbb{Z}$. Wir betrachten die binären Relationen *gleich* (=), *kleiner*(<) und *kleiner gleich* ($\leq$) und geben ihre Eigenschaften an:

	=	<	$\leq$
symmetrisch	ja	nein	nein
antisymmetrisch	ja	ja	ja
asymmetrisch	nein	ja	nein
reflexiv	ja	nein	ja
irreflexiv	nein	ja	nein
transitiv	ja	ja	ja
Äquivalenzrelation	ja	nein	nein
Ordnungsrelation	ja	nein	ja

Ist eine Äquivalenzrelation R auf einer Menge M gegeben, so wird M durch R in disjunkte nichtleere Äquivalenzklassen unterteilt:

$$M = M_1 \cup M_2 \cup \ldots$$

wobei für jedes i und j mit $i \neq j$ gilt

1) $M_i \cap M_j = \phi$
2) $\forall a,b \in M_i \Rightarrow aRb \text{ ist wahr}$

3) $\forall a \in M_i \wedge \forall b \in M_j \Rightarrow aRb \; ist \; falsch$
$(\neg(aRb))$

Definition 1.10 (Hüllen)

Sei R eine Relation auf M.
Seien ferner die Relationen

$R^o := \{(x,x) \mid x \in M\}$ *und*

$R^i := R^i R$

$:= \{(x,2) \in M \times M \mid \exists y \in M : (x,y) \in R^i, (y,2) \in R\}$

induktiv für alle $i \in \mathbb{N}_o$ gegeben.

Dann heißt

$$R^+ := R^1 \cup R^2 \cup \ldots$$

die transitive Hülle und

$$R^* := R^0 \cup R^1 R^2 \vee \ldots$$

die transitiv reflexive Hülle von R

Beispiel 1.4

Sei die Relation R gegeben durch

$$R = \{(1,2),(2,2),(2,3)\}.$$

Dann ergibt sich die transitive Hülle R^+ zu

$$R^+ = \{(1,2),(2,2),(2,3),(1,3)\}$$

und die transitiv reflexive Hülle R^* zu

$$R^* = \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\}$$

1.3 Funktion

Funktionen sind Relationen mit speziellen Eigenschaften, die zunächst definiert werden.

Definition 1.11 (Eindeutigkeit, Tetalität)

1. Eine Relation $R \subseteq M \times N$ heißt linkstotal, wenn gilt

$$\forall x \in M : ((\exists y \in N) : ((x, y) \in R))$$

2. Eine Relation $R \subseteq M \times N$ heißt rechtstotal, wenn gilt

$$\forall y \in N : ((\exists x \in M) : ((x, y) \in R))$$

3. Eine Relation $R \subseteq MN$ heißt rechtseindeutig, wenn gilt

$$\forall x, \in M, y, z \in N : ((x, y) \in R \wedge (x, z) \in R \Rightarrow y = z)$$

4. Eine Relation $R \subseteq M \times N$ heißt linkseindeutig, wenn gilt

$$\forall x \in N, y, z \in M : ((y, x) \in R \wedge (z, x) \in R \Rightarrow y = z)$$

Mit diesen Definitionen lassen sich Funktionen als spezielle Relationen einführen.

Definition 1.12 (Funktionen)

1. Eine rechtseindeutige Relation $F \subseteq M \times N$ heißt Funktion (partielle Funktion) von M nach N.
2. Ist F zusätzlich linkstotal, so heißt F eine total definierte Funktion.

Folgende Funktionen mit speziellen Eigenschaften spielen eine besondere Rolle.

Definition 1.13 (Surjektion, Injektion, Bijektion)

Sei $F \subseteq M \times N$ eine total definierte Funktion.

1. Ist F rechtstotal, so heißt F Surjektion (bzw. man sagt „F ist surjektiv“).
2. Ist F linkseindeutig, so heißt F Injektion (bzw. man sagt „F ist injektiv“).
3. Ist F gleichzeitig surjektiv und injektiv, so heißt F Bijektion (bzw. man sagt „F ist bijektiv“).

Zu beachten ist, daß bei den obigen Definitionen M und N auch kartesische Produkte sein könne. Ist $M = M_1 \times M_2 \times \ldots M_n$, so spricht man von einer n-stelligen Funktion.

Anstelle des Begriffs „Relation“ wird auch der Begriff „Abbildung“ verwendet. Seien $M = M_1 \times \ldots \times M_n$ und N Mengen. Die Abbildung (Relation) f von M in N ordnet jedem Element $x = (x_1 \ldots., x_n)$ einer gewissen Teilmenge D(f) von M genau ein Element $y \in N$ zu. Man schreibt

$$f : M_1 \times \ldots \times M_n \to N$$
$$f : (x_1, \ldots, x_n) = y$$

Hierbei ist n die Stelligkeit, D(f) der Definitionsbereich und N der Wertebereich. Die obige Schreibweise nennt man auch „Funktionsschreibweise“.

Einen Spezialfall der Abbildungen stellen Verknüpfungen dar. Sie sind definiert durch

Definition 1.14 (Verknüpfungen)

Gilt $T \subseteq \underbrace{(A \times A \times \ldots \times A)}_{n-mal} \times B$ *mit* $A \subseteq B$ so heißt T n-fache (n-stellige) Verknüpfung.

Beispiele für 2-stellige Verknüpfungen sind z. B. die Addition, Subtraktion und Multiplikation auf $\mathbb{N}_0$. Für 2-stellige Verknüpfungen lassen sich die folgenden Eigenschaften definieren:

Definition 1.15 (Verknüpfungseigenschaften)

Sei $\circ : A \times A \to B$ eine 2-stellige Verknüpfung

1. $\circ$ heißt abgeschlossen, wenn B=A gilt

2. $\circ$ heißt assoziativ, wenn für je drei beliebige Elemente $x, y, 2 \in A$ gilt:

$$(x \circ y) \circ 2 = x \circ (y \circ 2)$$

3. $\vee \in A$ heißt neutrales Element bzgl. wenn für alle $x \in A$ gilt: $\vee \circ x = x \circ \vee = x$
4. Für eine abgeschlossene 2-stellige Verknüpfung $\circ$ mit einem neutralen Element $\vee$ heißen $x, y \in A$ invers zueinander, wenn gilt

$$x \circ y = y \circ x = \vee$$

Beispiel 1.5

1. Die Verknüpfung

$$+ : \mathbb{N} \circ \times \mathbb{N} \circ \rightarrow \mathbb{N} \circ$$

ist abgeschlossen, assoziativ und bezieht das neutrale Element O

2. Die Verknüpfung

$$- : \mathbb{N} \circ \times \mathbb{N} \circ \rightarrow \mathbb{Z}$$

ist

- - nicht abgeschlossen $(\mathbb{Z} \neq \mathbb{N} \circ)$
- - nicht assoziativ $((5-3)-2 \neq 5-(3-2)$
- - ohne neutrales Element
- - ohne inverse Elemente

1.4 Halbgruppen und Monoide

Mengen mit auf ihnen definierte Verknüpfungen stellen wichtige algebraische Strukturen dar. Von besonderer Bedeutung sind Halbgruppen und Monoide.

Definition 1.15 (Halbgruppen, Monoid)

1. Eine Halbgruppe H besteht aus einer Menge Y und einer auf A abgeschlossenen und assoziativen Verknüpfung $\circ$ [in Zeichen [H, $\circ$]).
2. Existiert in A zusätzlich ein neutrales Element, so heißt H Monoid

Oft ist es nützlich und sinnvoll, Abbildungen zwischen algebraischen Strukturen wie Halbgruppen, Monoiden, etc. zu betrachten. Diese sind dann besonders wichtig, wenn sie die Verknüpfungseigenschaften der Strukturen erhalten.

Definition 1.17 (Homomorphismus)

Es seien $H_1 = [A, \circ]$ und $H_2 = [B, *]$ Halbgruppen (bzw. Monoide mit neutralen Elementen v_1 und v_2). Eine Abbildung $f : A \to B$ heißt ein Homomorphismus von A in B, wenn

$$f(x_1 \circ x_2) = f(x_1) * f(x_2)$$

für je zwei beliebige Elemente $x_1, x_2 \in A$ gilt. Im Falle eines Monoids muß zusätzlich noch $f(v_1) = v_2$ gelten.

Bei einem Homomorphismus ist es also gleichwertig, ob man zuerst die Verknüpfung $x_1 \circ x_2$ in A ausführt und das Ergebnis der Verknüpfung in B abbildet, oder ob man zuerst die Ausgangselemente x_1 und x_2 in B abbildet und dort die Verknüpfung * durchführt.

In Definition 1.11 wurden die Eigenschaften surjektiv, injektiv und bijektiv eingeführt. Die Übertragung dieser Begriffe auf Homomorphismen führt zu verschiedenen Arten von Momomorphismen.

Definition 1.18 (Epi-, Mono-, Iso-, Endo- und Automorphismus)

Es seien H_1, H_2 und f wie in Definition 1.17 gegeben. Ist f zusätzlich surjektiv bzw. injektiv bzw. bijektiv, so heißt f Epimorphismus bzw. Monomorphismus bzw. Isomorphismus. Im letzten Fall heißen H_1 und H_2 zueinander isomorph (in Zeichen: ($H_1 \simeq H_2$).

Ist weiter $H_1 = H_2$, so heißt f ein Endomorphismus, und falls f zusätzlich noch bijektiv ist, ein Automorphismus.

Aufgrund der Bijektivität folgt bei Isomorphismen aus $H_1 \simeq H_2$ auch $H_2 \simeq H_1$, und man kann leicht nachweisen, daß die Isomorphie eine Äquivalenzrelation für Halbgruppen bzw. Monoide ist.

1.5 Zeichenreiten

Maschinen verarbeiten Kombinationen von Zeichen und geben solche Kombinationen aus. Im einfachsten Fall sind diese Kombinationen Folgen von einfachen Zeichen, z. B. Folgen bestehend aus Buchstaben und Ziffern. Daneben existieren komplizierter Ein- und Ausgaben, z. B. graphischer Natur oder Folgen von komplexen Strukturen. Daher kommt der Theorie der Zeichenreihen eine besondere Bedeutung zu

Definition 1.19 (Zeichenvorrat, Alphabet)

1. Ein Zeichenvorrat ist eine nichtleere, endliche Menge A
2. Ist A durch eine Relation $\prec \cdot$ total geordnet, so heißt A Alphabet

Teilweise werden in der Literatur die beiden Begriffe aus Definition 1.19 synonym verwendet, d. h. auf 2. wird verzichtet.

Definition 1.20 (Zeichenreihe, Wort, Sprach)

Unter einer Zeichenreihe Z über einem Zeichenvorrat A versteht man eine total definierte Z, die ein Intervall [1:n], $n \in \mathbb{N}_0$, in A abb. n heißt Länge von Z (n=|Z|).

Eine Zeichenreihe wird auch Wort genannt. Das leere Wort wird mit ε bezeichnet.

Eine Menge von Zeichenreihen bezeichnet man als Sprache oder Programm.

Folgende Abkürzungen sind gebräuchlich:

- A^* bezeichnet die Menge aller Zeichenreihen über A, einschließlich ε.

- A^+ bezeichnet die Menge aller Zeichenreihen über A ohne ε .
- A^n bezeichnet die Menge aller Zeichenreihen über A mit der Länge n.
- $a^n, a \in A$ ist ein Wort der Länge n, das nur aus a's besteht.

Definiert man auf $A*$ die Verknüpfung $\circ$ durch die Konkatenation („Hintereinanderschreiben") zweier Worte

$$b_1,\ldots b_n \circ c_1 \ldots c_m = b_1 \ldots b_n c_1 \ldots c_m,$$

dann ist $\left[A^*, \circ\right]$ ein freies Monoid.

1.6 Aussagenlogik

Die klassische Aussagenlogik ist ein Teilgebiet der formalen Logik, in dem inhaltliches Denken und Schließen durch formale Kalküle modelliert wird. Aussagen werden dabei nicht nach inhaltlichen Überlegungen, sondern rein formal nach vorgegebenen Vorschriften bewertet (extensionaler Standpunkt). Kennzeichnend für die klassische Aussagenlogik ist die Beschränkung auf zwei Wahrheitswerte (wahr, falsch), das sogenannte Zweiwertigkeitsprinzip.

Ein zulässiger Ausdruck (aussagenlogischer Ausdruck, Formel) ist eine endliche Zeichenreihe (Wort), die induktiv über dem Alphabet (Zeichenvorrat) $A = \{a,\ldots,z,\vee,\wedge,\neg,\rightarrow,\leftrightarrow,(,),W,F\}$ definiert ist.

Definition 1.21 (Syntax der Aussagenlogik, zulässiger Ausdruck)

1. Kleine lateinische Buchstaben sind zulässige Ausdrücke.
2. Die Zeichen W, F sind zulässige Ausdrücke.
3. Sind A und B zulässige Ausdrücke, so sind auch
   $(A), \neg A, (A \vee B), (A \wedge B), (A \rightarrow B), (A \leftrightarrow B)$
   zulässige Ausdrücke.

Zur Vereinfachung der Schreibweise werden folgende Prioritäten für die Verknüpfung von zulässigen Ausdrücken vereinbart: „$\neg$" vor „$\vee$" (und „$\wedge$") vor „$\rightarrow$" vor „$\leftrightarrow$". Die damit entbehrlichen Klammern und die äußersten Begrenzungsklammern können entfallen. Bei gemeinsamen Auf-

treten von „$\vee$“ und „$\wedge$“ muß die Reihenfolge ihrer Ausführung in jedem Fall durch entsprechende Klammerung geregelt werden, da sie gleiche Priorität besitzen.

Dem Begriff des zulässigen Ausdrucks wird mit Hilfe von Wahrheitswerten und der Wahrheitswertefunktion δ eine Bedeutung (Interpretation) zugeordnet.

Definition 1.22 (Semantik der Aussagenlogik)

1. „W“ steht für den Wahrheitswert „wahr“ (also eine wahre Aussage), „F“ steht für den Wahrheitswert „falsch“ (also eine falsche Aussage).
2. Ein kleiner lateinischer Buchstabe bezeichnet eine Aussagenvariable. Eine Aussagenvariable a läßt sich als „wahr“ oder „falsch“ interpretieren, indem ihr durch eine Funktion δ wie folgt ein Wahrheitswert zugeordnet wird:
   $\delta(a) = W$: a wird mit einer wahren Aussage belegt
   $\delta(a) = F$: a wird mit einer falschen Aussage belegt.
3. Die Zeichen $\neg$ (nicht), $\vee$ (oder), $\wedge$ (und), $\rightarrow$ (wenn – dann) und $\leftrightarrow$ (genau dann – wenn) heißen Negation, Disjunktion, Konjunktion, Subjunktion bzw. Bijunktion und dienen nach Definition 3.4 zur Verknüpfung von zulässigen Ausdrücken. Ihre Bedeutung wird durch die Tabelle 3.2 festgelegt.
4. Für einen n-stelligen zulässigen Ausdruck $A(x_1,\cdots,x_n)$ berechnet sich der Wahrheitswert $\delta(A(x_1,\cdots,x_n))$ bei gegebenen Wahrheitswerten $\delta(x_1),\cdots,\delta(x_n) \in \{W,F\}$ gemäß
   $$\delta(A(x_1,\cdots,x_n)) = A(\delta(x_1),\cdots,\delta(x_n)).$$

Tabelle 1.3 Formale Bedeutung der Verknüpfungsoperatoren für Aussagen

		Negation	Disjunktion	Konjunktion	Subjunktion	Bijunktion
$\delta(a)$	$\delta(b)$	$\delta(\neg(a))$	$\delta(a \vee b)$	$\delta(a \wedge b)$	$\delta(a \rightarrow b)$	$\delta(a \leftrightarrow b)$
W	W	F	W	W	W	W
W	F	F	W	F	F	F
F	W	W	W	F	W	F
F	F	W	F	F	W	W

Der Wahrheitswert eines zulässigen Ausdrucks läßt sich mit Hilfe dieser Definition bei vorgegebener Belegung der Aussagevariablen formal berechnen, wobei es für einen n-stelligen zulässigen Ausdruck $A(x_1,\cdots,x_n)$ in der klassischen (zweiwertigen) Logik genau 2^n mögliche Belegungen gibt.

Beispiel 1.6

Es sei $A(a,b,c) = \neg(\neg a \to b) \wedge (c \vee b)$ ein zulässiger Ausdruck. Für die Belegung $\delta(a) = W, \delta(b) = F, \delta(c) = W$ ergibt sich für $A(a,b,c)$ folgender Wahrheitswert:

$$\begin{aligned} \delta(A(a,b,c)) &= \neg(\neg W \to F) \wedge (W \vee F) \\ &= \neg(F \to F) \wedge W \\ &= \neg W \wedge W \\ &= F \wedge W \\ &= F \end{aligned}$$

Neben der in Beispiel 1.6 dargestellten Auswertung eines zulässigen Ausdrucks bei vorgegebener Belegung der vorhandenen Aussagevariablen durch schrittweise Berechnung von innen nach außen, kann ein n-stelliger zulässiger Ausdruck durch eine Wahrheitstafel für alle 2^n möglichen Belegungen bewertet werden. Ein komplexer zulässiger Ausdruck wird dabei meist in mehrere zulässige Teilausdrücke getrennt, die sukzessive berechnet werden, wobei auf die Prioritäten der Junktoren geachtet werden muß. So lautet die Wahrheitstafel für den in Beispiel 1.6 angegebenen zulässigen Ausdruck $A(a,b,c)$ mit den Teilausdrücken

$$d := (\neg a \to b) \; und \; e := (c \vee b) \; :$$

a	b	c		$\neg a$	$d = (\neg a \to b)$	$\neg d$	$e := (c \vee b)$	$\neg d \wedge e$
W	W	W		F	W	F	W	F
W	W	F		F	W	F	W	F
W	F	W		F	W	F	W	F
W	F	F		F	W	F	F	F
F	W	W		W	W	F	W	F
F	W	F		W	W	F	W	F
F	F	W		W	F	W	W	W
F	F	F		W	F	W	F	F

Neben zulässigen Ausdrücken, die für einige Belegungen wahr sind, für die übrigen falsch sind, gibt es zulässige Ausdrücke, die für jede Belegung der Aussagevariablen falsch bzw. richtig sind:

Definition 1.23 (konsistent, inkonsistent, Tautologie)

Es sei $A(x_1,\cdots,x_n)$ ein n-stelliger zulässiger Ausdruck.

1. $A(x_1,\cdots,x_n)$ wird erfüllbar oder konsistent genannt, wenn es eine Belegung von Wahrheitswerten für $x_1,\cdots,x_n$ gibt, so daß $\delta(A(x_1,\cdots x_n)) = W$ gilt.
2. Gibt es keine Wahrheitswertebelegung der $x_1,\cdots,x_n$, für die $\delta(A(x_1,\cdots,x_n)) = W$ ist, so wird die Aussage $A(x_1,\cdots,x_n)$ als inkonsistent oder unerfüllbar bezeichnet.
3. Ist die Aussage $A(x_1,\cdots,x_n)$ für jede der 2^n möglichen Wahrheitswertebelegungen wahr, so heißt die Aussage allgemeingültig oder Tautologie.

Beispiele 1.7

1. $A \wedge (\neg A)$ ist unerfüllbar, denn es gilt

$\delta(A)$	$\delta(\neg A)$	$\delta(A \wedge (\neg A))$
W	F	F
W	F	F
F	W	F
F	W	F

2. $A \vee (\neg A)$ ist allgemeingültig, denn es gilt

$\delta(A)$	$\delta(\neg A)$	$\delta(A \vee (\neg A))$
W	F	W
W	F	W
F	W	W
F	W	W

Anstelle von $\delta(a) = W$ schreibt man auch oft vereinfacht $a = W$.

Für einen 2-stelligen zulässigen Ausdruck gibt es genau 4 verschiedene Belegungen und 16 verschiedene Verteilungen von Wahrheitswerten. Allgemein gibt es für einen n-stelligen zulässigen Ausdruck genau 2^n verschiedene Belegungen und 2^{2^n} verschiedene Verteilungen von Wahrheitswerten. Die Menge der n-stelligen zulässigen Ausdrücke ist für festes $n \in IN$ jedoch unendlich groß. Demzufolge gibt es für ein festes n zulässige Ausdrücke, deren Wahrheitswerte für jede Belegung übereinstimmen.

Definition 1.24 (aussagenlogische Äquivalenz)

Es seien $A = A(x_1, \cdots, x_n)$ und $B = B(x_1, \cdots, x_n)$ zwei n-stellige zulässige Ausdrücke. Ist für jede Belegung $(\delta(x_1), \cdots, \delta(x_n))$ der Wahrheitswert von A gleich dem Wahrheitswert von B, also $\delta(A) = \delta(B)$, so ist die Bijunktion $A \leftrightarrow B$ allgemeingültig. Man schreibt dann auch

$$A \Leftrightarrow B \qquad (\text{„}A \text{ ist äquivalent zu } B\text{“})$$

und nennt die Beziehung zwischen A und B eine aussagenlogische Äquivalenz.

Das Äquivalenzzeichen „⇔ „ ist ein sogenanntes metasprachliches Symbol, welches zum Ausdruck bringt, daß die Bijunktion A ↔ B allgemeingültig ist, und stellt keine aussagenlogische Verknüpfung dar.

Mit Hilfe von Wahrheitstafeln läßt sich ferner zeigen, daß sich jeder Junktor bereits mit Hilfe der Negation und der Disjunktion aussagenlogisch äquivalent darstellen läßt, d.h. jeder zulässige Ausdruck läßt sich unter ausschließlicher Verwendung dieser beiden Junktoren darstellen. Dies nutzt man z.B. bei der Realisierung von Schaltkreisen aus, da dadurch die Anzahl der notwendigen Gattertypen auf zwei reduziert werden kann. Verknüpfungssymbole wie $\wedge$ oder $\rightarrow$ sind daher für den logischen Kalkül nicht unbedingt notwendig, sie dienen jedoch zur besseren Darstellung bzw. der besseren Verständlichkeit.

Analog zur klassischen Mengentheorie läßt sich auch in der Aussagenlogik eine Reihe von Äquivalenzen mit Hilfe der Wahrheitstafeln beweisen:

Tabelle 1.4 Äquivalenzen für die Aussagenlogik

1.	$A \wedge A \equiv A$ $A \vee A \equiv A$	(Idempotenz)
2.	$A \wedge B \equiv B \wedge A$	(Kommutativität)
3.	$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ $(A \vee B) \vee C \equiv A \vee (B \vee C)$	(Assoziativität)
4.	$A \wedge (A \vee B) \equiv A$ $A \vee (A \wedge B) \equiv A$	(Absorption)
5.	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$	(Distributivität)
6.	$\neg\neg A \equiv A$	(Doppelnegation)
7.	$\neg(A \wedge B) \equiv \neg A \vee \neg B$ $\neg(A \vee B) \equiv \neg A \wedge \neg B$	(de Morgansche Regeln)
8.	$A \vee B \equiv A$ $A \wedge B \equiv B$ } falls A allgemeingültig	(Tautologieregeln)
9.	$A \vee B \equiv B$ $A \wedge B \equiv A$ } falls A unerfüllbar	(Unerfüllbarkeitsregeln)
10.	$A \Rightarrow B \equiv \neg B \Rightarrow \neg A$	(Kontraposition)
11.	$A \Rightarrow B \equiv \neg A \vee B$	(Implikation)
12.	$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$	(Koimplikation)

Eine der wichtigsten Anwendungen der klassischen Aussagenlogik ist die aussagenlogische Folgerung (aussagenlogischer Schluß). Hierbei geht es darum, aus bekannten gültigen Aussagen neue Aussagen herzuleiten:

Definition 1.25 (Aussagenlogische Folgerung)

Es seien $A_1, \cdots, A_m, B_1, \cdots, B_k$ *n-stellige zulässige Ausdrücke. Die* $B_1, \cdots, B_k$ *heißen aussagenlogische Folgerung aus* $A_1, \cdots, A_m$, *wenn mit jeder Belegung* $(\delta(x_1), \cdots, \delta(x_n))$ *für die*

$$\delta(A_1) = \cdots = \delta(A_m) = W \quad (m \geq 1)$$

ist, auch gilt

$$\delta(B_1) = \cdots = \delta(B_k) = W \quad (k \geq 1).$$

Es ist also *B* eine aussagenlogische Folgerung aus den $A_1, \cdots, A_n$, wenn die Subjunktion $A_1 \wedge \cdots \wedge A_n \rightarrow B$ eine Tautologie ist. Hierfür verwendet man auch die Schreibweise $A_1 \wedge \cdots \wedge A_n \Rightarrow B$, („$(A_1 \wedge \cdots \wedge A_n)$ impliziert *B*“) und spricht von einer aussagelogischen *Implikation.*

1.7 Graphen

Graphen stellen eine einfache, leicht verständliche Darstellungsform für die Modellierung von Prozessen dar. Sie bestehen aus Knoten und Kanten.

Knoten sind einfache Objekte, die Namen besitzen und Träger von Werten, Eigenschaften, Situationen usw. sein können.

Kanten sind Verbindungen zwischen den Knoten. Sie können, müssen jedoch nicht, zusätzliche Informationen enthalten.

Formal läßt sich ein Graph definieren durch

Definition 1.26 (Graph)

Ein Graph G ist eine zweistellige Relation auf einer Menge V. V ist die Menge der Knoten (eng. Vertex). Zwei Knoten v_1 und v_2 sind in der Relation G, d. h. $v_1, v_2 \in G$ falls es eine Kante (engl. Edge) zwischen v_1 und v_2 gibt.

Ein Graph G auf V ist somit eine Menge von Paaren der Form (v_1, v_2) mit $v_1, v_2 \in V$ und jede beliebige Teilmenge $G \leq V \times V$ ist ein Graph. Zu seiner Darstellung gibt es verschiedene Möglichkeiten:

1. Darstellung als Menge
   Da ein Graph eine Teilmenge $G \in V \times V$ ist läßt sich G darstellen durch die Zusammenfassung aller Paare (v_1, v_2) zu einer Menge, d. h. $G = \left\{\left(v_1^1, v_2^1\right), \left(v_2^1, v_2^2\right), \ldots, \left(v_1^n, v_2^n\right)\right\}$

2. Graphische Darstellung
   Die Graphische Darstellung ist in vielen Fällen die beliebteste, da sie besonders leicht verständ-

lich ist. Hierbei werden die Knoten des Graphen als Kreise mit ihrem Namen dargestellt, während die Knoten durch Pfeile repräsentiert werden.

3. Matrixdarstellung
Ein Graph kann ferner durch eine boolesche Matrix, die sogenannte Adjuzenzmatrix dargestellt werden. Hierbei entsprechen die Zeilen und Spalten den einzelnen Knoten. Eine „1“ oder ein Name in der Matrix entspricht einem Knoten von dem zur Zeile gehörenden Knoten zu dem Knoten der die Spalte repräsentiert.

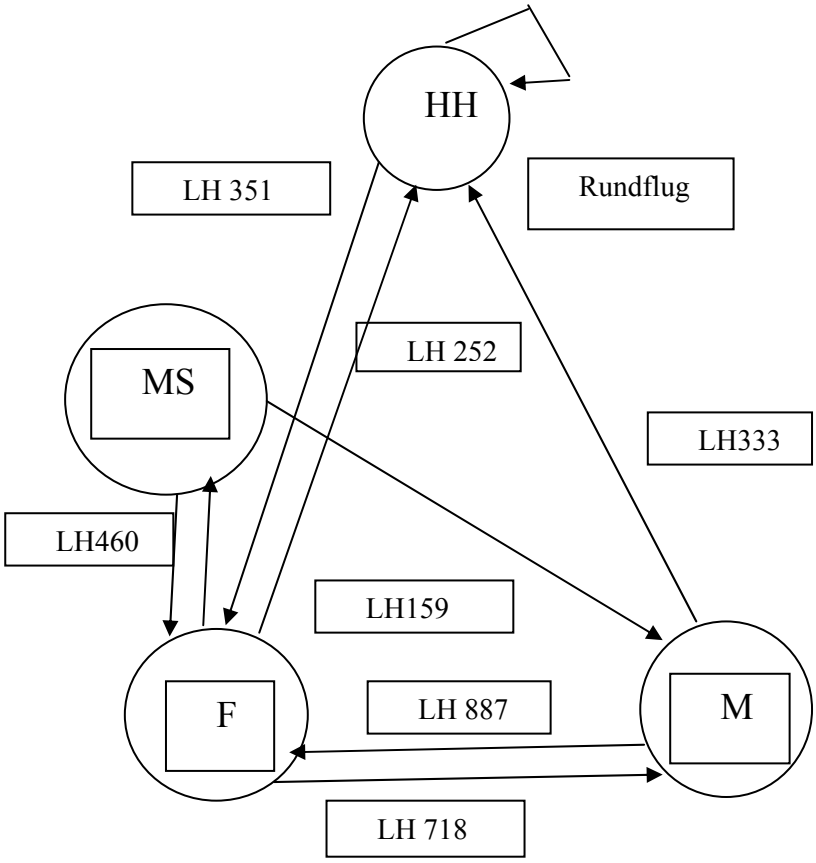
Beispiel 1.8

Gegeben seien die Städte Hamburg, Frankfurt, München und Münster und ferner eine Reihe von Flugverbindungen. Ein Graph, der diese Situation beschreibt, kann z. B. folgendermaßen aussehen:

1. Mengendarstellung

$$
\begin{aligned}
G = \{ & \{(F, HH)/LH \quad 351\}, \\
& \{(HH, F/LH \quad 252\}, \\
& \{(MS, F)/LH \quad 460\}, \\
& \{F, MS)/LH \quad 159\}, \\
& \{(MS, M)/LH \quad 519\} \\
& \{(F, M)/LH \quad 718\}, \\
& \{(M, F)/LH \quad 887\}, \\
& \{(HH, HH)/Rundflug\}, \\
& \{(M, HH)/LH \quad 333\}\}
\end{aligned}
$$

2. Graphische Darstellung



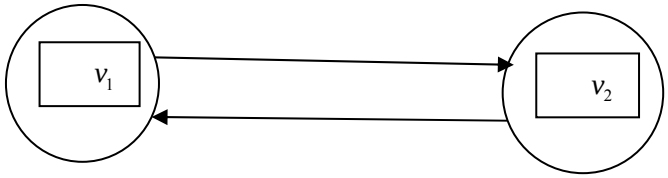
3. Matrixdarstellung

	HH	F	M	MS
HH	Rund-flug	LH252		
F	LH351		LH718	LH159
M	LH333	LH887		
MS		LH 460	LH519	

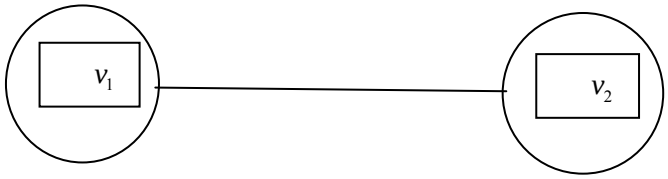
Definition 1.27 (ungerichteter Graph)

Ein Graph $G \subseteq V \times V$ heißt ungerichteter Graph, wenn G symmetrisch ist.

Bei symmetrischen Graphen gehört zu jedem Pfeil von v_1 nach v_2 auch ein Pfeil von v_2 nach v_1:



oft werden in diesem Fall die beiden Pfeile durch eine Linie ersetzt



Wie bereits erwähnt, können auch den Knoten Namen zugeordnet sein:

Definition 1.28 (gewichtetet Graph)

Ist jeder Knote in einem Graphen G ein Name oder Wert zugeordnet, so bezeichnet man G als gewichteten Graphen.

In Beispiel 1.8 handelt es sich somit um einen gewichteten Graphen. anstelle von „gewichteter“ Graph ist auch der Begriff „gewerteter“ Graph gebräulich

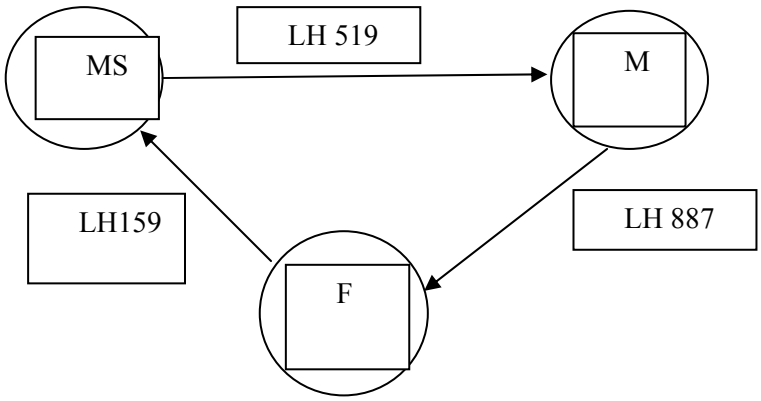
Definition 1.29 (Weg, Pfad)

Ein Weg (oder Pfad) von w nach v in einem Graphen G ist eine Folge

$$w = k_1, k_2, \ldots, k_n = v$$

von Knoten, derart das es jeweils Knoten von k_i nach $k_{i+1}(i = 1, 2, \ldots, n-1)$ gibt. Kommt kein Knoten doppelt vor, so spricht man von einem einfachen Weg. Ein einfacher Weg von x nach x heißt Zyklus.

In Beispiel 1.8 ist z. B. der Weg



ein Zyklus.

Definition 1.30 (zusammenhängender Graph)

Ein ungerichteter Graph G heißt zusammenhängend, wenn es zwischen je zwei verschiedenen Knoten einen Weg gibt.

Ist G nicht zusammenhängend, so zerfällt er in eine Vereinigung zusammenhängender Komponenten, Bäume sind spezielle Graphen. Sie ergeben sich durch

Definition 1.31 (Bäume)

Ein Baum ist ein zusammenhängender, zyklenfreier und ungerichteter Graph.

Definition 1.32 (Teilgraphen, Spannbäume)

Ist $G \subseteq V \times V$ ein Graph und $R \subseteq G$, so ist auch R ein Graph von V und heißt Teilgraph von G auf V. Ist G ein zusammenhängender Graph und R ein zyklenfreier und zusammenhängender Teilgraph von G auf V, denn ist R ein Spannbaum oder erzeugender Baum.

Im Zusammenhang mit Graphen und Bäumen existieren für die Lösung von verschiedenen Fragestellungen eine Reihe von Algorithmen, z. B. zur Bestimmung des kürzesten Weges zwischen zwei Knoten oder zur Traversierung. Auf sie wird nicht näher eingegangen, sie finden sich in den entsprechenden Lehrbüchern zur Theorie der Algorithmen und Datenstrukturen.

2 Automaten, Grammatiken und formale Sprachen

2.1 Einleitung

2.1.1 Mathematischen Maschinen

Bei mathematischen Maschinen handelt es sich um mehr oder weniger abstrakte Modelle von elektronischen Rechnen. Rechnern „Mathematisch“ ist als Gegensatz zu „physikalisch“ zu verstehen. Es handelt sich also nicht um die Realisierung einer physikalischen Maschine, sondern um eine mathematische (theoretische) Konstruktion. Daher hat sich auch für derartige Modelle der Begriff „Automaten“ eingebürgt.

In Analogie zu einem von – Neumann-Rechner besteht ein Automat aus einem Eingabemedium, einer Zentraleinheit (oder Kontrolleinheit) und einem Speichermedium. Generell läßt sich ein Automat daher darstellen durch

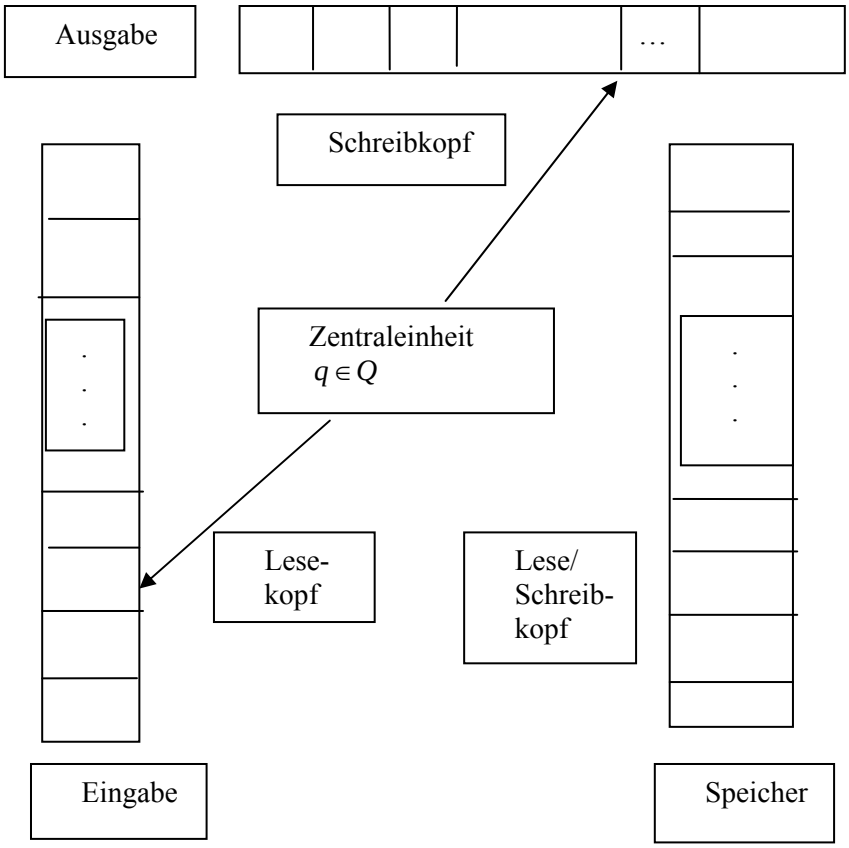


Abb. 2.1 Generelles Modell eines Automaten

Das Eingabe- und Speichermedium kann man als einseitiges unendliches Band mit einzelnen Speicherzellen interpretieren.

Das Verhalten (Übergangsverhalten) eines Automaten hängt ab von

1. dem momentanen Zustand q der Kontrolleinheit
2. dem durch den Lesekopf gelesenen Symbol auf der Eingabe
3. dem durch den Lese/Schreibkopf gelesenen Symbol im Speicher

In einem Schritt kann der Automat

1. den Zustand q der Kontrolleinheit ändern
2. die Position des Lesekopfs auf der Eingabe um eine Stelle verschieben
3. das gelesene Symbol im Speicher verändern
4. die Position des Lese/Schreibkopfes im Speicher um eine Stelle verschieben.

Einige Automatenmodelle besitzen zusätzlich eine Ausgabe. In diesem Fall kann der Automat zusätzlich in jedem Schritt

5. ein Ausgabesymbol erzeugen und ggf. in ein spezielles Ausgabemedium schreiben.

Zu Beginn befindet sich üblicherweise der Lesekopf auf dem ersten Symbol der Eingabe, in einem besonderen Startzustand und der Speicher ist leer oder nur mit einem speziellen Startzustand belegt. Die Anfangskonfiguration (Startkonfiguration) läßt sich somit darstellen durch

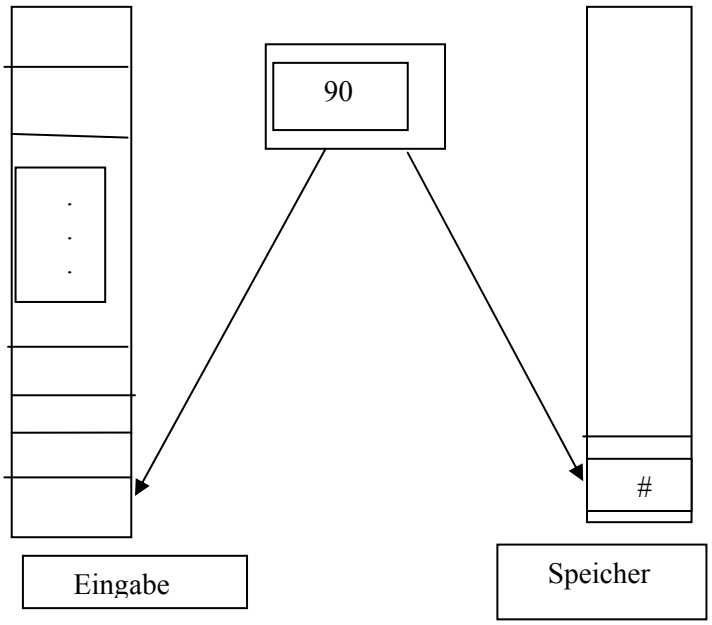


Abb. 2.2 Anfangskonfiguration

Formal läßt sich das allgemeine Modell für einen Automaten beschreiben durch

Definition 2.1 (Allgemeiner Automat)

Ein Automat A ist ein Tupel

$$A = (Q, \Sigma, \Gamma, \mathrm{B}, \delta, q\circ, F, \#)$$

1. Q ist eine endliche Menge von Zuständen
2. Σ ist eine endliche Menge von Eigabesymbolen
3. Γ ist eine endliche Menge von Speichersymbolen
4. B ist eine endliche Menge von Ausgabesymbolen
5. δ ist eine Übergangsfunktion, die das Verhalten des Automaten bestimmt und gegeben als Abbildung von
   $Q \times \Sigma \times \Gamma$ in endliche Teilmengen von
   $Q \times \{l, r, \varepsilon\} \times (\Gamma \times \{l, r, \varepsilon\}) \times B$
6. $q\circ \in Q$ ist ein Startzustand
7. $F \subseteq Q$ ist eine Menge von Endzuständen
8. $\# \in \Gamma$ ist ein Initialisierungssymbol für den Speicher

Sofern nicht anders angegeben, werden

1. die Eingabesymbole mit Kleinbuchstaben vom Anfang des Alphabets
2. Zeichenreihen von Eingabesymbolen mit Kleinbuchstaben vom Endes des Alphabets
3. Kellersymbole mit Großbuchstaben und
4. Zeichenreihen von Kellersymbolen mit griechischen Buchstaben

bezeichnet.

Die momentane Konfiguration des Automaten wird eindeutig beschrieben durch

1. Zustand q der Kontrolleinheit
2. Inhalt der Eingabe
3. Position des Lesekopfes auf der Eingabe
4. Inhalt des Speichers
5. Position des Lese/Schreibkopfes im Speicher
6. Bisher erzeugte Ausgabe

Die Wirkungsweise von

$$S(q,a,M) = (q',x,(N,y),b)$$

läßt sich folgendermaßen beschreiben:

1. Der Automat muß sich im Zustand q befinden, auf der Eingabe a und im Speicher M lesen.
2. Der Automat ändert seine Konfiguration wie folgt:
   - Der Zustand wechselt von q nach q'.
   - Bei $x = l$ bewegt sich der Lesekopf der Eingabe in Richtung des Anfangs der Eingabe (d. h. in den Abb. nach unten). Bei $x = r$ bewegt sich der Lesekopf der Eingabe in Richtung des Endes der Eingabe.
   Bei $x = \varepsilon$ bleibt der Lesekopf der an seiner Position stehen.

   - Im Speicher wird das Symbol M durch das Symbol N überschrieben.

   - Bei $y = l$ bewegt sich der Lese/Schreibkopf des Speichers in Richtung Speicheranfang (d. h. in den Abb. nach „unten").
   Bei $y = r$ bewegt sich der Lese/Schreibkopf des Speichers in Richtung Speicheranfang
   Bei $y = \varepsilon$ bleibt der Lese/Schreibkopf an seiner Position stehen.

   - Der Automat gibt B aus.

Ein einzelner Konfigurationsübergang wird durch das Symbol „$\vdash$ „gekennzeichnet. Mit „ $\vdash^{+}$ “ und „ $\vdash^{*}$ “ bezeichnet man die transitive bzw. transitiv reflexive Hülle von $\vdash$.

Charakteristisch für einen Automaten A ist die von ihm akzeptierte Sprache L(A). Hierzu muß zunächst der Begriff des „Akzeptierens“ erklärt werden Anschaulich ist ein von einem Automaten akzeptiertes Wort diejenige Zeichenreihe, die auf der Eingabe vom Anfang bis zur aktuellen Position des Lesekopfes steht, wenn der Automat einen Endzustand erreicht. Die Menge aller akzeptierten Wörter ist die von A akzeptierte Sprache. Anstelle des Begriffes „Wort“ findet man auch den Begriff „Programm“.

Bei einigen Automatentypen wird anstelle über Endzustände das Akzeptieren eines Wortes auch über das „leeren“ des Speichers definiert.

Die Definition eines allgemeinen Automaten aus Definition 2.1 erlaubt es, durch entsprechende Einschränkungen, verschiedene Automatenmodelle zu definieren. Die wesentlichsten Variationen sind hierbei:

Varianten des Speichermediums

- Kein Speicher
- Speicher ist ein Keller, d. h. der Lese/Schreibkopf steht stets auf dem obersten Eintrag (LIFO).
- Im Speicher dürfen Veränderungen nur beim obersten Eintrag erfolgen, alle Speichereinträge können jedoch gelesen werden. Der Lese/Schreibkopf darf somit nur auf dem obersten Eintrag lesen und schreiben, aber er darf sich auch lesend im übrigen Teil des Speichers bewegen
- Der Speicher ist in Abhängigkeit von der Länge der Eingabe beschränkt
- Im Speicher darf beliebig gelesen und geschrieben werden
- Der Speicher besitzt eine komplexere Struktur, z. B. mehrere parallele Speicher oder ein Speicher mit einer baumartigen Struktur.

Variationen des Eingabemediums

- der Lesekopf darf sich nur in eine Richtung bewegen (1-Weg-Automaten)

- der Lesekopf darf sich in beide Richtungen bewegen (2-Weg-Automaten)

Variationen beim Übergangsautomaten

- Der Automat arbeitet deterministisch, d. h. zu einer Konfiguration existiert höchsten eine Nachfolge Konfiguration
- Der Automat arbeitet nichtdeterministisch

Auf Grund dieser Variationsmöglichkeiten mit ihren vielen Möglichkeiten zu unterschiedlichen Kombinationen. Im Rahmen der Automatentheorie werden diese unterschiedlichen Automatenmodelle hinsichtlich der Mächtigkeit der von ihnen akzeptierten Sprachen und ihrer speziellen Eigenschaften untersucht und verglichen.

2.1.2 Grammatiken

Während Automaten Sprachen erkennen, erzeugen Grammatiken Sprachen. Bei Grammatiken handelt es sich um Systeme, die Zeichenreihen manipulieren, in dem sie Teilzeichenreihen durch andere Teilzeichenreihen ersetzen. Von praktischer Bedeutung sind sie bei der Definition von Programmiersprachen, bei der Formalisierung von Syntaxanalysen und allen Prozessen, bei dem Zeichenreihen verarbeitet werden. So wurden z. B. in den letzten Jahren Grammatiken zur Beschreibung von Dokumentenformaten verwendet und zwar mit Hilfe sogenannter Dokumenttypdefinition (DTD), die von XML-Programmierern (Extensible Mark-up-Language) zum Informationsaustausch im Internet eingesetzt werden.

Die allgemeinste Form der Definition einer Grammatik lautet

Defintion 2.2 (Allgemeine Grammatiken)

Eine allgemeine Grammatik G (nicht eingeschränkte Grammatik, Typ-0-Grammatik) ist ein Quadrupel

$$G=(N,T,P,S)$$

mit

1. N ist eine endliche Menge von nichtterminalen Symbolen oder Variablen. Wegen des letzteren Begriffs

findet man auch den Buchstaben „V“ anstelle von „N“ als Abkürzung

2. T ist eine endliche Menge von terminalen Symbolen mit $N \cap T = \varnothing$.
3. Mit $A := N \cup T$ bezeichnet man den Gesamtzeichenvorrat
4. $\mathrm{P} \subseteq (A^+ \times A^*)$ ist eine Menge von Produktionen oder Regeln
5. $S \in N$ heißt Axiom oder Startsymbol

Für die Schreibweise der Produktionen sind unterschiedliche Schreibweisen üblich. Für $(\alpha, \beta) \in \mathrm{P}$ schreibt man auch

$$\alpha \to \beta \qquad \textit{oder}$$
$$\alpha ::= \beta.$$

In Analogie zu den Automaten lassen sich ausgehend von Def. 2.2 durch Einschränkungen und Modifikationen, unterschiedliche Typen von Grammatiken definieren. Unterschiedliche Einschränkungen beider Form der Produktionen liefern z. B.

Definition 2.3 (Typ - 1,2,3 – Grammatiken)

1. G heißt kontextsensitiv, oder Typ-1-Grammatik, wenn die Anzahl der Produktionen endlich ist und jede Produktion $\rho \in \mathrm{P}$ von der Form

   $$uZv :: u\beta v$$

   ist, mit $u, \beta, v \in A^*$ *und* $Z \in N$.
2. G heißt kontextfrei oder Typ-0-Grammatik, wenn die Anzahl der Produktionen endlich ist, und jede Produktion $p \in \mathrm{P}$ von der Form

   $$Z ::= \beta$$

   ist, mit $Z \in N$ *und* $\beta \in A^*$.
3. G heißt linkslineare Grammatik, wenn die Anzahl der Produktionen endlich ist, und jede Produktion $\rho \in \mathrm{P}$ von der Form

   $$\mathrm{Z}:: = \mathrm{a} \qquad \text{oder}$$

Z:: = Ya

ist, mit $Z,Y \in N$ und

G heißt rechtslineare Grammatik, wenn die Anzahl der Produktionen endlich ist, und jede Produktion $\rho \in \mathrm{P}$ von der Form

Z :: = a oder
Z :: = a Y

ist, mit $Z,Y \in N$ und $a \in T^*$.

Eine links- oder rechtslineare Grammatik wird auch als Typ-3-Grammatik oder reguläre bzw. lineare Grammatik bezeichnet.

Weitere Variationsmöglichkeiten erhält man durch die Definition von deterministischen und nichtdeterministischen Grammatiken bzw. der Erweiterung von der Manipulation einfacher Zeichenreihen zu der Manipulation komplexer Strukturen wie Graphen und Bäumen. Letztere werden jedoch nicht weiter betrachtet. Grammatiken manipulieren Zeichenreihen. Eine Produktion $(\alpha,\beta) \in \mathrm{P}$ enthält die Zeichenreihen α *und* β. Ensprechend kann das Arbeiten mit Produktionen auf zwei verschiedenen Arten erfolgen. Ist eine Zeichenreihe Υ gegeben, so kann man entweder in Υ die Teilzeichenreihe β durch α ersetzen (falls β in Υ als Teilzeichenreihe enthalten ist), oder die Teilzeichenreihe α durch β ersetzen (falls α in Υ als Teilzeichenreihe enthalten ist). Wendet man die Regeln stets nach der ersten Vorgehensweise an, so spricht man von Reduktion, im anderen Fall von Ableitung.

Definition 2.4 (reduzieren)

Sei G=(N,T,P,S) eine Grammatik. Dann kann $\Upsilon \in A^*$ unmittelbar auf Υ' reduziert werden $(\Upsilon \Leftarrow \Upsilon')$ wenn $(\alpha,\beta) \in \mathrm{P}$ *und* $\Upsilon = a\beta b, \Upsilon' = a\alpha b$ *mit* $a,b \in A^*$.

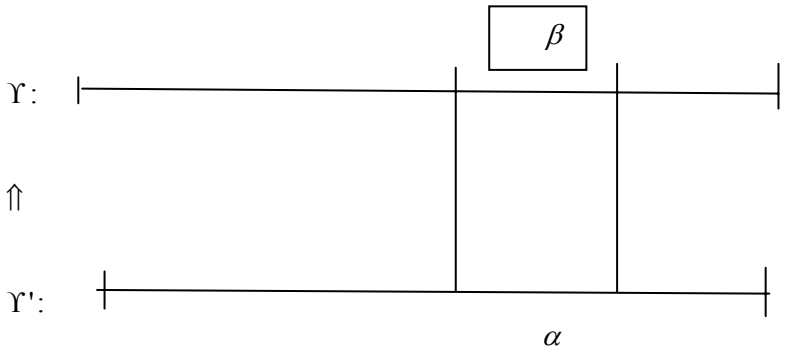
Mit $\overset{+}{\Leftarrow}$ *und* $\overset{*}{\Leftarrow}$ sie die transitive bzw. transitiv reflexive Hülle von $\Leftarrow$ bezeichne Man sagt, daß Υ *auf* Υ' reduziert wird, falls $\Upsilon \overset{*}{\Leftarrow} \Upsilon'$ gilt und daß Υ in mindestens einem Schritt auf Υ' reduziert wird, falls $\Upsilon \overset{+}{\Leftarrow} \Upsilon'$ gilt.

Definition 2.5 (ableiten)

Sei G=(N,T,P,S) eine Grammatik. Dann kann $\Upsilon \in A^*$ unmittelbar aus Υ' abgeleitet werden $(\Upsilon' \Rightarrow \Upsilon)$, wenn $(\alpha, \beta) \in \mathrm{P} \quad \textit{und} \quad \Upsilon = a\beta b, \Upsilon' = a\alpha b \quad \textit{mit} \quad a, b \in A^*$.

Mit $\overset{+}{\Rightarrow}$ *und* $\overset{*}{\Rightarrow}$ sei die transitive bzw. transitiv reflexive Hülle von $\Rightarrow$ bezeichnet. Man sagt, daß Υ aus Υ' abgeleitet wird, falls $\Upsilon' \overset{+}{\Rightarrow} \Upsilon$ gilt und daß Υ in mindestens einem Schritt aus Υ' abgeleitet wird, falls $\Upsilon' \overset{+}{\Rightarrow} \Upsilon$ gilt.

Graphisch lassen sich Reduktion und Ableitung darstellen durch



Es ist klar, daß aus $\Upsilon \Leftarrow \Upsilon'$ stets auch $\Upsilon' \Rightarrow \Upsilon$ folgt und umgekehrt.

Definition 2.6 (erzeugte Sprache)

Die von einer Grammatik G=(N,T,P,S) erzeugt Sprache L(G) ist gegeben durch

$$L(G) = \left\{ w \in T^* \mid w \overset{+}{\Leftarrow} S \right\}$$
$$= \left\{ w \in T^* \mid S \overset{+}{\Rightarrow} u \right\}$$

Die Elemente aus L(G) heißen Sätze. Die Zeichenreihe $v \in A^*$ mit $v \overset{*}{\Leftarrow} S$ heißen Satzformen. Sätze einer Programmiersprache heißen auch Programme oder syntaktisch korrekte Programme (syntaktische Programme)

Aus der Definition 2.6 ergibt sich, daß jeder Satz auch eine Satzform ist, jedoch nicht jede Satzform ein Satz.

Im Zusammenhang mit Programmiersprachen sind die terminalen Symbole diejenigen, die vom Programmierer benutzt werden können.

Um zu überprüfen, ob eine gegebene Zeichenreihe zu L(G) gehört, muß festgestellt werden, ob sie durch sukzessive Anwendung der Produktionen von G auf des Axion von G reduziert werden kann, oder ausgehend von Axion aus diesem mit Hilfe der Produktion von G erzeugt werden kann.

Als Beispiel sei eine Grammatik gegeben, die die Menge aller arithmetischen Ausdrücke erzeugt.

Beispiel 2.1

Die Grammatik G=(N,T,P,S) sei gegeben durch

$$
\begin{aligned}
&N = \{\pi, \varphi, \tau, \alpha\},\\
&T = \{\vee, +, *, \uparrow, (,)\},\\
&S = \alpha,\\
&P = \{\pi ::= \vee / (\alpha),\\
&\qquad \varphi ::= \pi / \varphi \uparrow \pi,\\
&\qquad \tau ::= \varphi / \tau * \varphi,\\
&\qquad \alpha ::= \tau / \alpha + \tau\}
\end{aligned}
$$

Hierbei ist $\alpha ::= \beta / \beta'$ eine abkürzende Schreibweise für die beiden Produktionen $\alpha ::= \beta$ und $\alpha ::= \beta'$. Gemäß Def. 2.2 handelt es sich hiermit um eine kontextfreie Grammatik um zu zeigen, daß die Zeichenreihe

$$V \times V \uparrow V$$

von G erzeugt wird, d. h. zu L(G) gehört, muß sie auf α reduziert werden können:

$$
\begin{array}{lllll}
V & \times & V & \uparrow & V \\
| & & & & \\
\pi & \times & V & \uparrow & V \\
| & & & & \\
\varphi & \times & V & \uparrow & V \\
| & & & & \\
\tau & \times & V & \uparrow & V \\
 & & | & & \\
\tau & \times & \varphi & \uparrow & V \\
 & & & & | \\
\tau & \times & \varphi & \uparrow & \pi \\
\tau & \times & & \varphi & \\
 & | & & & \\
 & \tau & & & \\
 & | & & & \\
 & \alpha & & &
\end{array}
$$

Mit Hilfe eines Kontrollwortes lassen sich die für eine Folge von Reduktions- bzw. Ableitungsschritte benötigten Produktionen fest halten. Hierzu müssen die Produktionen eindeutig numeriert werden.

Beispiel 2.2

Sei die Grammatik G=(N,T,P,S) gegeben durch

N={S}, T={0,1} und
P={ (1) S::=0,
(2) S::=1,
(3) S::=S0,
(4) S::=S1}

Diese Grammatik G erzeugt die Sprache

$$L(G) = \{0,1\}^+ .$$

Die Ableitung des Satzes 0 1 1 0 ist gekennzeichnet durch das Kontrollwort (3,4,4,1)

$$S \overset{3}{\Rightarrow} S0 \overset{4}{\Rightarrow} S10 \overset{4}{\Rightarrow} S110 \overset{1}{\Rightarrow} 0110$$

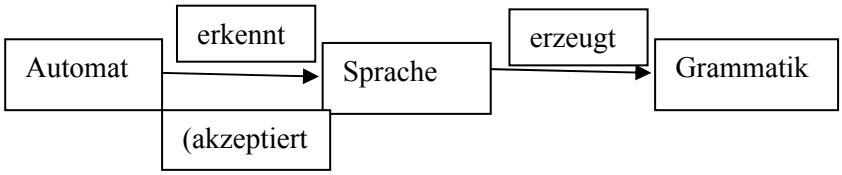
2.1.3 Formale Sprachen

Automaten erkennen Sprachen, Grammatiken erzeugen Sprachen. Derartige Sprachen nennt man formale Sprachen. Sie stellen das Bindeglied zwischen den Automaten und den Grammatiken dar.

Man sagt:

„Ein Automat und eine Grammatik sind äquivalent, genau dann, wenn sie die gleiche Sprache erkennen bzw. erzeugen."

Anschaulich läßt sich eine Äquivalenz darstellen durch



Das Finden und Beweisen derartiger Äquivalenz ist einer der Schwerpunkte der Theorie der formalen Sprachen. Da formale Sprachen Mengen von Zeichenreihen über einem Alphabet sind, stellt sich z. B. die Frage der Abgeschlossenheit einer Menge gegenüber Operationen wie Komplementbildung, Vereinigung usw.

Die Feststellung von Äquivalenz zwischen Automaten, Grammatiken und formalen Sprachen ist nicht nur relevant für deren Charakterisierung, sondern auch wichtig für die Beweisführung. Ist bekannt, daß eine Grammatik G, ein Automat A und eine Sprache L äquivalent sind, so ist es gleich, ob der Beweis über G oder A oder L geführt wird. Das Ergebnis ist stets übertragbar. Man kann daher diejenige Beweisführung auswählen, die Beweistechnisch am einfachsten ist.

2.2 Endliche Automaten

2.2.1 Grundlegende Definitionen

Endliche Automaten sind die einfachsten Automatentypen. Anschaulich handelt es sich um (universelle) Automaten ohne Speichermedium. Graphisch lassen sich somit darstellen durch

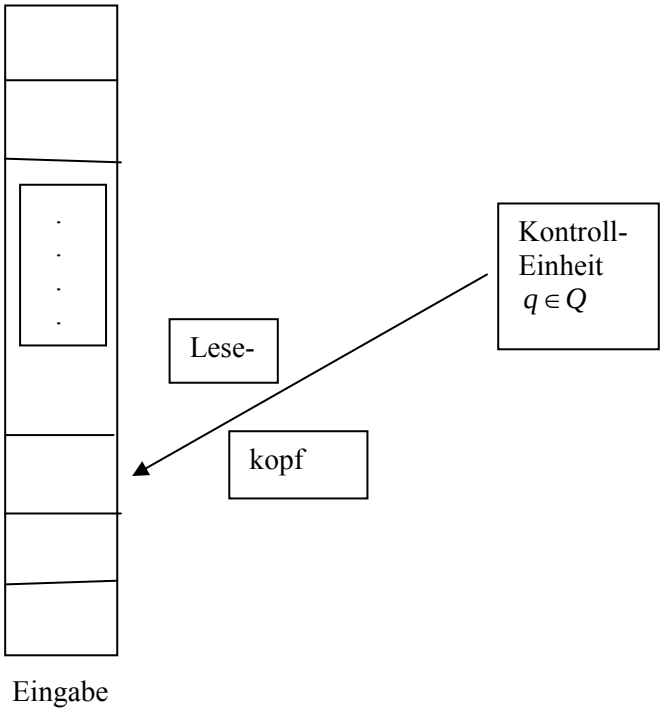


Abb. 2.3 Endliche Automaten

In Abhängigkeit vom aktuell gelesenen Symbol auf der Eingabe und dem aktuellen Zustand der Kontrolleinheit kann der endliche Automat den Zustand ändern und zum nächsten Eingabesymbol übergehen.

Endliche Automaten besitzen in der Praxis ein weites Anwendungsspektrum. Sie sind hervorragend geeignet, Prozesse zu modellieren. Hierbei können es sich sowohl technische Prozesse als auch um Verwaltungspro-

zesse handeln. Sie bilden daher die Grundlagen für die meisten Spezifikations- Sprachen und – Systeme.

Ein weiterer wesentlicher Anwendungsbereich ist die lexikalische Analyse im Rahmen der Übersetzung eines Programms. Endliche Autoamten sind die Grundlagen für den lexikalischen Analysator eines Compilers.

Formal ist ein endlicher Automat (EA) in Analogie zur Definition 2.1 gegeben.

Definition 2.7 (Endlicher Automat)

Ein endlicher Automat EA ist ein Tupel

$$E\,A = (Q, \Sigma, \delta, q_\circ, F)$$

mit

1. Q ist eine endliche Menge von Zuständen
2. Σ ist eine endliche Menge von Eingabesymbolen (Eingabealphabet)
3. δ: Q x $\Sigma \rightarrow$Q ist die Übergangsfunktion
4. $q_\circ \in Q$ ist der Startzustand
5. $F \subseteq Q$ ist eine Menge von Endzuständen

Die Übergangsfunktion δ stellt das „Programm“ des endlichen Automaten dar. Sie gibt den Folgezustand an, den der endliche Automat in Abhängigkeit von seinem aktuellen Zustand und dem aktuellen Eingabesymbol annimmt.

Zur Darstellung von endlichen Automaten existieren im wesentlichen drei Darstellungsformen:

1. Mengendarstellung
   Die Mengendarstellung orientiert sich direkt an der formalen Beschreibung eines endlichen Automaten gemäß Def. 2.7. Sie besteht in der Angabe von Q und Σ als Mengen von Symbolen und der Angabe von δ als Menge von Paaren aus ((Q x Σ), Q).
2. Matrixdarstellung
   Stellt man die Übergangsfunktion δ in Form einer Matrix dar, bei der die Zeilen die Zustände, die

Spalten die Eingabesymbole sind und bei der innerhalb der Matrix der Folgezustand eingetragen ist, so spricht man von einer Matrixdarstellung.

3. Graphische Darstellung
In der graphischen Darstellung wird ein endlicher Automat als Graph dargestellt. Die Zustände sind die Knoten. Existiert für einen Knoten (Zustand) q und eine Eingabe a $\in \Sigma$ eine Regel δ (q, a) = q', so existiert zwischen q und q' eine gerichtete Kante (Pfeil), die mit a markiert ist. Endzustände werden als Doppelkreise dargestellt. Der Startzustand q_o wird als Kreis mit einer unmarkierten Eingangskante (Pfeil), die keinen Ausgangsknoten besitzt dargestellt. Die Menge aller Knoten ergibt die Zustandsmenge Q, die Menge aller Kantenmarkierungen die Menge Σ und die Menge aller Kanten repräsentieren δ.

Beispiel 2.3

1. Der endliche Automat A=(Q, Σ, δ, q_o, F) ist in Mengendarstellung gegeben durch:
Q = $\{q_0, q_1, q_2\}$, $\Sigma = \{a, b, c\}, q_0 = q_0,$
F = $\{q_1, q_2\}$ und
$\delta = \{((q_0 a), q_0), ((q_0 b), q_1), ((q_0, c), q_2)\}$

2. Die Darstellung des gleichen Automaten in Matrixdarstellung unterscheidet sich lediglich in der Form der Angabe von δ:

δ	a	b	c
q_0	q_0	q_1	q_2
q_1	-	-	-
q_2	-	-	-

3. Die graphische Darstellung dieses Automaten ist

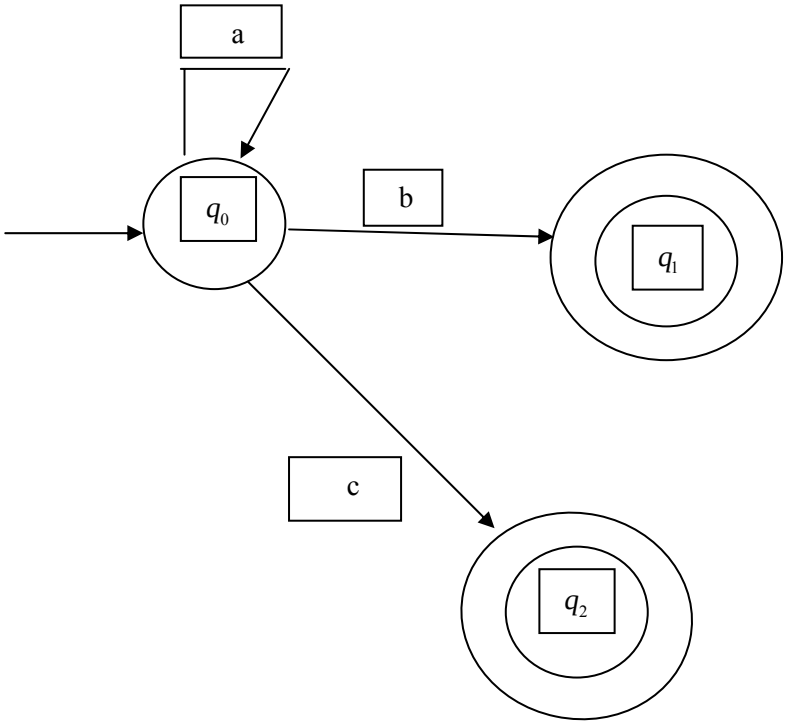


Abb. 2.4 Graphische Darstellung eines endlichen Automaten

In der Startkonfiguration befindet sich der endliche Automat im Zustand q_0 und liest das erste Symbol der Eingabe.

Ein von einem endlichen Automaten akzeptiertes Wort ist diejenige Zeichenreihe, die auf der Eingabe die gelesen wurde bis der Automat einen Endzustand erreicht. Die von diesem Automaten akzeptierte Sprache ist die Menge aller akzeptierten Worte.

Um die von einem endlichen Automaten akzeptierte Sprache formal definieren zu können, muß die Übergangsfunktion δ erweitert werden zu einer Funktion $\hat{\delta}$. Diese Übergangsfunktion $\hat{\delta}$ ist definiert durch

$$\hat{\delta}: Q \times \Sigma^* \to Q,$$

d. h. sie kann auf einen Zustand und eine Zeichenreihe anstatt nur auf einen Zustand und in einzelnes gelesenes Eingabesymbol angewandt werden, und ist induktiv gegeben durch

1. $\hat{\delta}(q, \varepsilon) = q$
2. $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a) \forall q \in Q, w \in \Sigma^*, a \in \Sigma$

Hierbei besagt 1., daß der endliche Automat seinen Zustand nicht ändern kann, ohne ein Eingabesymbol zu lesen, und 2., daß sich der Zustand nach dem Verarbeiten (Lesen) einer nichtleeren Eingabe – Zeichenreihe w a dadurch ergibt, daß zunächst der Zustand $q' = \hat{\delta}(q, w)$ bestimmt, und danach δ (q', a) bestimmt wird.

Betrachtet man die graphische Darstellung eines endlichen Automaten, so bedeutet

$$\hat{\delta}(q, wa) = q',$$

daß es einen Pfad vom Konten q zum Knoten q' gibt, der mit *wa* markiert ist.

Da

$$\hat{\delta}(q, a) = \delta\left(\hat{\delta}(q, \varepsilon), a\right) = \delta(q, a)$$

gilt, stimmen δ und $\hat{\delta}$ für gleiche Argumente überein. Daher wird aus Gründen der Vereinfachung im folgenden im Regelfall nur δ anstatt $\hat{\delta}$ benutzt.

Definition 2.8 (Akzeptierte Sprache)

Eine Zeichenreihe $w \in \Sigma^*$ wird von einem endlichen Automaten E A akzeptiert, falls

$$\hat{\delta}(q_0 w) = q' \in F.$$

In diesem Fall heißt w akzeptiertes Wort.

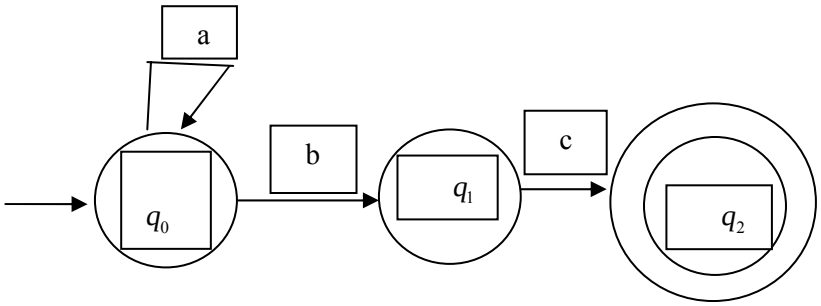
Die von E A akzeptierte Sprache L(EA) ist gegeben durch

$$L(EA) = \left\{ w \mid \hat{\delta}(q_0, w) = q' \in F \right\}$$

Die von endlichen Automaten akzeptierten Sprachen nennt man reguläre Sprachen.

Beispiel 2.4

1. Die akzeptierte Sprache für den Automaten EA aus Beispiel 2.3 ist
   L(EA)={ a*b, a*c }
2. Der Automat aus 1. sei wie folgt zu Automat EA' modifiziert:



Die Sprache L(EA') ist gegeben durch (EA')={bc, abc, aabc, aa … abc}
= {a* b c}

Für EA' gilt offensichtlich

$$\delta(q_0,\varepsilon)=q_0,\hat{\delta}(q_1,\varepsilon)=q_1,\hat{\delta}(q_1,c)=q_2,$$

und

$$\hat{\delta}(q_0,\varepsilon)=q_0,\hat{\delta}(q_1,\varepsilon)=q_1,\hat{\delta}(q_1,c)=q_2,$$
$$\hat{\delta}(q_0,a)=q_0,\hat{\delta}(q_0,b)=q_1,\hat{\delta}(q_1,c)=q_2,$$
$$\hat{\delta}(q_0,ab)=q_1,\hat{\delta}(q_0,abc)=q_2,$$
$$\hat{\delta}(q_0,bc)=q_2$$

Zu beachten ist, daß von einem Endzustand durchaus noch weitere Kannten ausgehen können.

Beispiel 2.5

In Beispiel 2.4 sei der Automat aus 2. derart modifiziert, daß auch q_1 ein Endzustand ist. Die graphische Darstellung dieses neuen E A'' ergibt sich somit zu

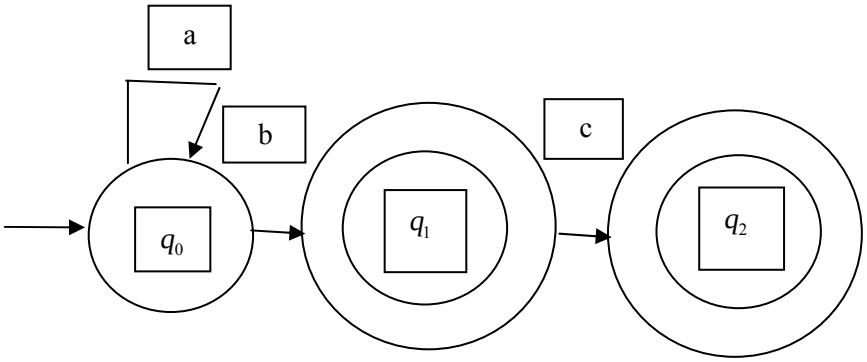


Abb. 2.5 Automat aus Bsp. 2.5

Die von E A'' akzeptierte Sprache L(E A'') ist
L(E A'') = {a * b} ∪ {a * b c}.

Betrachtet man die Def. 2.7 genau und vergleicht sie mit den Ausführungen über die Variationsmöglichkeiten für Automaten aus Kap. 2.1.1, so stellt man fest, daß durch die obige Definition genaugenommen ein 1-Weg-deterministischer endlicher Automat (1 – D – EA) definiert wurde. Es handelt sich um einen 1-Weg-Automaten, da bei jedem Zustandsübergang ein neues Eingabesymbol gelesen wird. Der Automat ist deterministisch, da δ nach Q und nicht in die Potenzmenge von Q abbildet.

Dies ist die klassische Form der Definition eines endlichen Automaten. Vereinzelt findet man in der Literatur auch eine Definition mit zwei speziellen Ausgaben; akzeptiert bzw. nicht akzeptiert. Hierfür ist auch der Begriff „endlicher Akzeptor" gebräuchlich. Da jedoch das Erreichen eines Endzustandes mit „akzeptieren" und das Nicht-Erreichen eines Endzustandes mit „nicht akzeptieren" gleichgesetzt werden kann, sind beide Modelle äquivalent.

In den folgenden Abschnitten werden nun verschiedene Modifikationen (Varianten) des endlichen Automaten untersucht. Im Vordergrund steht

hierbei die Frage, ob die jeweils akzeptierte Sprache gleich, mehr oder weniger mächtig ist als die in Def. 2.8 definierte Sprache.

2.2.2 Nichtdeterministische endliche Automaten.

Die erste „Erweiterung“ besteht im Übergang zu einem nichtdeterministischen Automaten, d. h. δ bildet nicht nach Q sondern in die Potenzmenge von Q ab.

Bei der graphischen Darstellung ist eine nichtdeterministischer endlicher Automat dadurch gekennzeichnet, daß es Knoten gibt, von denen mehrere Kanten ausgehen, die eine Markierung mit demselben Eingabesymbol besitzen. Eine derartige Situation zeigt Abb. 2.6

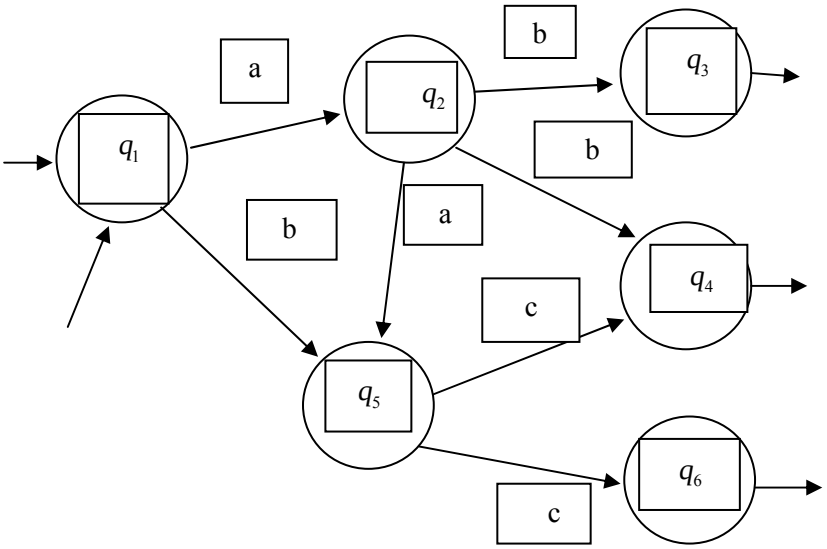


Abb. 2.6 Nichtdeterministischer Endlicher Automat

In Abb. 2.6 tritt ein Nichtdeterminismus an zwei Stellen auf. Von Knoten q_2 gehen zwei Knoten mit der Markierung „b“ aus und von Knoten q_5 gehen zwei Knoten mit der Markierung „c“ aus.

Formal ist die Definition eines nicht deterministischen endlichen Automaten gegeben durch

Definition 2.9 (Nichtdeterministischer endlicher Automat)

Ein nichtdeterministischer endlicher Automat N E A ist ein Tupel

$$NEA = (Q, \Sigma, \delta, q_0, F)$$

mit

1. Q ist eine endliche Menge von Zuständen
2. Σ ist eine endliche Menge von Eingabesymbolen (Eingabealphabet)
3. δ:Q x Σ → = (Q) ($\aleph\ 2^Q$) ist die Übergangsfunktion
4. $q_0 \in Q$ ist der Startzustand
5. F ⊆ Q ist eine Menge von Endzuständen

Auch beim N E A läßt sich δ zu einer Funktion

$$\hat{\delta} : Q \times \Sigma^* \rightarrow \wp(Q)$$

analog zum EA erweitern:

1. $\hat{\delta}(q, \varepsilon) = \{q\}$
2. $\hat{\delta}(q, wa) = \{p \mid$ es gibt einen Zustand r in $\hat{\delta}(q, w),$ für den p in δ (r, a) ist$\}$

Sofern dies nicht zu Komplikationen führt, wird jedoch $\hat{\delta}$ durch das Symbol δ ersetzt.

Die von einem NEA akzeptierte Sprache L(NEA) ist analog zum EA über das Erreichen eines Endzustandes akzeptiert.

Es läßt sich zeigen, daß die Klasse der von nichtdeterministischen endlichen Automaten akzeptierten Sprachen identisch mit der Klasse der von (deterministischen) endlichen Automaten ist, d. h. beide Automatentypen sind gleich mächtig. Dies ist keine Existenzaussage, sondern der Beweis hierfür läßt sich konstruktiv führen. Es existiert ein effektives Verfahren, welches jeden nichtdeterministischen endlichen Automaten in einen –

bzgl. der akzeptierten Sprache – äquivalenten (deterministischen) endlichen Automaten überführt.

Bevor dieser Beweis formalisiert wird, soll das Konstruktionsprinzip anschaulich erläutert werden. Hierzu sei der folgende Auszug aus der graphischen Darstellung eines nichtdeterministischen endlichen Automaten NEA gegeben:

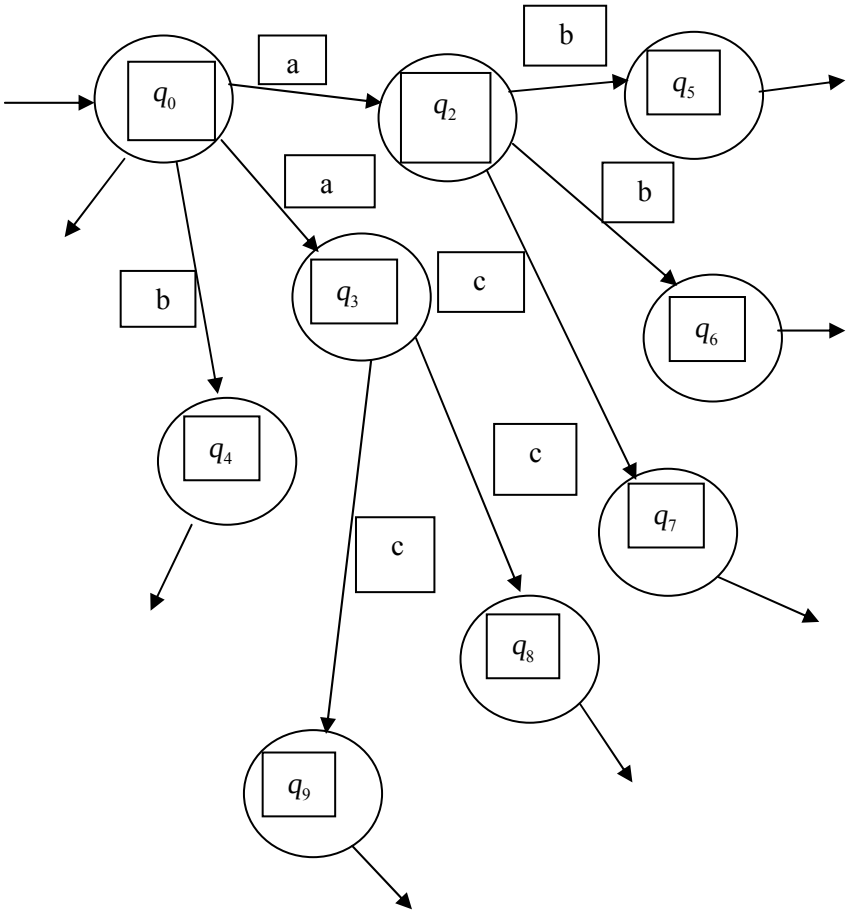


Abb. 2.7 Ausgangsgraph zur Überführung eines NEA in einen EA

Man sieht den Nichtdeterminismus an mehreren Stellen. Von q_1 gehen zwei Kanten mit der Markierung a aus, von q_2 gehen zwei Kanten mit der

Markierung b aus und von q_3 gehen zwei Kanten mit der Markierung c aus. Die Konstruktion des äquivalenten (deterministischen) EA erfolgt durch sukzessive Eliminierung der nicht deterministischen Situationen von „links nach rechts“. Das Prinzip ist hierbei, daß alle Folgeknoten q_i^j zu einem Knoten q_i, die die gleiche Kantenmarkierung besitzen, zu einem neuen Zustand $\left[q_i^1, q_i^2, \ldots\right]$ zusammengefaßt werden.

Bei dem NDA, der Abb. 2.7 dargestellt ist, ergibt sich im ersten Schritt

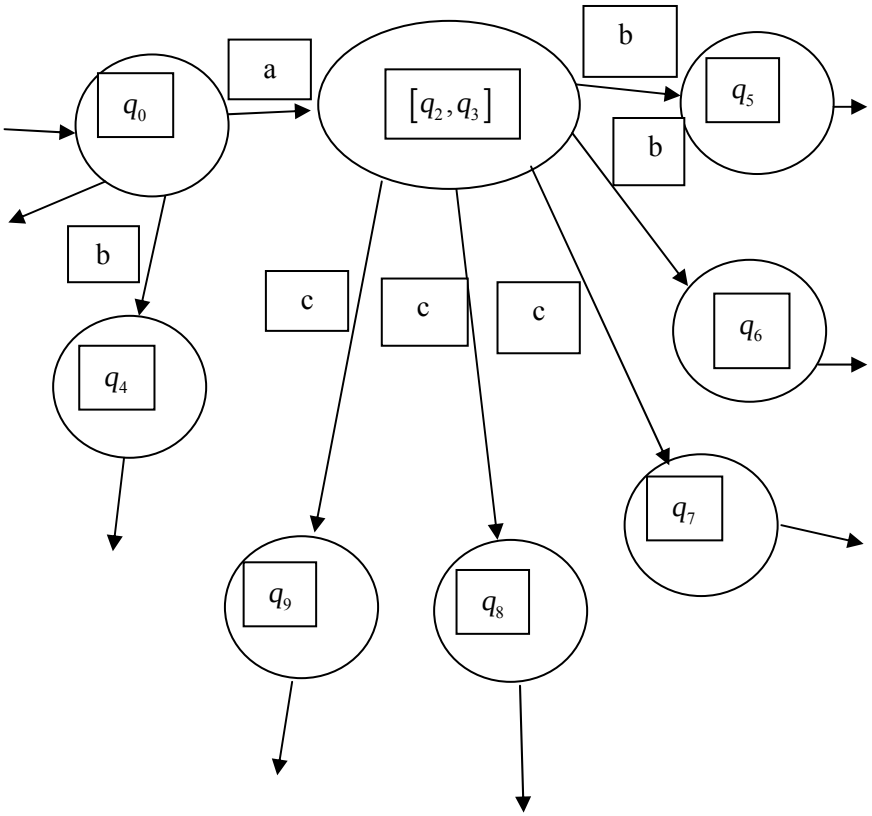


Abb. 2.8 1. Schritt bei der Elimination des Determinismus des Automaten aus Abb. 2.7

Der neu konstruierte Zustand $\left[q_2, q_3\right]$ besitzt zwei abgehende Kanten mit der Markierung b und drei abgehende Kanten mit der Markierung c,

deren Zielknoten jeweils zu einem neuen Knoten zusammengefaßt werden. Das Ergebnis ist in Abb. 2.9 dargestellt

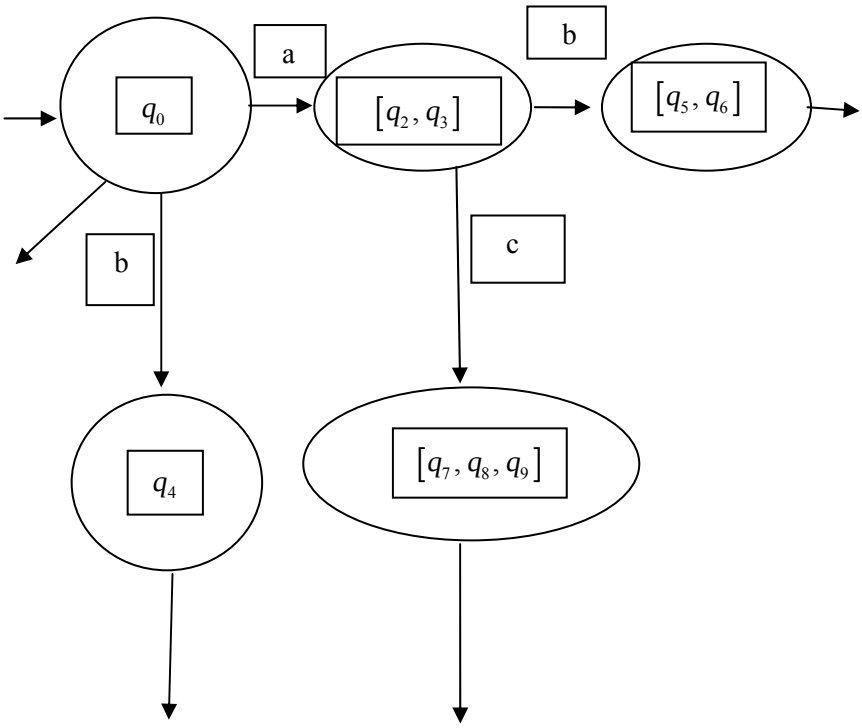


Abb. 2.9 2. Schritt bei der Elimination

Zu begründen ist noch, warum der so konstruierte EA äquivalent zu dem Ausgangs – NEA ist, d. h. die gleiche Sprache akzeptiert. Hierzu wird festgelegt, daß ein Zustand $\left[q_i^1, q_i^2, \ldots\right]$ aus EA genau dann ein Endzustand ist, wenn mindestens es ein q_i^j in diesem Zustand gibt welches Element der Endzustandsmenge von NEA ist.

Sei o. B. d. A. angenommen, daß q_2 ein Endzustand des NEA ist, dann gehört die Zeichenreihe „a“ zu L(NEA). Da $[q_2, q_3]$ auch zur Endzustandsmenge von EA gehört, ist „a“ auch ein Wort aus L(EA).

Ist zusätzlich q_8 ein Endzustand des NEA, dann gehört neben „a“ auch „ac“ zu L(NEA). Da in diesem Fall neben $[q_2, q_3]$ auch $[q_7, q_8, q_9]$ zur

Endzustandsmenge von EA gehören, sind sowohl „a“ als auch „ac“ auch Worte aus L(EA).

Dies sei im folgenden formalisiert.

Satz 2.1

Sei ein NEA mit der Sprache L(NEA) gegeben. Dann läßt sich effektiv ein äquivalenter EA konstruieren mit L(EA)=L(NEA).

Beweis

Sei der nichtdeterministische endliche Automat NEA = (Q, Σ, δ, q_0 , F) gegeben. Der zugehörige äquivalente (deterministische) endliche Automat EA ist gegeben durch

$$EA = \left(Q', \Sigma, \delta', q_0', F'\right)$$

mit

1. Q’ = $\wp$(Q), d. h. die Zustandsmenge von EA ergibt sich als die Potenzmenge von Q (Menge aller Teilmengen von Q). Die Elemente aus Q’ sind mit $[q_1, q_2, \ldots, q_i]$ bezeichnet, wobei $q_1, q_2, \ldots, q_i \in Q$ sind.
2. F’ ist die Menge aller Zustände aus Q’, die einen Zustand aus F enthalten.
3. $q_0' = [q_0]$
4. $\delta'[q_1, q_2, \ldots, q_i], a] = [p_1, p_2, \ldots, p_j]$ genau dann, wenn $\{p_1, p_2, \ldots, p_j\} = \bigcup_{l=1,\ldots,i} \delta(q_l, a)$ ist.

Punkt 4 besagt folgendes:

Durch Anwendung von δ auf jedem Zustand aus $q_1, q_2, \ldots, q_i$ und die Eingabe a und anschließende Bildung der Vereinigung erhält man eine neue Menge von Zuständen $p_1, p_2, \ldots, p_j$. Diese neue Menge wird in Q’

durch den Zustand $\left[p_1, p_2, \ldots, p_j \right]$ repräsentiert. Der Wert von $\delta'\left(\left[q_1, q_2, \ldots, q_i\right], a\right)$ ist genau dieser eine Zustand $\left[p_1, p_2, \ldots, p_j \right]$.

Um den Beweis zu Vervollständigen, wird durch Induktion über die Länge einer Eingabe-Zeichenkette x gezeigt, daß gilt

$\delta'\left(q_0', x\right) = \left[q_1, q_2, \ldots, q_i\right]$ genau dann, wenn $\delta\left(q_0, x\right) = \left\{q_1, q_2, \ldots, q_i\right\}$

Für |x| = 0 ist der Induktionsanfang klar, da $q_0' = \left[q_0\right]$ und x = ε gilt.

Für den Induktionsschritt wird angenommen, daß der Satz für |x| ≤ m richtig ist.

Sei xa eine Zeichenkette aus Σ der Länge m + 1 und a ∈ Σ. Dann gilt

$$\delta'\left(q_0', xa\right) = \delta'\left(\delta'\left(q_0', x\right), a\right).$$

Gemäß der Induktionsannahme gilt

$$\delta'\left(q_0', x\right) = \left[p_1, p_2, \ldots, p_j \right]$$

genau denn, wenn

$$\delta\left(q_0, x\right) = \left\{ p_1, p_2, \ldots, p_j \right\}$$

Nach der Definition von δ′ gilt jedoch

$$\delta'\left(\left[p_1, p_2, \ldots, p_j \right], a\right) = \left[r_1, r_2, \ldots, r_k\right]$$

genau dann, wenn

$$\delta\left(\left\{ p_1, p_2, \ldots, p_j \right\}, a\right) = \left\{r_1, r_2, \ldots, r_k\right\}$$

somit gilt

$$\delta'\left(q_0', x_a\right) = \left[r_1, r_2, \ldots, r_k\right]$$

genau dann, wenn

$$\delta(q_0, x_a) = \{r_1, r_2, \ldots, r_k\}.$$

Da $\delta'(q_0', x)$ genau dann ein Zustand in F' ist, wenn $\delta(q_0, x)$ einen Zustand aus F enthält, ist der Satz bewiesen, d. h. L(NEA) = L(EA).

Da nichtdeterministische und deterministische endliche Automaten die gleich Sprache akzeptieren, braucht man zwischen beiden nicht zu unterscheiden.

Bei der Konstruktion im Beweis von Satz 2.1 wurde bei 1. die neue Zustandsmenge Q' als die Potenzmenge von Q konstruiert. Dadurch sind in der Praxis viele dieser Zustande überflüssig, da nicht erreichbar. Es ist daher bei der konkreten Konstruktion sinnvoll, mit dem Anfangszustand $[q_0]$ zu beginnen, und nur dann neue Zustände zum DEA (=EA) hinzuzufügen, wenn sie Ergebnis eines Zustandsübergangs sind, der von einem bereits hinzugeführten Zustand ausgeht.

2.2.3 Endliche Automaten mit ε - Übergängen

Bei der Definition des (deterministischen) endlichen Automaten wurde festgelegt, daß ein Zustandsübergang stets mit dem Lesen eines Eingabesymbols verbunden ist, d. h. bei jedem Zustandsübergang bewegt sich der Lesekopf auf der Eingabe um ein Symbol weiter.

Dieses Konzept kann erweitert werden, in dem man auch Zustandsübergänge zuläßt, bei denen keine Bewegung des Lesekopfes auf der Eingabe erfolgt. Derartige Zustandsübergänge bezeichnet man als ε-Übergänge. Bei der Darstellung von endlichen Automaten als Graphen sind die entsprechenden Kanten mit ε markiert.

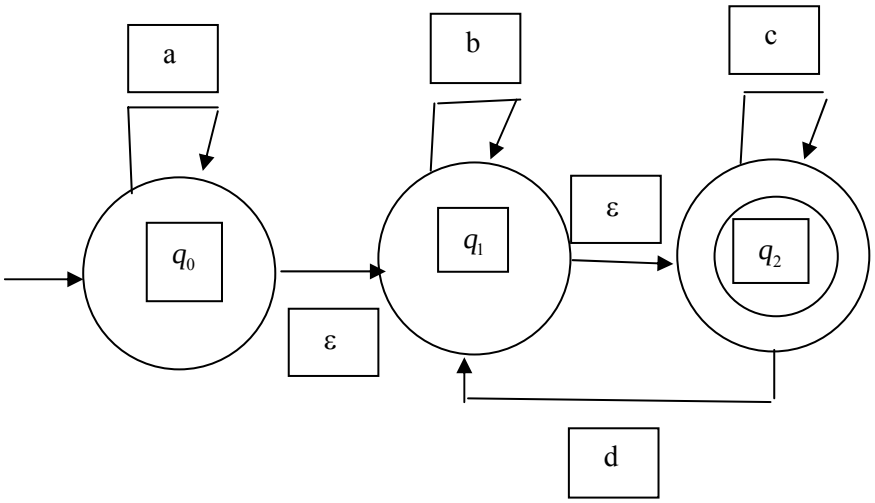
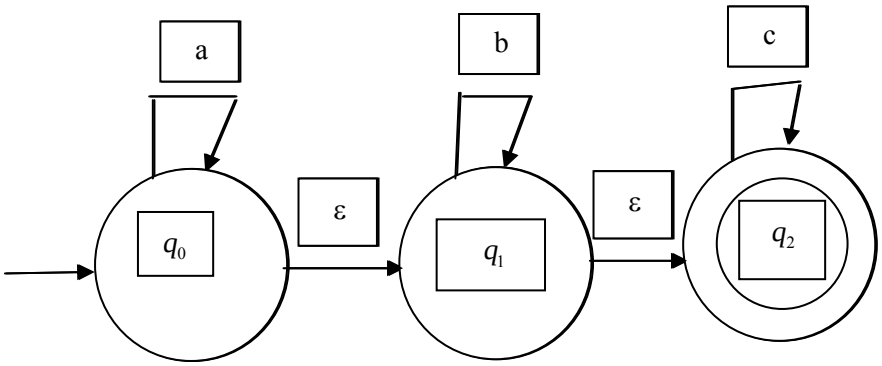


Abb. 2.10 Beispiel für einen endlichen Automaten mit ε - Übergängen

Alle anderen Begriffe und Definitionen werden entsprechend übertragen, d. h. z. B. eine Zeichenkette w wird akzeptiert, wenn es einen mit w markierten Pfad von q_0 in einen Endzustand gibt, wobei auf diesem Pfad auch mit ε markierte Kanten auftreten dürfen. In der Zeichenkette w werden die ε üblicherweise weggelassen.

Beispiel 2.6

Gegeben sei der ε - NEA



Dann ergibt sich

$$\begin{aligned}
\hat{\delta}(q_0, \varepsilon) &= \varepsilon - Hülle(q_0) = \{q_0, q_1, q_2\} \\
\hat{\delta}(q_0, a) &= \varepsilon - Hülle\left(\delta\left(\hat{\delta}(q_0, \varepsilon), a\right)\right) \\
&= \varepsilon - Hülle\left(\delta\left(\{q_0, q_1, q_2\}, a\right)\right) \\
&= \varepsilon - Hülle\left(\delta(q_0, a) \cup \delta(q_1, a), \cup \delta(q_2, a)\right) \\
&= \varepsilon - Hülle\left(\{q_0\} \cup \varnothing \cup \varnothing\right) \\
&= \varepsilon - Hülle\left(\{q_0\}\right) \\
&= \{q_0, q_1, q_2\} \\
\hat{\delta}(q_0, ab) &= \varepsilon - Hülle\left(\delta\left(\hat{\delta}(q_0, a), b\right)\right) \\
&= \varepsilon - Hülle\left(\delta\left(\{q_0, q_1, q_2\}, b\right)\right) \\
&= \varepsilon - Hülle\left(\{q_1\}\right) \\
&= \{q_1, q_2\}
\end{aligned}$$

und

$$L(\varepsilon - NEA) = \{w_1\ w_2\ w_3 \mid w_1 \in \{a\}^*, w_2 \in \{b\}^* \text{ und } w_3 \in \{c\}^*\}$$

Nachdem bereits die Äquivalenz von NEA und EA gezeigt wurde, wird jetzt die Äquivalenz von ε-NEA und NEA gezeigt. Auch hier ist der Beweis konstruktiv

Satz 2.2

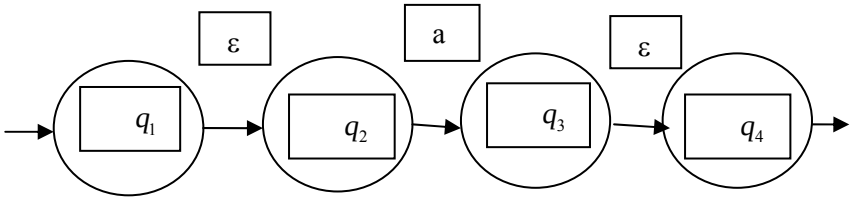
Zu jedem ε-NEA läßt sich effektiv ein NEA konstruieren sodaß gilt

$$L(\varepsilon - NEA) = L(NEA)$$

$\hat{\delta}(q, a)$ enthält alle Zustände, die von q aus über einen Pfad erreichbar sind, der eine einzige mit a markierte Kante besitzt und bei dem alle davor liegender oder nachfolgende Kanten mit ε markiert sind

Beispiel 2.7

Gegeben sei der nachfolgende Teilausschnitt aus dem Graphen eines ε-NEA:



Es gilt

$$\varepsilon \text{ - Hülle } (q_1) = \{q_1, q_2\}$$
$$\varepsilon \text{ - Hülle } (q_2) = \{q_2\}$$
$$\varepsilon \text{ - Hülle } (q_3) = \{q_3, q_4\}$$

und

$\hat{\delta}(q_1, a) = \hat{\delta}(q_1, \varepsilon a) = \varepsilon - Hülle(P) = \varepsilon - Hülle(q_3) = \{q_3, q_4\}$,
$P = \{q_3\}$, *da* $\hat{\delta}(q_1, \varepsilon) = \varepsilon - Hülle(q_1) = \{q_1, q_2\}$ *und* $\delta(q_1, a)$
nicht definiert und $\delta(q_2, a) = q_3$ ist.

Dagegen gilt

$\delta(q_1, a)$ ist undefiniert.

Definition 2.10 (Sprache eines ε-NEA)

Die von einem ε-NEA = (Q, Σ, δ, q_0 , F) akzeptierte Sprache L (ε-NEA) ist gegeben durch L (ε-NEA) = { $w \mid \hat{\delta}(q_0, w)$ enthält einen Zustand aus F} um die Menge aller Knoten p, für die es einen ausschließlich mit ε-Kanten markierten Pfad von q nach p gibt (hierzu gehört auch q selbst). Sei P eine Menge von Zuständen. Dann ist entsprechend die ε-Hülle von P definiert durch

ε-Hülle (P) = $\bigcup_{q \in P} \varepsilon - Hülle(q)$.

Damit läßt sich jetzt die Erweiterung von δ nach $\hat{\delta}$ definieren durch

Definition 2.11 (Zustandsübergänge)

Sei ein ε-NEA gegeben. Dann ist $\hat{\delta}$ induktiv definiert durch

1. $\hat{\delta}$ (q, ε) = ε-Hülle (q)
2. $\hat{\delta}$ (q, wa) = ε-Hülle (P) mit w∈ Σ *, a ∈ Σ und
   P = {p | es ex. Ein r in $\hat{\delta}$ (q, w) und p ∈ δ (r, a)}

Diese Definition läßt sich auf Zustandsmengen R übertragen durch

3. δ (R, a) = $\bigcup_{q \in R} \delta(q, a)$
4. $\hat{\delta}(R, w) = \bigcup_{q \in R} \hat{\delta}(q, w)$

Aufgrund dieser Definitionen müssen δ (q, a) und $\hat{\delta}(q, a)$ nicht unbedingt übereinstimmen:

δ (q, a) enthält nur diejenigen Zustände, die von q aus über eine Kante a erreichbar sind.

Da bereits gezeigt wurde, daß sich jeder nichtdeterministische endliche Automat in einem äquivalenten deterministischen endlichen Automaten überführen läßt, erfolgt die formale Definition der Erweiterung um ε-Übergänge auf der Basis von NEA's. Hierdurch lassen sich die Beweise vereinfachen.

Definition 2.12 (EA's mit ε-Übergängen)

Ein deterministischer endlicher Automat mit ε-Übergängen ε-NEA ist ein Tupel

$$\varepsilon\text{ - NEA} = (Q, \Sigma, \delta, q_0, F)$$

mit

1. Q ist eine endliche Menge von Zuständen
2. Σ ist eine endliche Menge von Eingabesymbolen (Eingabealphabet)
3. $\delta : Q \times (\Sigma \times \{\varepsilon\} \rightarrow \wp(Q)$ ist die Übergangsfunktion
4. $q_0 \in Q$ ist der Startzustand
5. $F \subseteq Q$ ist eine Menge von Endzuständen

Um L(ε - NEA) formal zu definieren muß δ wieder zu $\hat{\delta}$ erweitert werden zu einer Funktion, die von $Q \times \Sigma^*$ in $\wp(Q)$ abbildet. Hierbei enthält $\hat{\delta}$ (q, w) alle Zustände p, für die es einen mit w markierten Pfad von q nach p gibt, der u. U. mit ε markierte Kanten einschließt.

Hierzu wird zunächst der Begriff der ε-Hülle von q eingeführt. Es handelt sich hierbei

?????????

Gegeben sei ein ε - NEA = (Q, Σ, δ, q_0 , F).
Der NEA = (Q, Σ, δ', q_0 , F') ist gegeben durch

1. $$F' = \begin{cases} F \cup \{q_0\} & \textit{falls die } \varepsilon - \textit{Hülle}\,(q_0) \\ F & \textit{einen Zustand aus } F \\ & \textit{sonst} \end{cases}$$

2. δ' (q, a) = $\hat{\delta}$ (q, a) q ∈ Q, Q ∈ Σ

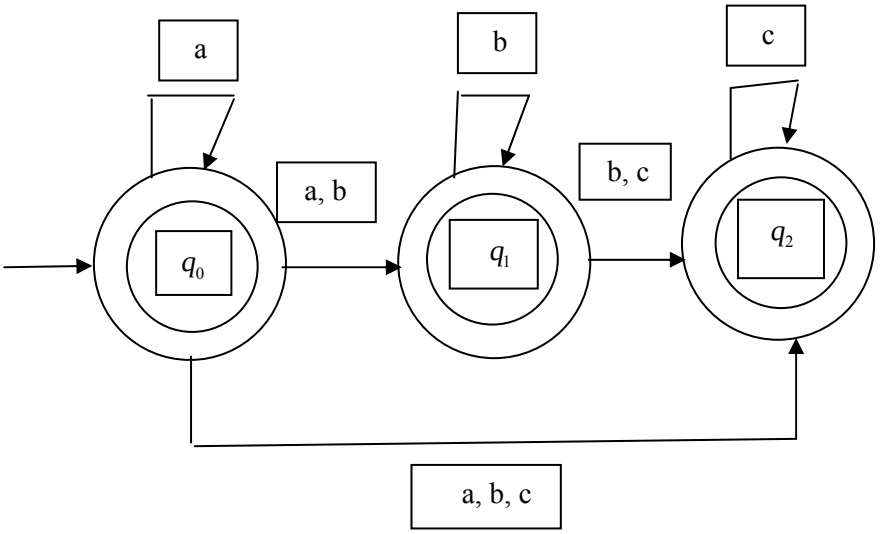
Durch Induktion über |x| , x ∈ Σ*, läßt sich zeigen, daß

$$\delta'(q_0, x) = \hat{\delta}(q_0, x)$$

gilt. Ferner enthält $\delta'(q_0, x)$ genau einen Zustand aus F', wenn $\hat{\delta}(q_0, x)$ einen Zustand aus F enthält. Damit ist der Satz bewiesen.

Beispiel 2.9

Es sei wieder der ε-NEA aus dem Beispiel 2.7 gegeben. Der zugehörige äquivalente NEA ergibt sich zu

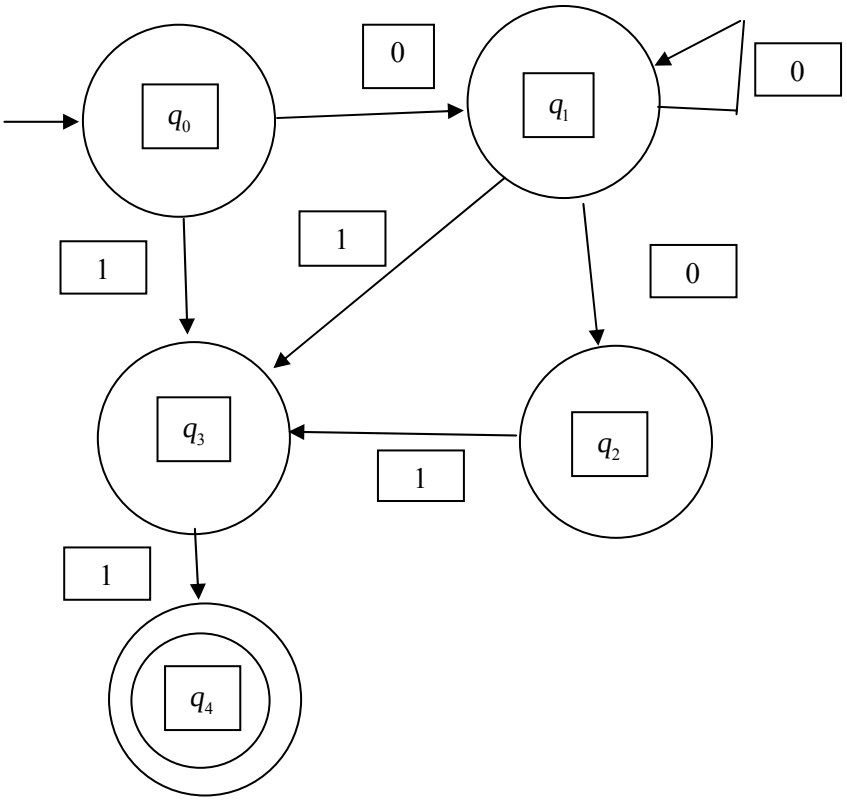


2.2.4 Minimierung endlicher Automaten

Aus der Definition endlicher Automaten ist ersichtlich, daß es mehrere endliche Automaten geben kann, die die gleiche Sprache akzeptieren

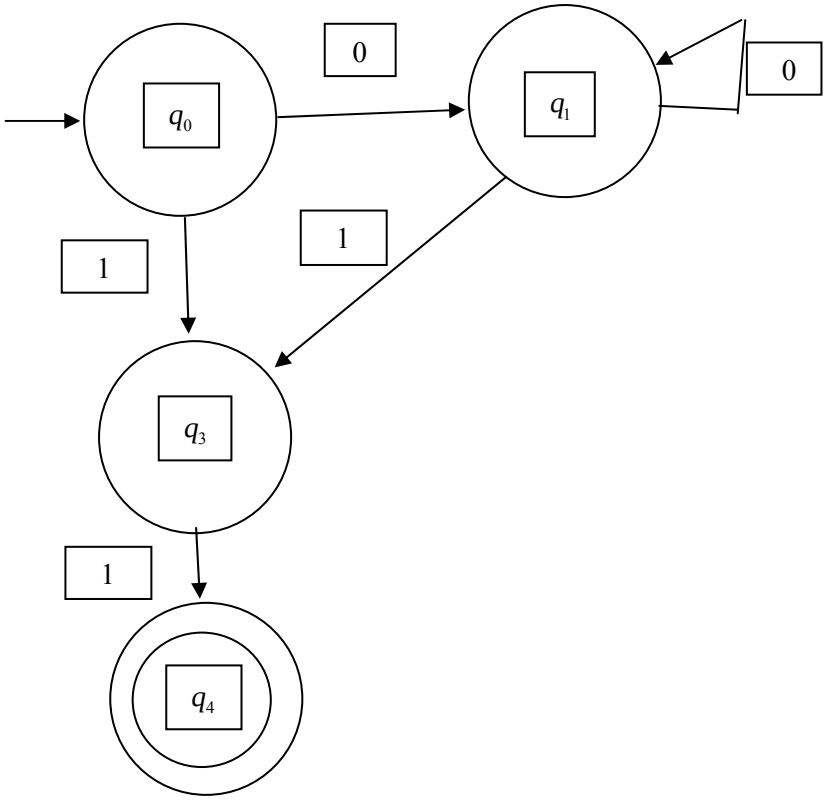
Beispiel 2.10

Gegeben sei der NEA



Die von diesem NEA akzeptierte Sprache lautet
L (NEA) = {11} ∪ {0 0* 11} ∪ {0 0* 0 1 1}

Löscht man den Zustand q_2, so erhält man den Automaten EA′.



EA′ akzeptiert die Sprache

$$L\,(E\,A') = \{11\} \cup \{O\,O^*\,11\}$$

Somit sind beide Sprachen identisch. Der EA besitzt jedoch 5 Zustände, EA′ dagegen nur 4 Zustände. Es stellt sich daher die Frage, ob es generell möglich ist, zu einem gegebenen EA einen (bzgl. der Sprache) äquivalenten endlichen Automaten EA′ zu konstruieren, der eine minimale Anzahl von Zuständen besitzt.

Hierzu existiert tatsächlich ein effektives Verfahren. Die Grundidee des Verfahrens beruht darauf, daß ein EA nicht minimal ist, wenn zwei Zustände q, q′ ∈ Q mit q ≠ q′ existieren, für die gilt

$\hat{\delta}$ (q, x) ∈ F dann, wenn $\hat{\delta}$ (q′, x) ∈ F.

In diesem Fall werden die beiden Zustände q, q′ zu einem neuen Zustand zusammengefaßt. Das Verfahren wird im Zusammenhang mit dem Beweis des nachfolgenden Status vorgestellt.

Das Verfahren der Minimierung hängt eng mit der Frage zusammen ob zwei endliche Automaten äquivalent ind. Wie bereits mehrfach erwähnt, wird die Äquivalenz zweier endlichen Automaten über die von ihnen akzeptierte Sprache definiert, d. h. EA_1 und EA_2 (bzw. NEA_1 und NEA_2) sind äquivalent, wenn $L(EA_1) = L(EA_2)$ (bzw. $L(NEA_1) = L(NEA_2)$).

Um die Äquivalenz zweier Automaten zu überprüfen, soll zunächst die Äquivalenz zweier Zustände berechnet werden. Wegen der Äquivalenz beschränken wir uns auf deterministische endliche Automaten.

Definition 2.13

Zwei Zustände p und q heißen äquivalent, wenn für alle (!) Zeichenreihen w ∈ Σ* gilt:

$$\hat{\delta}(p, w) \in F \text{ genau dann, wenn } \hat{\delta}(q, w) \in F .$$

Zu beachten ist, daß nicht gefordert wird, daß die akzeptierenden Zustände, in die $\hat{\delta}(p, w)$ bzw. $\hat{\delta}(q, w)$ führen, identisch sind. Zwei Zustände p, q heißen unterscheidbar, wenn sie nicht identisch sind, sind zwei Zustände p, q unterscheidbar, so existiert mindestens eine Zeichenreihe w ∈ Σ*, so daß ausgehend von p ein mit w markierter Weg existiert, der in einem Endzustand endet, und ausgehend von q kein derartiger Weg existiert (oder umgekehrt).

Aus dieser Definition folgt unmittelbar

Satz 2.3

Zwei endliche Automaten sind äquivalent, wenn ihre Anfangszustände äquivalent sind.

Gesucht ist noch ein Algorithmus, der effektiv überprüfen kann, ob zwei Zustände äquivalent sind. Das folgende induktive Verfahren, das unter dem Begriff „Table-filling-Algorithmus“, leistet dies Algorithmus „Table-filling-Algorithmus konstruiert wird eine Tabelle T: Q x Q → {ä, nä} Initialiserung.

$$T(q,q') := \begin{cases} nä & falls\ (q \in F \wedge q' \notin F) \\ & oder\ (q \notin F \wedge q' \in F) \\ ä & sonst \end{cases}$$

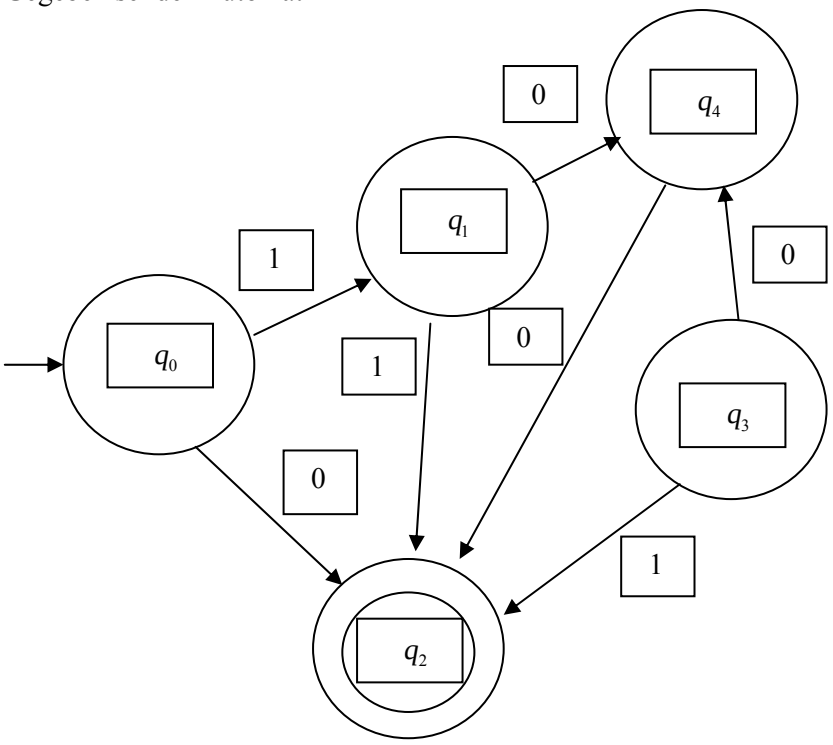
Schleife
repeat

T (q, q') : = nä, falls ein a ∈ Σ ex, mit
T (δ (q, a) , δ(q', a)) = nä

until „keine Veränderung mehr"

Beispiel 2.11

Gegeben sei der Automat



Nach der Initialisierung ergibt sich T zu

T	q_0	q_1	q_2	q_3	q_4
q_0		ä	nä	ä	ä
q_1			nä	ä	ä
q_2				nä	nä
q_3					ä
q_4					

Das Eingabealphabet $\sum$ besteht aus 0 und 1. Für den ersten Durchlauf der Schleife wird zunächst a = 0 und danach w = 1 gesetzt.

$$\delta(q_0,0)=q_2;\quad \delta(q_1,0)=q_4, T(q_2,q_4)=n\ddot{a}$$
$$\Rightarrow\quad T(q_0,q_1)=n\ddot{a}$$
$$\delta(q_2,0)=\phi$$
$$\delta(q_3,0)=q_4, T(q_2,q_4)=n\ddot{a}$$
$$\Rightarrow\quad T(q_0,q_3)=n\ddot{a}$$
$$\delta(q_4,0)=q_2, T(q_2,q_2)=-$$
$$\delta(q_1,0)=q_4:\quad \delta(q_2,0)=\phi$$
$$\delta(q_3,0)=q_4, T(q_4,q_4)=-$$
$$\delta(q_4,0)=q_2, T(q_4,q_2)=n\ddot{a}$$
$$\Rightarrow\quad T(q_1,q_4)=n\ddot{a}$$
$$\delta(q_2,0)=\phi$$
$$\delta(q_3,0)=q_4:\delta(q_4,0)=q_2,\ T(q_4,q_2)=n\ddot{a}$$
$$\Rightarrow T(q_3,q_4)=n\ddot{a}$$
$$\delta(q_0,1)=q_1:\delta(q_1 1)=q_2,\quad T(q_1,q_2)=n\ddot{a}$$
$$\Rightarrow T(q_0,q_1)=n\ddot{a}$$
$$\delta(q_2,1)=\phi$$
$$\delta(q_3,1)=q_2,\quad T(q_1,q_2)=n\ddot{a}$$
$$\Rightarrow T(q_0,q_3)=n\ddot{a}$$
$$\delta(q_4,1)=q_2,\quad T(q_1,q_2)=n\ddot{a}$$
$$=\quad T(q_0,q_4)=n\ddot{a}$$
$$\delta(q_1,1)=q_2:\delta(q_2,1)=\phi$$
$$\delta(q_3,1)=q_2\quad T(q_2,q_2)=-$$
$$\delta(q_4,1)=\phi$$
$$\delta(q_2,1)=\phi$$
$$\delta(q\ ,1)=q_2:\delta(q_4,1)=\phi$$

Somit ergibt sich nach dem ersten Durchlauf durch die Schleife T zu:

T	q_0	q_1	q_2	q_3	q_4
q_0		nä	nä	nä	nä
q_1			nä	ä	nä
q_2				nä	nä
q_3					nä
q_4					

Im nächsten Schleifendurchlauf ändert sich nichts mehr. Aus T ist abzulesen, daß q_1 und q_3 äquivalent sind.

Generell läßt sich zeigen, daß, wenn man für jeden Zustand, eines EA einen Block (Menge) erstellt, der aus q selbst und zusätzlich allen zu q äquivalenten Blöcke eine Partition der Zustandsmenge, d.h.

1. Jeder Zustand gehört genau einem Block an.
2. Alle Elemente eines Blocks sind äquivalent.
3. Keine zwei Zustände, die aus unterschiedlichen Blöcken stammen sind äquivalent

Damit erhält man folgenden Algorithmus zur Minimierung eines (deterministische) EA:

Algorithmus zur Bestimmung des Minimalautomaten.

E A = (Q, $\sum$, δ, q_0 , F)

1. Eliminiere alle Zustände, die vom Startzustand nicht erreichbar sind
2. Ermittle mit Hilfe des Table-filling-Algorithmus alle Paare äquivalenter Zustände
3. Partitioniere Q gemäß der oben beschriebenen Vorgehensweise, d.h. falls (q, p) und (p, r) äquivalent sind, dann sind auch (q, r) äquivalent
4. Die Blocke sind die Zustände des minimalen Automaten
5. Die Übergangsfunktion δ' des minimalen EA ergibt sich folgendermaßen:
   Sei S ein Block der Partionierung des EA. Dann gilt für ein beliebiges a $\in \sum$:
   1. Entweder ex. Ein Block T, so daß für alle Zustände q aus S

gilt, daß δ (q, a) in T liegt,

oder

2. δ (q, a) ist für keinen Zustand q definiert.

Im Fall 1. gilt, daß δ' (S, a) = T.

6. Der Startzustand des minimalen Automaten ist derjenige Block, den den Anfangszustand enthält.
7. Enthält ein Block einen Endzustand von EA, so gehört er zu F'.

Beispiel 2.12

a) Für den Automaten aus Bsp. 2. ergibt sich der Minimalautomat zu

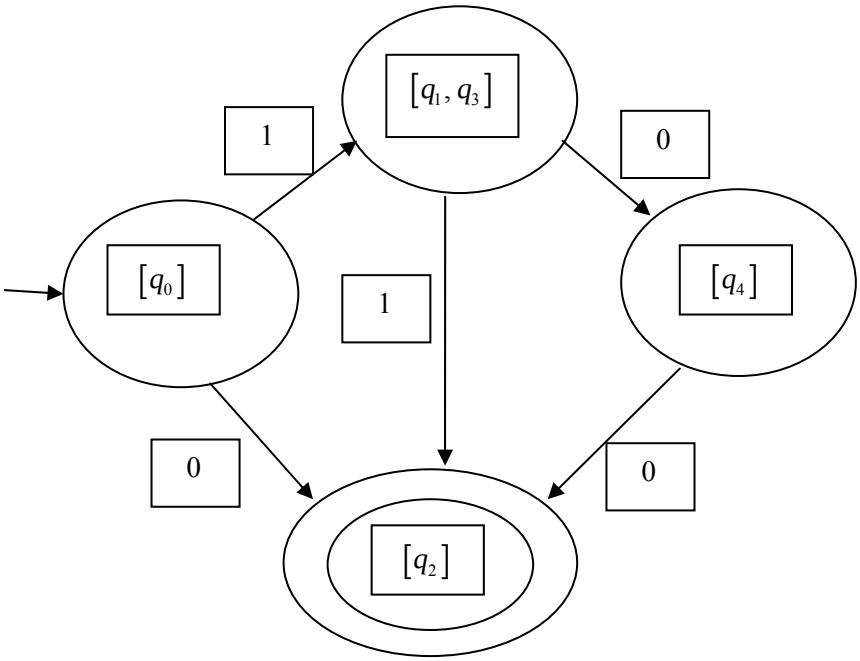


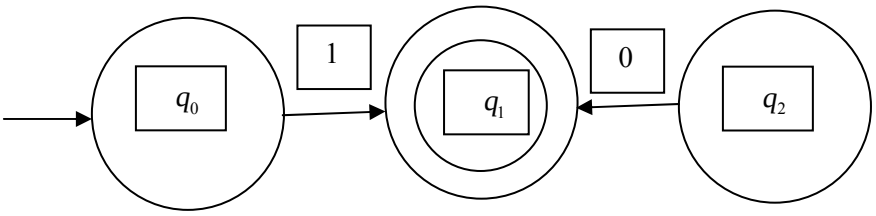
Abb. 2. 8 Minimalautomat zu dem EA aus Bsp. 2.??

da nur q_1 und q_3 äquivalent sind.

Bei diesem Beispiel wird der Zustand q_3 nicht von q_0 aus erreicht, d.h. er hätte eigentlich schon bei Schritt 1 entfernt werden müssen.

Dennoch ist er zusätzlich äquivalent zu q_1.

b) Gegeben sei der Automat



Auch hier wird q_2 nicht von q_0 aus erreicht und muß in Schritt 1. entfernt werden. Da q_2 zu keinem anderen Zustand äquivalent ist wird es bei der Partionierung nicht „verschmolzen“ und wurde daher nicht entfernt werden.

2.2.5 Endliche Automaten mit Ausgaben

Erweitert man das Konzept der endlichen Automaten um die Möglichkeit, auch Ausgaben zu produzieren, so hat man zwei prinzipielle Möglichkeiten:

1. Die Ausgabe erfolgt, wenn sich der Automat in einem Zustand befindet.
2. Die Ausgabe erfolgt bei einem Zustandsübergang.

Der Teilausschnitt eines Graphen zu einem EA in Abb. 2.?? Erläutert dies.

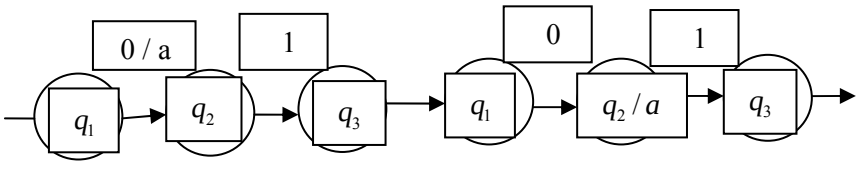


Abb. 2.9 Zwei verschiedene Möglichkeiten für einen EA, Ausgaben zu erzeugen

Bei Abb. 2.9 Wwrd links beim Übergang von q_1 nach q_2 (durch lesen von „0“) die Ausgabe a erzeugt. Graphisch ist dies gekennzeichnet durch „0/a“ an der Kante. Rechts erfolgt die Ausgabe a im Zustand q_2, gekennzeichnet durch q_2 / a im Knoten.

Definiert man endliche Automaten mit Ausgaben gemäß der ersten Möglichkeit, so spricht man von Moore-Maschinen, im zweiten Fall von Mealy-Maschinen.

Definition 2.14 (Moore-Maschinen)

Eine Moore-Maschine ist ein Tupel

$$MO - EA = (Q, \Sigma, \Delta, \delta, \lambda, q_0),$$

Wobei Q, $\sum$, δ und q_0 wie bei einem EA gegeben sind, und

Δ ist eine endliche Menge von Ausgabesymbolen (Ausgabealphabet)

λ ist eine Abb. $\lambda : Q \to \Delta$

Bei dieser Definition ist einiges zu beachten

1. Die Ausgabe von MO-EA als Antwort auf die Eingabe $a_1, \ldots a_n, n \geq 0,$ ist $\lambda(q_0), \lambda(q_1), \ldots, \lambda(q_n),$ wobei $q_0, q_1, \ldots, q_n$ die Zustandsfolge ist, für die $\lambda(q_{i-1}, a_i) = q_i, 1 \leq i \leq n,$ gilt

2. Bereits für die Eingabe ε gibt eine Moore-Maschine die Ausgabe $\lambda(q_0)$ aus, d.h. die erste Ausgabe erfolgt bereits, ohne daß eine Eingabe verarbeitet wurde.
3. In der obigen Definition ist keine Endzustandsmenge enthalten. Der Grund liegt darin, daß bei endlichen Automaten mit Ausgabe nicht die akzeptierte Sprache, sondern die erzeugte Ausgabe im Vordergrund steht. Ein klassischer EA kann als eine Moore-Maschine angesehen werden, für die $\Delta = \{0,1\}$ gilt und ein Zustand q als Endzustand angesehen wird, wenn $\lambda(q) = 1$ gilt.
4. Wegen 3. findet man in der Literatur vereinzelt auch eine Definition mit einer Menge von Endzuständen F. In diesem Fall erhält man in der Menge aller erzeugter Ausgaben eine Teilmenge derjenigen Ausgabezeichenreihen, die beim Akzeptieren eines Wortes erzeugt werden (durch Akzeptieren erzeugte Ausgabe).

Beispiel 2.13

Die Moore-Maschine, die für jede binäre Zeichenreihe den Rest modulo 3 ausgibt, wobei die binäre Zeichenreihe als ganze Zahl interpretiert wurde, ist wie folgt gegeben:

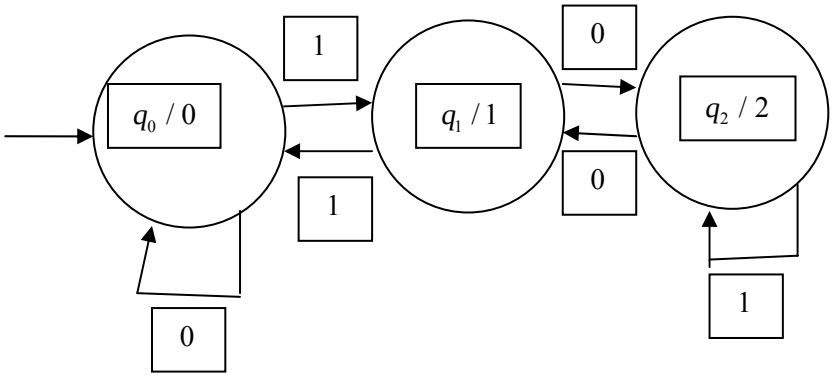


Abb. 2.10 Moore-Maschine aus Bsp. 2.13

Definition 2.15 (Mealy-Maschinen)

Eine Mealy-Maschine ist ein Tupel

$$ME - EA = (Q, \Sigma, \Delta, \delta, \lambda, q_0),$$

Wobei Q, $\sum$, Δ, δ und q_0 wie bei einer Moore-Maschine definiert sind und

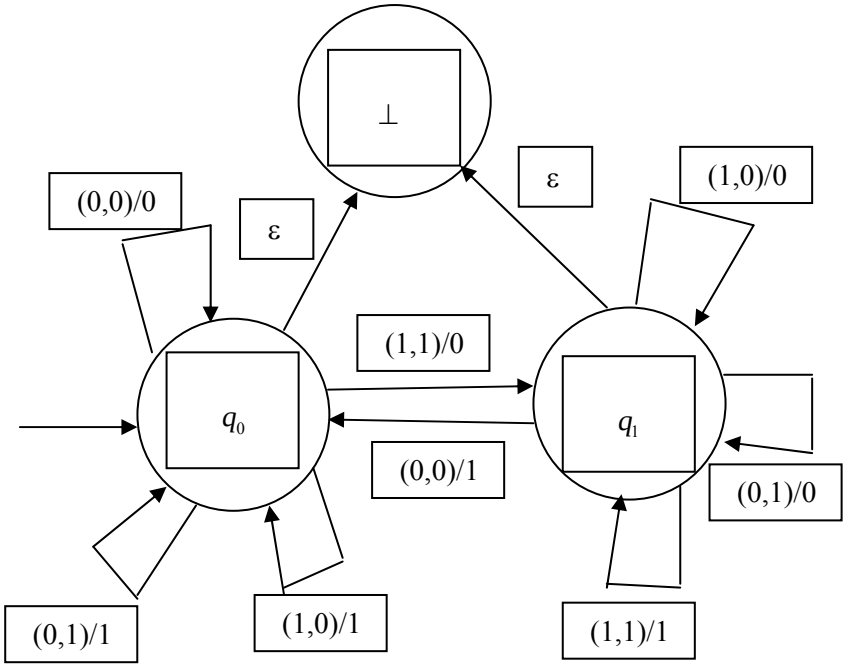
λ ist eine Abb. $\lambda : Q \times \Sigma \to \Delta$

Die Ausgabe von ME-EA bzgl. Der Eingabe $a_1, \ldots, a_n$ ist $\lambda(q_0, a_1), \lambda(q_1, a_2), \ldots, \lambda(q_{n-1}, a_n)$ wobei $q_0, q_1, \ldots, q_n$ diejenige Folge von Zuständen ist, für die $\delta(q_{i-1}, a_i) = q_i, 1 \le i \le n,$ gilt

Beispiel 2.14

Der folgende Mealy-Automat realisiert die Addition zweier Binärzahlen gleicher Länge und gibt das Ergebnis bitweise aus. Die Eingabe ist beendet, sobald ein Leerzeichen (ε) in der Eingabe erscheint (Übergang nach $\perp$):

$\sum$ = {(0, 0), (1, 0), (0, 1), (1, 1)}
Δ = {0, 1}
Q = { $q_0, q_1, \perp$}
δ und λ sind aus dem Übergangsgraphen



ersichtlich.

Vergleicht man die Definitionen der Moore- und Mealy-Maschinen, so ergeben sich sofort zwei Unterschiede bzgl. Des Ausgabeverhaltens:

1. Auf die Eingabe ε liefert eine Mealy-Maschine stets ε, eine Moore-Maschine dagegen a ∈ Δ
2. Bei einem Pfad mit n Kanten erzeugt eine Mealy-Maschine genau n Ausgabesymbole, die Moore-Maschine dagegen n + 1 Ausgabesymbole.

Es stellt sich daher die Frage der Äquivalenz zwischen Moore – und Mealy-Automaten. Wegen des oben skizzierten Unterschieds im Ausgabeverhalten kann es keine exakte Identität gegen. Ein Unterschied ist de facto jedoch nur bei der ersten Ausgabe gegeben. Verlagert man bei einer Moore-Maschine die Ausgabe eines Zustandes in die Eingangskanten des entsprechenden Knotens, so erhält man eine Mealy-Maschine, die bis auf die Ausgabe von q_0, die gleiche Ausgabe erzeugt.

Entsprechend läßt sich eine spezielle Äquivalenz zwischen Moore und Mealy-Automaten definieren.

Definition 2.16 (Ausgabe von Moore- und Mealy-Maschinen)

Sei M eine Mealy- oder Moore-Maschine. Die Ausgabe $T_M(w)$ für eine Eingabe-Zeichenreihe w$\in \Sigma^*$ ist die Ausgabe, die von M infolge der Eingabe w erzeugt wird.

Sei ME eine Mealy-Maschine und MO eine Moore-Maschine, dann können T_{ME} und T_{MO} nie eine exakt identische Ausgabefunktion definierten.

Definition 2.17 (Äquivalenz zwischen Moore- und Mealy-Maschinen)

Eine Mealy-Maschine und eine Moore-Maschine sind äquivalent, wenn

$$b\, T_{ME}(w) = T_{MO}(w)$$

für alle Eingaben w $\in \Delta *$ und b die Ausgabe von MO in Anfangszustand q_0 ist.

Satz 2.4

Wenn $Mo = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ eine Moore-Maschine ist, dann läßt sich effektiv eine (gem. Gef. 2.??) äquivalente Mealy-Maschine ME konstruieren.

Beweis:

Die zu konstruierende ME unterscheidet sich nur durch λ'.
λ' ist gegeben durch

$$\lambda'(q,a) = \lambda(\delta(q,a)) \text{ für alle q} \in \text{Q, a} \in \sum$$

Satz 2.5

Wenn ME = (Q, $\sum$, Δ, δ, λ, q_0) eine Mealy-Maschine ist, dann läßt sich effektiv eine (gem. Def. 2.??) äquivalente Moore-Maschine MO konstruieren.

Beweis

Die zu konstruierende MO ist gegeben durch

$$MO = (Q', \sum, \Delta, \delta', \lambda', q_0')$$

Mit

$$q_0' = [q_0, b_0], b_0 \text{ beliebig aus } \Delta$$
$$Q' = \{[q,b] \mid q \in Q, b \in \Delta\}$$
$$\lambda' = ([q,b]) = b$$

Somit ist die zweite Komponente eines Zustands [q, b] $\in$ Q' diejenige Ausgabe, die ME bei einem Übergang in den Zustand q liefert. Durch Induktion über n läßt sich zeigen, daß ME genau dann infolge der Eingabe $a_1,\ldots,a_n$ in die Zustände $q_0, q_1,\ldots,q_n$ übergeht und die Ausgabe $b_1, b_2,\ldots,b_n$ liefert, wenn MO in die Zustände $[q_0, b_0],[q_1, b_1],\ldots,[q_n,b_n]$ übergeht und die Ausgabe $b_0, b_1,\ldots,b_n$ liefert.

Damit ist die Äquivalenz der beiden Varianten von endlichen Automaten mit Ausgaben gezeigt. Ergänzend sei vermerkt, daß sich die Ausgabefunktion auch auf $\lambda : Q \rightarrow \Delta \cup \{\varepsilon\}$ bzw. $\lambda : Q \times \sum \rightarrow \Delta \cup \{\varepsilon\}$ ausweiten ließt.

.

Brauer, W., Indermark, K. (1968) Algorithmen rekursive Funktionen und formale Sprachen. Bibliographisches Institut, Mannheim. Hochschultaschenbücher Verlag

Berendt, G. (1989) Mathematische Grundlagen für Informatiker. Diskrete Mathematik, Band 1. Bibliographisches Institut, Zürich. Hochschultaschenbücher Verlag

Bretz, M. (1992) Algorithmen und Berechenbarkeit. Vieweg

Hotz, G., Claus, V. (1966/67) Automatentheorie und formale Sprachen. III. formale Sprachen. Bibliographisches Institut Mannheim. Hochschultaschenbücher Verlag

Hotz, G., Walter, H. (1969) Automatentheorie und formale Sprachen. II. endliche Automaten. Bibliographisches Institut Mannheim. Hochschultaschenbücher Verlag

Kreowski, H.-J. (1991) Logische Grundlagen der Informatik. Handbuch der Informatik. R. Oldenbourg Verlag

Reischuk, K. R. (1990) Einführung in die Komplexitätstheorie. B. G. Teubner, Stuttgart