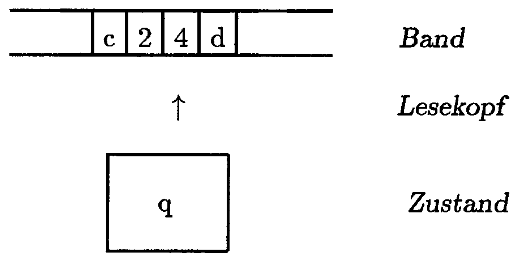
1.2. Endliche Automaten und reguläre Sprachen

<u>Deterministische endliche Automaten</u>

Beispiel 1.2.1: Wir wollen einen Automaten konstruieren, welcher Identifiers in einer Programmiersprache erkennt. Ein Identifier ist dabei eine Folge von Buchstaben und Ziffern, welche mit einem Buchstaben beginnt.

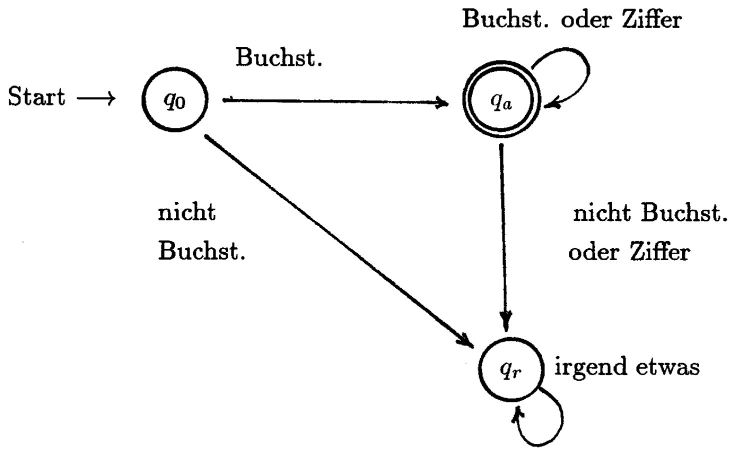
A:



Der Automat A beginnt im Zustand q_0 am Beginn der zu lesenden Folge von Symbolen. In jedem Verarbeitungsschritt liest A das Bandsymbol, auf welches der Lesekopf gerade zeigt, nimmt einen neuen Zustand an und bewegt den Lesekopf um eine Position nach rechts. Liest A im Zustand q_0 einen Buchstaben, so wird der Zustand q_a angenommen, ansonsten der Zustand q_r. Wird im Zustand q_a ein Buchstabe oder eine Ziffer gelesen, so verbleibt A im Zustand q_a. Wird im Zustand q_a etwas anderes als ein Buchstabe oder eine Ziffer gelesen, so wird der Zustand q_r angenommen. Befindet sich A im Zustand q_r, so wird dieser beibehalten, unabhängig vom gelesenen Symbol.

q_a is der sogenannte *akzeptierende* (oder *End*–) Zustand.

Der Überführungsgraph von A sieht wie folgt aus:

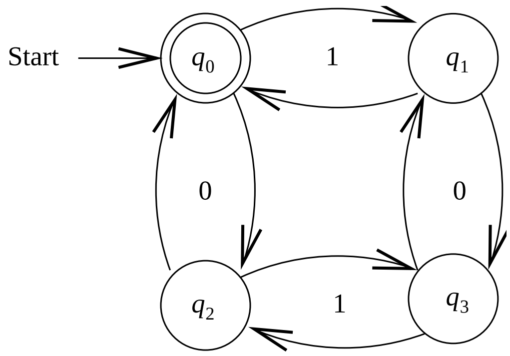


■

Beispiel 1.2.2: Ein Automat, der Folgen von 0 und 1 akzeptiert, in denen die Anzahl der 0en und 1en beide gerade sind.

Möglichkeiten:

#0	#1 gerade	#1 ungerade
gerade	q_0	q_1
ungerade	q_2	q_3



■

Wir wollen nun eine formale Definition für einen endlichen Automaten geben.

Def. 1.2.1: Ein *(deterministischer) endlicher Automat (DEA)* ist ein 5–tupel $M = (Q, \Sigma, \delta, q_0, F)$. Dabei ist

Q eine endliche Menge (von *Zuständen*),

Σ eine endliche Menge (*Eingabealphabet*),

q_0 ein Element von Q (*Startzustand*),

$F \subseteq Q$ (Menge der *Endzustände*),

δ eine Funktion von $Q \times \Sigma$ nach Q (*Überführungsfunktion*).

Im folgenden sprechen wir meist einfach von einem "endlichen Automaten", wenn wir einen DEA meinen.

Die Überführungsfunktion wird folgendermaßen auf Strings beliebiger Länge über Σ (also Elemente von Σ^*) ausgedehnt. Wir bezeichnen die erweiterte Funktion ebenfalls mit δ.

$$\delta(q, \epsilon) = q, \quad (\epsilon \text{ ist der leere String}) \qquad \text{für beliebiges } q \in Q,$$
$$\delta(q, x\alpha) = \delta(\delta(q, x), \alpha), \qquad \text{für bel. } q \in Q, \alpha \in \Sigma, x \in \Sigma^*.$$

Der String $x \in \Sigma^*$ wird von M *akzeptiert* genau dann wenn $\delta(q_0, x) \in F$.

Die von M *akzeptierte Sprache* (Teilmenge von Σ^*) ist definiert als

$$L(M) := \{x \in \Sigma^* \mid x \text{ wird von } M \text{ akzeptiert } \}.$$

Wir sagen auch M *akzeptiert* die Sprache $L(M)$. ■

Reguläre Sprachen

Im folgenden brauchen wir einige Notationen. Sei Σ ein Alphabet, also eine endliche Menge von Symbolen, und seien $L, L_1, L_2 \subseteq \Sigma^*$, also *Sprachen über* Σ.

$\overline{L} := \Sigma^* \setminus L$ (Komplement)
$L_1 \circ L_2 := \{xy \mid x \in L_1, y \in L_2\}$ (Verkettung)
$L^0 := \{\epsilon\}$
$L^i := L \circ L^{i-1}$ für $i \geq 1$
$L^* := \bigcup_{i=0}^{\infty} L^i$ (Kleene Abschluß)
$L^+ := \bigcup_{i=1}^{\infty} L^i$ (positiver Abschluß)

Def. 1.2.2: Sei Σ ein Alphabet. Ein *regulärer Ausdruck* über Σ ist entweder

$\emptyset, \epsilon, \alpha$ (für jedes Symbol $\alpha \in \Sigma$), oder

$(r + s)$, $(r \cdot s)$, (r^*), wobei r und s reguläre Ausdrücke sind.

Die Funktion L transformiert reguläre Ausdrücke über Σ in Untermengen von Σ^*:

$L(\emptyset) := \emptyset,\ L(\epsilon) := \{\epsilon\},\ L(\alpha) := \{\alpha\}$ (für jedes $\alpha \in \Sigma$),

$L(r + s) := L(r) \cup L(s)$,

$L(r \cdot s) := L(r) \circ L(s)$,

$L(r^*) := L(r)^*$.

Ist r ein regulärer Ausdruck, so heißt $L(r)$ die durch r bezeichnete *reguläre Menge* oder *reguläre Sprache.* ■

Streng genommen müßten wir in obiger Definition natürlich $L((r + s))$ schreiben anstatt $L(r + s)$, ebenso für $r \cdot s$ und r^*. Wir gestatten uns aber die abkürzende

Schreibweise, wenn kein Mißverständnis möglich ist. Dabei vereinbaren wir folgende Präzedenzregel: * vor $\cdot$ vor $+$.

Beispiel 1.2.3: Sprache der Identifiers L_I: $\Sigma = \{a, \ldots, z, 0, \ldots, 9\}$,

$$\begin{aligned} L_I = &(\{a\} \cup \cdots \cup \{z\}) \ \circ \ \Sigma^* = \\ &L(\ (a + \cdots + z) \ \cdot \ (a + \cdots + z + 0 + \cdots + 9)^* \ ). \end{aligned}$$

■

Endliche Automaten und reguläre Ausdrücke erzeugen dieselben Sprachen.

Satz 1.2.1: *Sei Σ ein Alphabet, $L \subseteq \Sigma^*$.*

(i) L ist eine reguläre Sprache

genau dann wenn

(ii) L ist die von einem endlichen Automaten akzeptierte Sprache.

Beweis: (i) $\Longrightarrow$ (ii): Sie t ein regulärer Ausdruck über Σ, der die Sprache L bezeichnet.

(a) Zunächst konstruieren wir einen neuen regulären Ausdruck t' über einem neuen Alphabet Σ', so, daß jedes Vorkommen eines $x \in \Sigma$ in t' durch einen eigenen Index gekennzeichnet ist.

Zum Beispiel: $t = a \cdot a + b^* \cdot a \cdot b^*$,
$t' = a_1 \cdot a_2 + b_1^* \cdot a_3 \cdot b_2^*$.

Sei nun $L' := L(t')$.

(b) Ausgehend von t' konstruieren wir nun folgende Mengen:
B' .... mögliche Anfangssymbole von Strings in L',
E' .... mögliche Endsymbole von Strings in L',
S' .... mögliche Paare aufeinanderfolgender Symbole in Strings in L'.

Zum Beispiel: $B' = \{a_1, b_1, a_3\}$,
$E' = \{a_2, a_3, b_2\}$,
$S' = \{(a_1, a_2), (b_1, b_1), (b_1, a_3), (a_3, b_2), (b_2, b_2)\}$.

(c) Ausgehend von B', E', S' konstruieren wir nun einen endlichen Automaten M, sodaß $L = L(M)$, auf folgende Weise:

$Q_0 := \{q_0\}$; (für irgendein Zustandssymbol q_0)
$\delta(q_0,\alpha) := \{\alpha_i \mid \alpha_i \in B'\}$ für alle $\alpha \in \Sigma$;
$k := 0$;
$Q_1 := Q_0 \cup \{\delta(q_0,\alpha) \mid \alpha \in \Sigma\}$;
while $Q_{k+1} \neq Q_k$ **do**
$\{k := k+1$;
$\delta(q,\alpha) := \{\alpha_i \mid$ es gibt ein $\beta_j \in q$ mit $(\beta_j,\alpha_i) \in S'\}$
für alle $q \in Q_k \setminus \{q_0\}$ und $\alpha \in \Sigma$;
$Q_{k+1} := Q_k \cup \{\delta(q,\alpha) \mid q \in Q_k, \alpha \in \Sigma\}$ };
$Q := Q_k$

Diese Prozedur terminiert offensichtlich, da es nur endlich viele Symbole in Σ' gibt.

Es ist klar, daß $x \in L$ genau dann wenn M, ausgehend von q_0 und dem Eingabewort x, einen Zustand erreicht, der ein Element von E' enthält. Wenn wir also setzen

$$q \in F :\Longleftrightarrow q \cap E' \neq \emptyset \quad \text{für } q \in Q \setminus \{q_0\}, \text{ und}$$
$$q_0 \in F :\Longleftrightarrow \epsilon \in L,$$

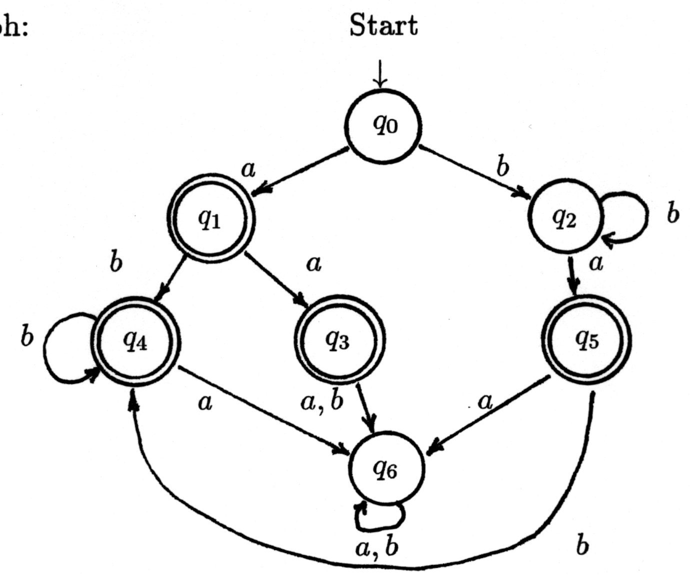
dann sind wir fertig.

Zum Beispiel: $Q_0 = \{q_0\}$,

$$\begin{aligned}
&\delta(q_0,a) = \{a_1,a_3\} =: q_1,\\
&\delta(q_0,b) = \{b_1\} =: q_2,\\
Q_1 &= \{q_0,q_1,q_2\},\\
&\delta(q_1,a) = \{a_2\} =: q_3,\\
&\delta(q_1,b) = \{b_2\} =: q_4,\\
&\delta(q_2,a) = \{a_3\} =: q_5,\\
&\delta(q_2,b) = \{b_1\} =: q_2,\\
Q_2 &= \{q_0,\ldots,q_5\},\\
&\delta(q_3,a) = \emptyset =: q_6,\\
&\delta(q_3,b) = \delta(q_4,a) = \delta(q_5,a) = \emptyset = q_6,\\
&\delta(q_4,b) = \delta(q_5,b) = \{b_2\} = q_4,\\
Q_3 &= \{q_0,\ldots,q_6\},\\
&\delta(q_6,a) = \delta(q_6,b) = \emptyset = q_6,\\
Q_4 &= Q_3.
\end{aligned}$$

Also $Q = Q_3 = \{q_0,\ldots,q_6\}$, und $F = \{q_1,a_3,q_4,q_5\}$.

Überführungsgraph:



(ii) $\Longrightarrow$ (i): Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein endlicher Automat mit $L = L(M)$. Für beliebige Zustände $q, p \in Q$ betrachten wir die Mengen

$$R_{q,p} := \{x \in \Sigma^* \mid \delta(q, x) = p\}.$$

Es gilt

$$L = \bigcup_{p \in F} R_{q_0,p}. \tag{1}$$

Nun sei $Q = \{q_0, q_1, \ldots, q_r\}$. Für beliebige $q, p \in Q$ und $1 \leq j \leq r+1$ definieren wir:

$$R^j_{q,p} := \{x \in R_{q,p} \mid \delta(q, y) \in \{q_0, \ldots, q_{j-1}\} \text{ für jeden Prefix } y \text{ von } x\} \tag{2}$$

und

$$R^0_{q,p} := \begin{cases} \{\alpha \in \Sigma \mid \delta(q, \alpha) = p\}, & \text{falls } q \neq p, \\ \{\alpha \in \Sigma \mid \delta(q, \alpha) = p\} \cup \{\epsilon\}, & \text{falls } q = p. \end{cases} \tag{3}$$

Dann gilt

$$R_{q,p} = R^{r+1}_{q,p}. \tag{4}$$

Für alle q, p ist $R^0_{q,p}$ eine reguläre Menge, da es die Vereinigung endlich vieler Mengen $\{a\}, a \in \Sigma$, oder $\{\epsilon\}$ ist. Weiters gilt

$$R^{j+1}_{q,p} = R^j_{q,p} \ \cup \ R^j_{q,q_j} \circ (R^j_{q_j,q_j})^* \circ R^j_{q_j,p}. \tag{5}$$

Aus (5) sehen wir, daß $R^j_{q,p}$ $(1 \leq j \leq r+1)$ reguläre Mengen sind. Somit ist wegen (4) und (1) auch L regulär. Tatsächlich haben wir sogar einen regulären Ausdruck konstruiert, der L bezeichnet. ∎

Beispiel 1.2.4: Wir betrachten den im ersten Teil des Satzes konstruierten Automaten. Für diesen gilt

$$L = R_{q_0,q_1} \cup R_{q_0,q_3} \cup R_{q_0,q_4} \cup R_{q_0,q_5}.$$

$$\begin{aligned} R_{q_0,q_1} = & R^7_{q_0,q_1} \\ & R^0_{q_0,q_1} = \{a\}, \\ & R^1_{q_0,q_1} = \{a\} \ \cup \ \underbrace{R^0_{q_0,q_1}}_{\{a\}} \circ \underbrace{(R^0_{q_1,q_1})^*}_{\{\epsilon\}^*} \circ \underbrace{R^0_{q_1,q_1}}_{\{\epsilon\}} = \{a\}, \\ & R^2_{q_0,q_1} = \cdots = R^7_{q_0,q_1} = \{a\} \end{aligned}$$

etc. ■

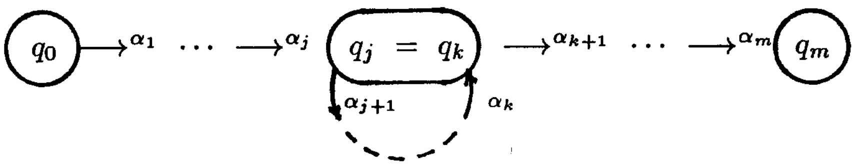
Der folgende Satz eignet sich oft sehr gut dazu zu entscheiden, ob eine vorliegende Sprache regulär ist oder nicht. Vereinfacht gesagt ist eine Sprache nur dann regulär, wenn genügend lange Wörter in ihr aufgebläht werden können ohne die Sprache zu verlassen. Daher kommt auch der Name für diesen Satz.

Satz 1.2.2: (Pumping Lemma) *Sei L eine reguläre Sprache. Dann gibt es eine Konstante $n (\in \mathbb{N})$, sodaß jedes Wort $z \in L$ der Länge mindestens n ($\mid z \mid \geq n$) auf solche Weise zerlegt werden kann in $z = uvw$, daß $\mid uv \mid \leq n$, $\mid v \mid \geq 1$, und für alle $i \geq 0$ auch $uv^iw \in L$. Die Konstante n ist dabei nicht größer als die Anzahl der Zustände des kleinsten Automaten, der L akzeptiert.*

Beweis: Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein endlicher Automat, der L akzeptiert. Sei $n = \mid Q \mid$. Wir betrachten nun ein Wort z, sodaß

$$z = \alpha_1 \cdots \alpha_m \in L \quad \text{und} \quad \mid z \mid = m \geq n.$$

Wenn wir mit q_i denjenigen Zustand bezeichnen, welcher nach Abarbeitung von $\alpha_1 \cdots \alpha_i$ angenommen wird, so müssen offensichtlich zwei solche Zustände identisch sein. Also für gewisse j, k ($0 \leq j < k \leq n$),



z kann also zerlegt werden in $z = uvw$, wobei

$$u = \alpha_1 \cdots \alpha_j, \quad v = \alpha_{j+1} \cdots \alpha_k, \quad w = \alpha_{k+1} \cdots \alpha_m.$$ ■

Beispiel 1.2.5: $L = \{a^m b^m \mid m \in \mathbb{N}\}$ ist keine reguläre Sprache.

Angenommen L wäre regulär. Sei n die Konstante im Pumping Lemma. Wir betrachten das Wort

$$z = a^n b^n,$$

das natürlich in L enthalten ist. Wegen des Pumping Lemmas muß z zerlegbar sein in

$$a^n b^n = uvw,$$

wobei $|uv| \leq n$, $|v| \geq 1$, und $uv^i w \in L$ für jedes i. Somit müßte gelten

$$\underbrace{a^{n_1}}_{u}\ \underbrace{a^{n_2}}_{v}\ \underbrace{a^{n_3} b^n}_{w}, \qquad \text{wobei } n = n_1 + n_2 + n_3,\ n_2 \geq 1.$$

Aber $a^{n_1}(a^{n_2})^2 a^{n_3} b^n$ ist nicht in L. Somit kann L nicht regulär sein. ■

Beispiel 1.2.6: $L = \{0^{i^2} \mid i \in \mathbb{N}\}$ ist keine reguläre Sprache.

Angenommen L wäre regulär. Sei n die Konstante im Pumping Lemma. Wir betrachten das Wort

$$z = 0^{n^2},$$

das natürlich in L enthalten ist. Wegen des Pumping Lemmas muß z zerlegbar sein in

$$0^{n^2} = uvw,$$

wobei $|uv| \leq n$, $|v| \geq 1$, und $uv^i w \in L$ für jedes i. Das ist aber für $i = 2$, also $uv^2 w$, schon nicht der Fall. $n^2 < |uv^2 w| \leq n^2 + n < (n+1)^2$. Somit ist $uv^2 w \notin L$. ■

<u>Nichtdeterministische endliche Automaten</u>

Man kann sich nun aber auch endliche Automaten vorstellen, die bei gegebenem Zustand und Eingabesymbol nicht nun einen einzig möglichen Nachfolgezustand haben, sondern aus verschiedenen (oder auch gar keinen) Möglichkeiten für Nachfolgerzustände wählen können.

Wie soll die Semantik eines solchen "nichtdeterministischen endlichen Automaten" aussehen? Wenn im Zustand p der Automat das Symbol a liest und etwa $\{q_1, q_2\}$ für diesen Fall als mögliche Nachfolgezustände definiert sind, so kann der Automat entweder in den Zustand q_1 oder in den Zustand q_2 übergehen. Der Automat akzeptiert ein Wort $w \in \Sigma^*$, wenn es einen möglichen Berechnungspfad gibt, sodaß der Automat von einem Anfangszustand und Wort w auf dem Band zu einem Endzustand gelangen kann.

Def. 1.2.3: Ein *nichtdeterministischer endlicher Automat (NEA)* ist ein 5–tupel $M = (Q, \Sigma, \delta, S, F)$. Dabei ist

Q eine endliche Menge (von *Zuständen*),

Σ eine endliche Menge (*Eingabealphabet*),

$S \subseteq Q$ (Menge der *Anfangszustände*),

$F \subseteq Q$ (Menge der *Endzustände*),

δ eine Funktion von $Q \times \Sigma$ nach $\mathcal{P}(Q)$, der Potenzmenge von Q (*Überführungsfunktion*).

Die Überführungsfunktion wird folgendermaßen ausgedehnt zu einer Funktion von $Q \times \Sigma^*$ nach $\mathcal{P}(Q)$. Wir bezeichnen die erweiterte Funktion ebenfalls mit δ.

$\delta(q, \epsilon) = \{q\}$, für bel. $q \in Q$,

$\delta(q, x\alpha) = \{p \mid p \in \delta(r, \alpha) \text{ für ein } r \in \delta(q, x)\}$, für bel. $q \in Q, x \in \Sigma^*, \alpha \in \Sigma$.

Weiters dehnen wir δ aus zu einer Funktion von $\mathcal{P}(Q) \times \Sigma^*$ nach $\mathcal{P}(Q)$:

$\delta(P, x) = \bigcup_{p \in P} \delta(p, x)$, für bel. $P \subseteq Q$, $x \in \Sigma^*$.

Der String $x \in \Sigma^*$ wird von M *akzeptiert* genau dann wenn $\delta(S, x) \cap F \neq \emptyset$.

Die von M *akzeptierte Sprache* (Teilmenge von Σ^*) ist definiert als

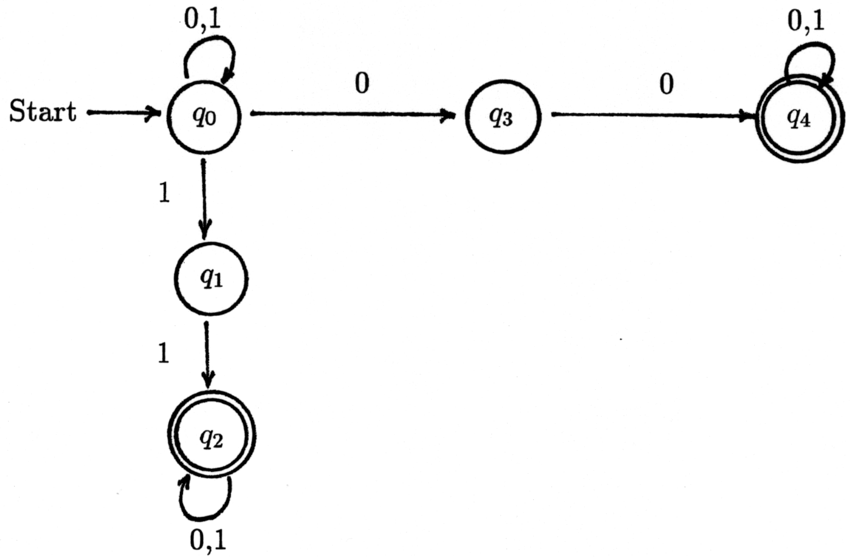
$$L(M) := \{x \in \Sigma^* \mid x \text{ wird von } M \text{ akzeptiert } \}.$$

Wir sagen auch M *akzeptiert* die Sprache $L(M)$. ■

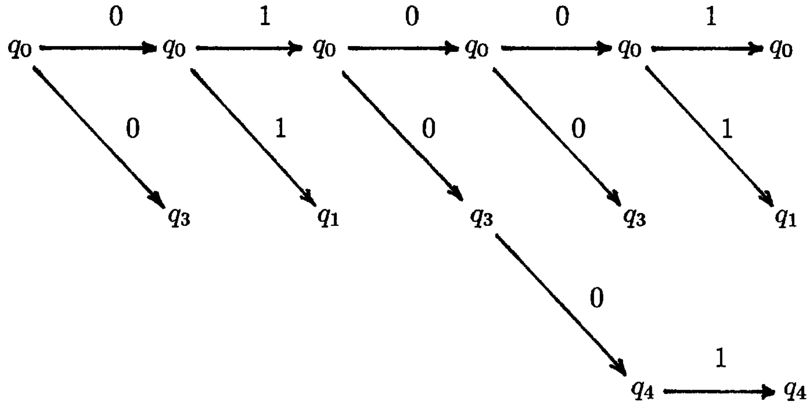
Beispiel 1.2.7: Wir betrachten folgenden NEA M, wobei $Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{0, 1\}, S = \{q_0\}, F = \{q_2, q_4\}$, und die Überführungsfunktion δ durch folgende Tabelle gegeben ist:

	0	1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$

Der Überführungsgraph dieses NEA M sieht folgendermaßen aus:



Wenn der Automat die Möglichkeit hat, wie im Zustand q_0 bei Eingabe 0, verschiedene Nachfolgezustände anzunehmen, so können wir uns vorstellen, daß entsprechend viele Kopien des Automaten hergestellt werden, deren jede eine dieser Möglichkeiten realisiert. Diese Form der Abarbeitung eines Eingabestrings führt zu einem Berechnungsbaum. Wir stellen den Berechnungsbaum für den NEA M bei Eingabe 01001 dar:



Die von M akzeptierte Sprache $L(M)$ besteht aus all denjenigen Wörtern, welche 00 oder 11 enthalten. ∎

Es ist offensichtlich, daß jeder DEA aufgefaßt werden kann als ein NEA, bei dem eben die Menge der möglichen Nachfolgezustände jeweils eine einelementige Menge ist. Somit ist klar, daß jede Sprache L, welche von einem DEA akzeptiert wird, auch von einem NEA akzeptiert werden kann. Interessanterweise gilt auch die Umkehrung.

Satz 1.2.3: *Sei M ein NEA und sei $L = L(M)$. Dann gibt es einen DEA M' mit $L(M') = L$.*

Beweis: Sei $M = (Q, \Sigma, \delta, S, F)$ der gegebene NEA. Wie konstruieren einen DEA M', der die gleiche Sprache akzeptiert, indem wir jede Teilmenge von Q als Zustand von M' auffassen. Alles übrige ergibt sich dann fast von selbst. Wir setzen also

$$M' = (Q', \Sigma, \delta', q_0', F'),$$

wobei

$$\begin{aligned} Q' &= \mathcal{P}(Q) \qquad \text{(Potenzmenge von } Q\text{)},\\ \delta'(q', \alpha) &= \bigcup_{p \in q'} \delta(p, \alpha) \ (= \delta(q', \alpha)\,), \quad \text{für } q' \in Q',\\ q_0' &= S,\\ F' &= \{q' \in Q' \mid q' \cap F \neq \emptyset\}. \end{aligned}$$

Nun sieht man leicht, daß für ein beliebiges Wort $x = \alpha_1 \ldots \alpha_n \in \Sigma^*$ gilt:

$x \in L(M)$

$\iff$

$\delta(S, x) \cap F \neq \emptyset$

$\iff$

es gibt eine Folge von Teilmengen $S = Q_0, \ldots, Q_n$ von Q, bzw. Zuständen $q_0' = Q_0, \ldots, q_n' = Q_n$ von M', sodaß $\delta'(q_0', \alpha_1) = q_1', \ldots, \delta'(q_{n-1}, \alpha_n) = q_n'$, und $q_n' \cap F \neq \emptyset$

$\iff$

$\delta'(q_0', x) \in F'$

$\iff$

$x \in L(M')$. ∎

Die Erweiterung des Begriffs des endlichen Automaten zu einem nichtdeterministischen endlichen Automaten bringt also keine Potenzsteigerung der betrachteten Klasse von Automaten. Diese Beobachtung werden wir auch später wieder machen, wenn wir uns mit Turing–Maschinen befassen.

Neben den besprochenen Beschreibungsmöglichkeiten für reguläre Sprachen (von DEA akzeptiert, von NEA akzeptiert, durch einen regulären Ausdruck erzeugt) gibt es noch einige andere. So könnte man etwa auch reguläre Grammatiken betrachten. Diese führen wiederum zur selben Menge der regulären Sprachen.

Endliche Automaten kann man natürlich auch mit einer Ausgabefunktion versehen und auf diese Weise Funktionen berechnen lassen.

Wir wollen auf diese Variationen aber nicht mehr näher eingehen sondern verweisen auf die angefügte Literaturliste, insbesondere [HU] und [LP].