## 4.3 Reguläre Sprachen

In diesem Kapitel beschäftigen wir uns etwas näher mit den regulären Sprachen, insbesondere mit der Möglichkeit verschiedener Charakterisierungen und den Eigenschaften.

### 4.3.1 Endliche Automaten

Zur Definition der regulären Sprachen benutzten wir reguläre Grammatiken. Grammatiken können Wörter über einem gewissen Alphabet *erzeugen* und so eine Menge von Wörtern, eine *Sprache*, beschreiben. Jetzt wollen wir einen anderen Mechanismus zur Beschreibung benutzen: Die *endlichen Automaten*. Der Automat erhält ein Wort als Eingabe, „arbeitet“ über diesem Wort und *erkennt* (oder *akzeptiert*) es oder auch nicht. Alle Wörter, die ein Automat akzeptiert, bilden die von ihm beschriebene *Sprache*.

**Definition 4.23 (Deterministischer endlicher Automat)** *Ein deterministischer endlicher Automat (Wir wollen ihn kurz mit DEA bezeichnen) $A$ ist ein 5-Tupel $A = (Z, \Sigma, \delta, z_0, E)$, wobei $Z$ das Zustandsalphabet und $\Sigma$ das Eingabealphabet mit $Z \cap \Sigma = \emptyset$ sind. $z_0 \in Z$ ist der Anfangszustand, $E \subseteq Z$ die Menge der Endzustände. Mit $\delta$ bezeichnen wir die Zustandsüberführungsfunktion $\delta\colon Z \times \Sigma \to Z$.*

Wir interpretieren einen DEA als endliche Kontrolle, die sich in einem Zustand $z \in Z$ befindet und eine Folge von Symbolen aus $\Sigma$, die auf einem Band geschrieben stehen, liest (siehe Abbildung 4.4). In einem Schritt bewegt sich der Lesekopf des Automaten, der augenblicklich über der Zelle des

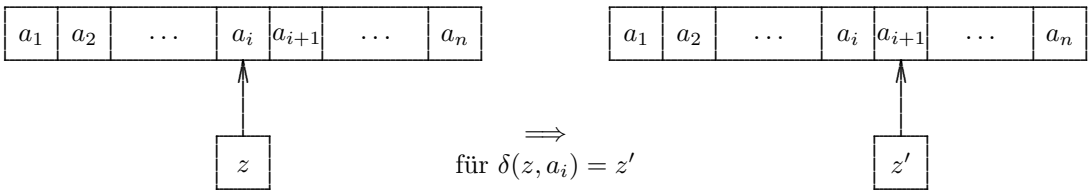


Abbildung 4.4: Interpretation der Arbeitsweise eines endlichen Automaten

Bandes mit dem Inhalt $a_i \in \Sigma$ steht und sich im Zustand $z \in Z$ befindet, einen Schritt nach rechts und geht in den Zustand $\delta(z, a_i) = z'$ über. Ist $\delta(z, a_i) = z' \in E$, d. h. ein akzeptierender Zustand, so hat der DEA das ganze Wort, das er, beginnend in Startzustand $z_0$, gelesen hat, akzeptiert.

Um die von einem DEA akzeptierte Sprache, d. h. die Menge aller von ihm akzeptierten Wörter, formal beschreiben zu können, erweitern wir die Zustandsfunktion $\delta$ zur Funktion $\hat{\delta}\colon Z \times \Sigma^* \to Z$ rekursiv durch folgende Definition.

**Definition 4.24 (Erweiterte Zustandsüberführungsfunktion eines DEA)** *Sei $A$ ein deterministischer endlicher Automat $A = (Z, \Sigma, \delta, z_0, E)$, die erweiterte Zustandsüberführungsfunktion $\hat{\delta}\colon Z \times \Sigma^* \to Z$ wird definiert durch*

(i) $\hat{\delta}(z, \varepsilon) = z$ *für alle $z \in Z$,*

(ii) $\hat{\delta}(z, wa) = \delta(\hat{\delta}(z, w), a)$ *für alle $z \in Z$, $a \in \Sigma$, $w \in \Sigma^*$.*

Ich erwähne, dass in der Literatur $\hat{\delta}$ oft auch nur als $\delta$ bezeichnet wird. Das ist in der Tatsache begründet, dass für alle $z \in Z$ und für alle $a \in \Sigma$ die Gleichheit $\hat{\delta}(z, a) = \delta(z, a)$ gilt, d. h. $\delta$ ist eine Einschränkung der Funktion $\hat{\delta}$ auf $Z \times \Sigma$.

Jetzt können wir die von dem Automaten akzeptierte Sprache definieren:

**Definition 4.25 (Akzeptierte Sprache eines DEA)** *Für einen DEA $A = (Z, \Sigma, \delta, z_0, E)$ sei die von ihm akzeptierte Sprache $T(A)$ definiert durch $T(A) = \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in E\}$.*

Nun ist es Zeit für ein Beispiel.

**Beispiel 4.26** Wir betrachten den deterministischen endlichen Automaten

$$A = (\{z_0, z_1, z_2, z_3\}, \{a, b\}, \delta, z_0, \{z_3\}),$$

wobei $\delta$ durch die Tabelle

| $\delta$ | $z_0$ | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|---|
| $a$ | $z_1$ | $z_1$ | $z_3$ | $z_3$ |
| $b$ | $z_0$ | $z_2$ | $z_0$ | $z_3$ |

gegeben ist. Um zu entscheiden, ob zum Beispiel die Wörter $ab$, $aaba$ oder $bababb$ von dem Automaten akzeptiert werden, müssen wir $\hat{\delta}(z_0, ab)$, $\hat{\delta}(z_0, aaba)$ und $\hat{\delta}(z_0, bababb)$ bestimmen. Dazu benutzen wir die Definition von $\hat{\delta}$ und natürlich die Definition des Automaten, insbesondere der Überführungsfunktion $\delta$.

$$\begin{aligned}
\hat{\delta}(z_0, ab) &= \delta(\hat{\delta}(z_0, a), b) \\
&= \delta(\delta(\hat{\delta}(z_0, \varepsilon), a), b) \\
&= \delta(\delta(z_0, a), b) \\
&= \delta(z_1, b) \\
&= z_2
\end{aligned}$$

Also gilt $\hat{\delta}(z_0, ab) = z_2$ und somit $ab \notin T(A)$, da $z_2 \notin E$ gilt, $z_2$ also kein akzeptierender Zustand ist.

$$\begin{aligned}
\hat{\delta}(z_0, aaba) &= \delta(\hat{\delta}(z_0, aab), a) \\
&= \delta(\delta(\hat{\delta}(z_0, aa), b), a) \\
&= \delta(\delta(\delta(\hat{\delta}(z_0, a), a), b), a) \\
&= \delta(\delta(\delta(\delta(\hat{\delta}(z_0, \varepsilon), a), a), b), a) \\
&= \delta(\delta(\delta(\delta(z_0, a), a), b), a) \\
&= \delta(\delta(\delta(z_1, a), b), a) \\
&= \delta(\delta(z_1, b), a) \\
&= \delta(z_2, a) \\
&= z_3
\end{aligned}$$

Wegen $z_3 \in E$ gilt demnach: $aaba$ wird vom Automaten akzeptiert.

$$\begin{aligned}
\hat{\delta}(z_0, bababb) &= \delta(\hat{\delta}(z_0, aab), a) \\
&= \delta(\delta(\hat{\delta}(z_0, baba), b), b) \\
&= \delta(\delta(\delta(\hat{\delta}(z_0, bab), a), b), b) \\
&= \delta(\delta(\delta(\delta(\hat{\delta}(z_0, ba), b), a), b), b) \\
&= \delta(\delta(\delta(\delta(\delta(\hat{\delta}(z_0, b), a), b), a), b), b) \\
&= \delta(\delta(\delta(\delta(\delta(\delta(\hat{\delta}(z_0, \varepsilon), b), a), b), a), b), b) \\
&= \delta(\delta(\delta(\delta(\delta(\delta(z_0, b), a), b), a), b), b) \\
&= \delta(\delta(\delta(\delta(\delta(z_0, a), b), a), b), b) \\
&= \delta(\delta(\delta(\delta(z_1, b), a), b), b) \\
&= \delta(\delta(\delta(z_2, a), b), b) \\
&= \delta(\delta(z_3, b), b) \\
&= \delta(z_3, b) \\
&= z_3
\end{aligned}$$

Wegen $z_3 \in E$ gilt auch $bababb \in T(A)$.

Wir haben also $ab \notin T(A)$ und $aaba$, $bababb \in T(A)$. Nun gilt es zu bestimmen, welche Sprache $T(A)$ von diesem deterministischen endlichen Automaten $A$ akzeptiert wird. Eine genaue Analyse der Überführungsfunktion $\delta$ würde ergeben (den exakten Beweis führen wir nicht, wird auch vom Hörer dieser Vorlesung nicht verlangt)

$$T(A) = \{w \in \{a,b\}^* \mid w = uabav \text{ für Wörter } u, v \in \{a,b\}^*\},$$

also akzeptiert der Automat alle Wörter über dem Alphabet $\{a,b\}$, die das Teilwort $aba$ besitzen.

Im obigen Beispiel ist es nicht so leicht, die akzeptierte Sprache des Automaten zu bestimmen. Oft wird es etwas leichter, falls wir die Darstellung eines gerichteten Graphen, des sogenannten *Überführungsgraphen* oder auch *Transitionsgraphen* des deterministischen endlichen Automaten benutzen. Diese Darstellung sieht folgendermaßen aus.

Die Menge der Zustände $Z$ wird die Knotenmenge des Graphen, $\delta(z_1, a) = z_2$ wird durch eine gerichtete Kante von $z_1$ zu $z_2$, die mit $a$ markiert ist, dargestellt. Der Anfangszustand wird durch einen zum Knoten gehenden Pfeil dargestellt, Endzustände durch zwei konzentrische Kreise (siehe Abbildung 4.5). Wollen wir jetzt wissen, ob ein Wort vom Automaten akzeptiert wird, dann

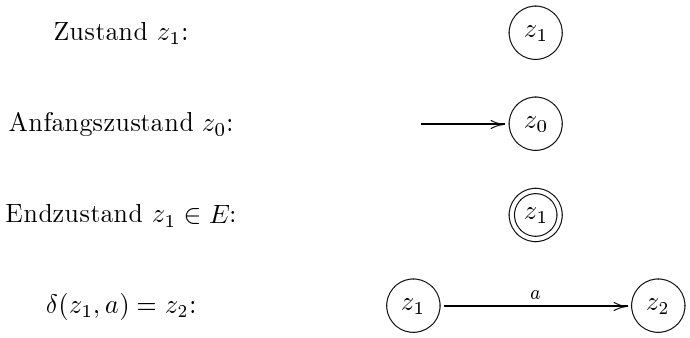


Abbildung 4.5: Konstruktion des Überführungsgraphen eines DEA

müssen wir, beginnend im Zustand $z_0$ die Kanten des Graphen entsprechend des Wortes entlang wandern.

In Abbildung 4.6 ist der Überführungsgraph zum DEA aus Beispiel 4.26 dargestellt. Wenn man

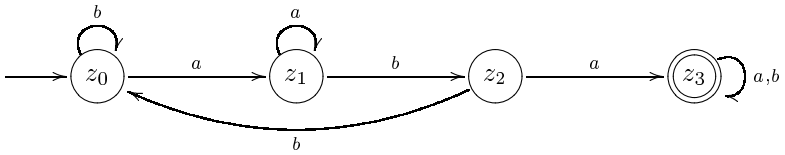


Abbildung 4.6: Überführungsgraph des DEA aus Beispiel 4.26

sich diesen Graphen genauer ansieht, erkennt man eher als aus der Tabelle, dass die akzeptierte Sprache genau die Menge aller Wörter mit dem Teilwort $aba$ ist. Die Zustände sind assoziert mit den Teilen des Wortes $aba$, die bereits eingelesen wurden: $z_0$: noch nichts von $aba$, $z_1$: schon das $a$ von $aba$, $z_2$: schon das $ab$ von $aba$, $z_3$: das gesamte Teilwort $aba$. Wenn $aba$ gelesen wurde (Zustand $z_3$), bleibt der Automat immer in diesem akzeptierenden Zustand.

Betrachten wir ein weiteres Beispiel.

**Beispiel 4.27** Es sei $A$ der deterministisch endliche Automat

$$A = (\{z_0, z_1, z_2, z_3\}, \{0,1\}, \delta, z_0, \{z_2, z_3\}),$$

mit der Überführungsfunktion $\delta$, gegeben durch folgende Tabelle.

| $\delta$ | $z_0$ | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|---|
| 0 | $z_0$ | $z_3$ | $z_3$ | $z_0$ |
| 1 | $z_1$ | $z_2$ | $z_2$ | $z_1$ |

Der dazugehörige Graph ist in Abbildung 4.7 dargestellt. Auch hier nutzen wir wiederum die

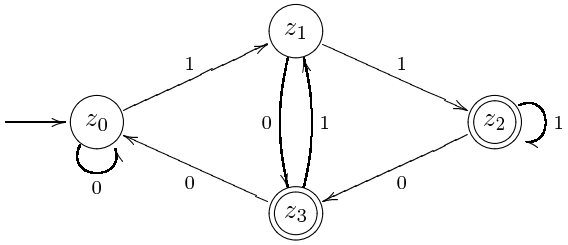


Abbildung 4.7: Überführungsgraph des DEA aus Beispiel 4.27

Zustände, um Informationen über den Teil des Wortes zu speichern, der bereits eingelesen wurde. Man erkennt, dass der Zustand die beiden zuletzt gelesenen Buchstaben repräsentiert. Der Zustand $z_0$ bedeutet, die letzten beiden Buchstaben waren 00, bei $z_1$: 01, bei $z_2$: 11 und bei $z_3$: 10. Akzeptiert wird im Zustand $z_2$ und $z_3$, also genau dann, wenn das vorletzte Zeichen eine 1 war. Also gilt

$$T(A) = \{w \in \{0,1\}^* \mid w = u1x \text{ mit } u \in \{0,1\}^*,\ x \in \{0,1\}\}$$

für die akzeptierte Sprache $T(A)$, also ist $T(A)$ die Menge aller Wörter über $\{0,1\}$, deren vorletztes Symbol eine 1 ist.

Es stellt sich natürlich die Frage, welche Sprachklasse durch deterministische endliche Automaten beschrieben wird. Die akzeptierten Sprachen in den Beispielen 4.26 und 4.27 sind regulär. Gilt das für alle von DEA akzeptierten Sprachen? Die Antwort gibt folgender Satz.

**Satz 4.28** *Sei $A$ ein deterministischer endlicher Automat. Dann ist die von $A$ akzeptierte Sprache $T(A)$ regulär (vom Typ 3).*

*Beweis.* Sei $A = (Z, \Sigma, \delta, z_0, E)$ ein DEA. Wir konstruieren eine Grammatik $G = (V, \Sigma, P, S)$ folgendendermaßen: wir setzen $V = Z$ und $S = z_0$. Weiterhin sei

$$P = \{z_1 \to az_2 \mid \delta(z_1, a) = z_2\} \cup \{z_1 \to a \mid \delta(z_1, a) \in E\}.$$

Zusätzlich müssen wir zu $P$ noch die Regel $z_0 \to \varepsilon$ hinzunehmen, falls $\varepsilon \in T(A)$, also $z_0 \in E$ gilt.

Offensichtlich ist die so konstruierte Grammatik $G$ vom Typ 3, also regulär und erzeugt die gleiche Sprache, die $A$ akzeptiert, was noch zu beweisen wäre. Wir verweisen aber an dieser Stelle für den interessierten Leser auf die Literatur. □

Wir wollen die Konstruktion aus obigem Beweis an einem Beispiel demonstrieren.

**Beispiel 4.29** Sei $A = (\{z_0, z_1, z_2, z_3\}, \{a, b\}, \delta, z_0, \{z_3\})$ der DEA aus Beispiel 4.26 mit der Überführungsfunktion $\delta$:

| $\delta$ | $z_0$ | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|---|
| $a$ | $z_1$ | $z_1$ | $z_3$ | $z_3$ |
| $b$ | $z_0$ | $z_2$ | $z_0$ | $z_3$ |

Wir konstruieren jetzt die äquivalente Typ-3-Grammatik $G = (V, \Sigma, P, S)$:

$$
\begin{aligned}
V &= \{z_0, z_1, z_2, z_3\},\\
\Sigma &= \{a, b\},\\
S &= z_0,\\
P &= \{z_0 \to az_1,\ z_0 \to bz_0,\ z_1 \to az_1,\ z_1 \to bz_2,\ z_2 \to az_3,\ z_2 \to bz_0,\ z_3 \to az_3,\ z_3 \to bz_3\}\\
&\quad \cup \{z_2 \to a,\ z_3 \to a,\ z_3 \to b\}.
\end{aligned}
$$

Bei der Konstruktion von $P$ handelt sich in der ersten Zeile um die Regeln, die ausgehend von $\delta$ im ersten Schritt konstruiert werden, also zum Beispiel $z_0 \to az_1$ wegen $\delta(z_0, a) = z_1$ oder $z_0 \to bz_0$ wegen $\delta(z_0, b) = z_0$. In der zweiten Zeile kommen die terminierenden Regeln hinzu für die Überführungen, die in einen Endzustand gehen, also zum Beispiel $z_2 \to a$ wegen $\delta(z_2, a) = z_3$ und $z_3 \in E$.

### 4.3.2 Nichtdeterministische endliche Automaten

Nachdem wir wissen, dass jede von einem deterministischen endlichen Automaten akzeptierte Sprache vom Typ 3, also regulär ist, interessiert natürlich die Umkehrung, also ob jede Typ-3-Sprache auch von einem deterministischen endlichen Automaten akzeptiert werden kann. Eine naheliegende Idee wäre, die Konstruktion aus dem obigen Beweis einfach entsprechend zu invertieren.

Das führt allerdings zu zwei Schwierigkeiten.

1. Wenn wir zum Beispiel eine reguläre Grammatik mit den Regeln $A \to aB$ und $A \to aC$ hätten, dann müssten gleichzeitig $\delta(A, a) = B$ und $\delta(A, a) = C$ gelten, was nicht möglich ist.
2. Wenn wir zum Beispiel eine reguläre Grammatik mit einer Regel $A \to aabB$ hätten, müßten wir eine Überführung vom Zustand $A$ zum Zustand $B$ mit $aab$ erzeugen, was nicht möglich ist.

Um die Idee der Invertierung des Beweises von Satz 4.28 jedoch nicht fallen zu lassen und zu zeigen, dass reguläre Sprachen von DEA akzeptiert werden können, führen wir einfach neue Modelle ein, um die Schwierigkeiten 1 und 2 zu überwinden.

Um die Schwierigkeit 2 zu beseitigen, betrachten wir eine Normalform von regulären Grammatiken:

**Satz 4.30** *Zu jeder regulären (Typ-2) Grammatik $G = (V, \Sigma, P, S)$ gibt es eine äquivalente Grammatik $G' = (V', \Sigma, P', S')$, die nur Regeln der Form $A \to aB$ oder $A \to a$ mit $A, B \in V'$ und $a \in \Sigma$ hat, mit der Ausnahme $S' \to \varepsilon$, falls $S'$ nicht auf der rechten Seite einer Regel vorkommt.*

*Beweis.* Der Beweis würde in zwei Schritten ablaufen: zuerst müssen wir die Regeln $A \to \varepsilon$ für $A \neq S$ der Grammatik $G$ beseitigen. Das wollen wir hier nicht ausführen, sondern auf den entsprechenden Satz für kontextfreie Sprachen verweisen, dessen Beweis hier vollständig übernommen werden kann.

Zweitens müssen wir dann die Regeln $A \to wB$ oder $A \to w$ mit $|w| \geq 2$ ersetzen. Sei also $A \to a_1a_2 \ldots a_nB$ eine solche Regel mit $A, B \in V$ und $n \geq 2$, dann nehmen wir neue Nichtterminale $A_1, A_2, \ldots, A_{n-1}$ in die Menge $V'$ auf, die noch nicht verwendet wurden und ersetzen die Regel $A \to a_1a_2 \ldots a_nB$ durch die Regeln

$$A \to a_1A_1,\ A_1 \to a_2A_2\ , \ldots,\ A_{n-1} \to a_nB$$

in $P'$. Entsprechend ersetzt man eine Regel $A \to a_1a_2 \ldots a_n$ mit $A \in V$ und $n \geq 2$ durch die Regeln

$$A \to a_1B_1,\ B_1 \to a_2B_2\ , \ldots,\ B_{n-1} \to a_n$$

mit den neuen Ncihtterminalen $B_1, B_2, \ldots, B_{n-1}$ in $V'$.

Man kann dann leicht zeigen, dass dann die Regeln $A \to a_1 a_2 \dots a_n B$ und $A \to a_1 a_2 \dots a_n$ durch die Ableitungen

$$A \Longrightarrow a_1 A_1 \Longrightarrow a_1 a_2 A_2 \Longrightarrow \cdots \Longrightarrow a_1 a_2 a_3 \dots a_{n-1} A_{n-1} \Longrightarrow a_1 a_2 a_3 \dots a_{n-1} a_n B$$

bzw.

$$A \Longrightarrow a_1 B_1 \Longrightarrow a_1 a_2 B_2 \Longrightarrow \cdots \Longrightarrow a_1 a_2 a_3 \dots a_{n-1} B_{n-1} \Longrightarrow a_1 a_2 a_3 \dots a_{n-1} a_n$$

simuliert werden und andererseits aber keine neuen Wörter durch die Grammatik $G'$ erzeugt werden können, also $L(G) = L(G')$ gilt. □

Zur Umgehung der oben genannten Schwierigkeit 2 führen wir ein neues Automatenmodell ein, indem wie den Begriff des DEA erweitern und solche Überführungen $\delta(A, a) = B$ und $\delta(A, a) = C$ gleichzeitig zulassen, indem wir *Nichtdeterminismus* benutzen.

**Definition 4.31 (Nichtdeterministischer endlicher Automat)** *Ein nichtdeterministischer endlicher Automat (kurz mit NEA bezeichnet) $A$ ist ein 5-Tupel $A = (Z, \Sigma, \delta, z_0, E)$, wobei $Z$ das Zustandsalphabet und $\Sigma$ das Eingabealphabet mit $Z \cap \Sigma = \emptyset$ sind. $z_0 \in Z$ ist der Anfangszustand, $E \subseteq Z$ die Menge der Endzustände. Mit $\delta$ bezeichnen wir die Zustandsüberführungsfunktion $\delta \colon Z \times \Sigma \to 2^Z$.*

Wie man an der Definition erkennt, ist der NEA gar nicht so weit vom DEA entfernt. Der einzige Unterschied liegt in der Definition der Überführungsfunktion $\delta$. Funktionswerte von $\delta$ sind nicht einzelne Zustände (wie beim DEA) sondern *Mengen von Zuständen*, d. h. $\delta(z, a) = \{z_1, z_2, \dots, z_r\}$ mit $\{z_1, z_2, \dots, z_r\} \subseteq Z$. Wir bemerken, dass $\delta(z, a)$ auch die leere Menge sein kann.

Auch den NEA können wir wiederum in derselben Art und Weise wie beim DEA als Graph darstellen, wobei von einem Zustand für ein und dasselbe Symbol mehrere Pfeile ausgehen können (oder auch keiner), siehe Abbildung 4.8.

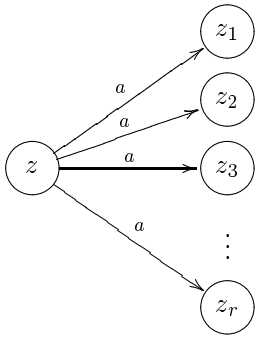


Abbildung 4.8: Nichtdeterminismus beim endlichen Automaten

Die Interpretation der Arbeitsweise des NEA ist analog der des DEA, wobei wir $\delta(z, a) = \{z_1, z_2, \dots, z_r\}$ so interpretieren, dass der Automat, wenn er sich im Zustand $z$ befindet und ein $a$ einliest, in einen der Zustände $z_1, z_2, \dots, z_r$ übergehen kann. Das heisst, der NEA kann ein und dasselbe Wort über *verschiedene* Wege einlesen, wobei alle Wege gleichwertig sein sollen. Das hat natürlich zur Folge, dass der NEA beim Einlesen ein und desselben Wortes einmal einen akzeptierenden Zustand erreichen kann und einmal nicht. Wir werden sagen, dass der NEA genau dann *ein Wort akzeptiert*, wenn er beim Einlesen des Wortes einen akzeptierenden Zustand erreichen *kann*, also *wenn es für das Wort einen Pfad vom Anfangszustand zu einem Endzustand gibt.* Man kann diesen Nichtdeterminismus in gewisser Weise als Parallelverarbeitung auffassen.

Um die von einem NEA akzeptierte Sprache formal definieren zu können, benötigen wir wieder die erweiterte Zustandsfunktion $\hat{\delta} \colon Z \times \Sigma^* \to 2^Z$. Im Prinzip wird sie wieder wie beim NEA definiert, allerdings ist ja jetzt $\hat{\delta}(z, w)$ eine Menge von Zuständen und kann nicht direkt als Argument verwendet werden.

**Definition 4.32 (Erweiterte Zustandsüberführungsfunktion eines NEA)** *Sei $A$ ein NEA mit $A = (Z, \Sigma, \delta, z_0, E)$, die erweiterte Zustandsfunktion $\hat{\delta}\colon Z \times \Sigma^* \to 2^Z$ wird definiert durch*

(i) $\hat{\delta}(z, \varepsilon) = \{z\}$ *für alle $z \in Z$,*

(ii) $\hat{\delta}(z, wa) = \bigcup_{z' \in \hat{\delta}(z,w)} \delta(z', a)$ *das heißt*
$= \{z'' \in Z \mid \exists z' \in Z \text{ mit } z' \in \hat{\delta}(z, w) \text{ und } z'' \in \delta(z', a)\}$ *für $z \in Z$, $a \in \Sigma$, $w \in \Sigma^*$.*

**Definition 4.33 (Akzeptierte Sprache eines NEA)** *Für einen NEA $A = (Z, \Sigma, \delta, z_0, E)$ sei die von ihm akzeptierte Sprache $T(A)$ definiert durch $T(A) = \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \cap E \neq \emptyset\}$.*

**Bemerkung 4.34** Bitte beachten Sie, dass die Definitionen des NEA sowie der erweiterten Zustandsfunktion von den Definitionen bei SCHÖNING in [9] etwas abweichen. Man kann aber sehr leicht zeigen, dass die Klasse der akzeptierbaren Mengen gleich sind.

Sehen wir uns ein Beispiel an, um den Mechanismus des Nichtdeterminismus besser zu verstehen.

**Beispiel 4.35** Sei $A = (\{z_0, z_1, z_2\}, \{0, 1\}, \delta, z_0, \{z_2\})$, wobei $\delta$ durch die Tabelle

| $\delta$ | $z_0$ | $z_1$ | $z_2$ |
|---|---|---|---|
| 0 | $\{z_0\}$ | $\{z_2\}$ | $\emptyset$ |
| 1 | $\{z_0, z_1\}$ | $\{z_2\}$ | $\emptyset$ |

gegeben ist, ein nichtdeterministischer endlicher Automat. Der dazugehörige Graph ist in der Abbildung 4.9 dargestellt. Betrachten wir nun die Eingabe 111 und fragen nach $\hat{\delta}(z_0, 111)$, also

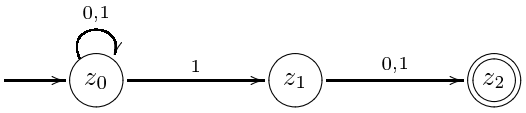


Abbildung 4.9: Überführungsgraph des NEA aus Beispiel 4.35

nach den Zuständen, die bei der Eingabe von 111 durch den NEA erreicht werden können. Beim Einlesen des ersten Symbols (eine 1) kann der Automat wählen zwischen dem Folgezustand $z_0$ oder $z_1$. Im letzteren Fall liest er dann die zweite 1 ein und landet (keine Wahlmöglichkeit) im Zustand $z_2$, von dem aus er die dritte 1 nicht mehr einlesen kann (es gibt keine Überführung mehr), also erreicht er über diesen Weg *keinen* Zustand. Im ersteren Fall jedoch kann er bei der Eingabe der zweiten 1 wiederum wählen zwischen den Folgezuständen $z_0$ und $z_1$: Wählt er $z_1$, landet er mit der letzten 1 in $z_2$, wählt er jedoch $z_0$, so kann er bei der letzten 1 wiederum zwischen den Folgezuständen $z_0$ und $z_1$ wählen. Summa summarum kann der Automat beim Einlesen von 111 die Zustände $z_0, z_1, z_2$ erreichen, also

$$\hat{\delta}(z_0, 111) = \{z_0, z_1, z_2\}.$$

Nun gilt

$$\hat{\delta}(z_0, 111) \cap E = \{z_0, z_1, z_2\} \cap \{z_2\} = \{z_2\} \neq \emptyset,$$

das heißt, für die Eingabe 111 gibt es einen Weg vom Anfangszustand in einen akzeptierenden Zustand, also gilt $111 \in T(A)$, d. h. die Eingabe 111 wird akzeptiert, gehört also zur akzeptierten Sprache.

Für weitere Eingaben gilt:

$$\begin{aligned}\hat{\delta}(z_0, 0) &= \{z_0\},\\ \hat{\delta}(z_0, 1) &= \{z_0, z_1\},\\ \hat{\delta}(z_0, 01) &= \{z_1\},\\ \hat{\delta}(z_0, 11) &= \{z_0, z_1, z_2\},\\ \hat{\delta}(z_0, 001) &= \{z_0, z_1\},\\ \hat{\delta}(z_0, 011) &= \{z_0, z_1, z_2\},\end{aligned}$$

also werden 11 sowie 011 akzeptiert und 0, 1, 01 sowie 001 nicht akzeptiert. Für die akzeptierte Sprache $T(A)$ gilt:

$$T(A) = \{w \in \{0,1\}^* \mid w = u1x \text{ mit } u \in \{0,1\}^*,\ x \in \{0,1\}\},$$

also ist $T(A)$ die Menge aller Wörter über $\{0,1\}$, deren vorletztes Symbol eine 1 ist.

Man kann jeden DEA natürlich als NEA auffassen, nämlich für den dann $\delta(z,a)$ immer eine Einermenge ist. Also:

**Folgerung 4.36** *Jede von einem deterministischen endlichen Automaten akzeptierbare Sprache ist auch von einem nichtdeterministischen endlichen Automaten akzeptierbar.* □

Die Menge, die vom NEA im Beispiel 4.35 akzeptiert wurde, kann auch von einem DEA akzeptiert werden (siehe Beispiel 4.27). Natürlich ergibt sich sofort die Frage, ob jede von einem NEA akzeptierte Sprache auch von einem DEA akzeptiert werden kann. Die Antwort liefert der folgende Satz.

**Satz 4.37** *Jede von einem nichtdeterministischen endlichen Automaten akzeptierbare Sprache ist auch von einem deterministischen endlichen Automaten akzeptierbar.*

*Beweis.* Sei $A = (Z, \Sigma, \delta, z_0, E)$ ein NEA. Wir konstruieren einen DEA $A' = (Z', \Sigma, \delta', z_0', E')$ durch:

$$\begin{aligned}Z' &= 2^Z,\\ z_0' &= \{z_0\},\\ E' &= \{z' \in Z' \mid z' \cap E \neq \emptyset\},\end{aligned}$$

sowie

$$\delta'(z', a) = \bigcup_{z \in z'} \delta(z, a)$$

für alle $z' \in Z'$ und $a \in \Sigma$. Dann gilt

$$T(A) = T(A'),$$

was wir an dieser Stelle nicht beweisen werden. □

Im obigen Beweis ist die Zustandsmenge des konstruierten DEA genau die Potenzmenge der Zustandsmenge des gegebenen NEA. Falls man zu einem konkret gegebenen DEA den äquivalenten NEA konstruiert, stellt man fest, dass oft nicht alle Teilmengen von $Z$ auch wirklich erreicht werden können. Folglich reicht es aus, wenn wir, beginnend mit der Menge $\{z_0\}$ jeweils für alle $z' \in Z'$ die Teilmengen $\delta'(z', x)$ für alle $x \in \Sigma$ berechnen und die neu erzeugten Teilmengen in die Menge der Zustände $Z'$ aufnehmen, falls sie noch nicht enthalten sind. Kommt kein neuer Zustand mehr hinzu, wären wir fertig mit der Konstruktion von $\delta'$. Wir wollen dieses Vorgehen an einem Beispiel demonstrieren.

**Beispiel 4.38** Wir nehmen den NEA

$$A = (\{z_0, z_1, z_2\}, \{0, 1\}, \delta, z_0, \{z_2\})$$

aus Beispiel 4.35 mit der in folgender Tabelle gegebenen Überführungsfunktion $\delta$.

| $\delta$ | $z_0$ | $z_1$ | $z_2$ |
|---|---|---|---|
| 0 | $\{z_0\}$ | $\{z_2\}$ | $\emptyset$ |
| 1 | $\{z_0, z_1\}$ | $\{z_2\}$ | $\emptyset$ |

Wir konstruieren jetzt den äquivalenten DEA

$$A' = (Z', \{0, 1\}, \delta', z', E')$$

gemäß Beweis des Satzes 4.37. Zuerst gilt $z'_0 = \{z_0\}$. Die weiteren Zustände sowie die Überführungsfunktion berechnen wir per Definition in folgender Tabelle (spaltenweise).

| $\delta'$ | $\{z_0\}$ | $\{z_0, z_1\}$ | $\{z_0, z_2\}$ | $\{z_0, z_1, z_2\}$ |
|---|---|---|---|---|
| 0 | $\{z_0\}$ | $\{z_0, z_2\}$ | $\{z_0\}$ | $\{z_0, z_2\}$ |
| 1 | $\{z_0, z_1\}$ | $\{z_0, z_1, z_2\}$ | $\{z_0, z_1\}$ | $\{z_0, z_1, z_2\}$ |

Also gilt

$$Z' = \{\{z_0\}, \{z_0, z_1\}, \{z_0, z_2\}, \{z_0, z_1, z_2\}\}$$

und da nur die Zustände $\{z_0, z_2\}$ und $\{z_0, z_1, z_2\}$ einen Zustand aus $E$ enthalten, sind sie die einzigen neuen Endzustände, also

$$E' = \{\{z_0, z_2\}, \{z_0, z_1, z_2\}\}.$$

Damit wäre der äquivalente DEA $A'$ zum NEA $A$ konstruiert. Zur besseren Lesbarkeit bezeichnen wir die Zustände um: $\{z_0\} =: q_0$, $\{z_0, z_1\} =: q_1$, $\{z_0, z_2\} =: q_2$ und $\{z_0, z_1, z_2\} =: q_3$. Dann gilt

$$A' = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta', q_0, \{q_2, q_3\})$$

mit

| $\delta'$ | $q_0$ | $q_1$ | $q_2$ | $q_3$ |
|---|---|---|---|---|
| 0 | $q_0$ | $q_2$ | $q_0$ | $q_2$ |
| 1 | $q_1$ | $q_3$ | $q_1$ | $q_3$ |

In Abbildung 4.10 finden Sie den Graphen zum Automaten $A'$. Man erkennt, dass die Automaten in

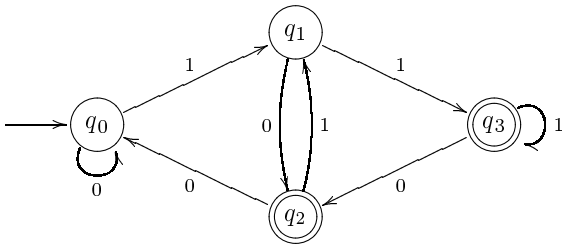


Abbildung 4.10: Überführungsgraph des DEA $A'$ aus Beispiel 4.38

den Abbildungen 4.7 und 4.10 bis auf Bezeichnungen der Zustände identisch sind. Also akzeptieren Sie auch die gleiche Sprache.

**Bemerkung 4.39** 1. Beim Konstruieren des äquivalenten DEA zum gegebenen NEA im obigen Beispiel haben wir den gleichen Automaten (bis auf Bezeichnungen) erhalten, den wir auch schon vorher betrachtet hatten. Das muss natürlich nicht immer sein. Insbesondere erhält man im Allgemeinen bei dieser Konstruktion Automaten, die nicht *minimal* in dem Sinne sind, dass man äquivalente DEA finden kann, die eventuell weniger Zustände haben.

2. Im obigen Beispiel hatte der NEA drei Zustände, der DEA vier. Führt man den gleichen Übergang vom NEA zum DEA für die (von der Struktur gleiche) Sprache

$$T(A) = \{w \in \{0,1\}^* \mid w = u1v \text{ mit } u \in \{0,1\}^*,\ v \in \{0,1\}^9\}$$

durch, also für die Menge aller Wörter über dem Alphabet $\{0,1\}$, deren zehntletztes Symbol eine 1 ist, so benötigt der NEA 11 Zustände, der DEA aber $2^{10}$ Zustände. Es lässt sich zeigen, dass es keinen DEA für diese Sprache mit weniger Zustände gibt. Dieses Resultat kann man für beliebiges $n$ verallgemeinern. Die Anzahl der Zustände beim Übergang vom NEA zum DEA kann also *exponentiell wachsen*.

Wir haben das Modell des nichtdeterministischen endlichen Automaten eingeführt, weil wir letztendlich beweisen wollten, dass die Klasse der Typ-3-Sprachen und die Klasse der Sprachen, die von deterministischen endlichen Automaten akzeptiert werden, identisch sind. Mit dem folgenden Satz kommen wir diesem Beweis sehr nahe.

**Satz 4.40** *Sei $G$ eine reguläre Grammatik, also vom Typ 3, dann existiert ein nichtdeterministischer endlicher Automat $A$ mit $T(A) = L(G)$.*

*Beweis.* Wie bereits angekündigt, benutzt der Beweis eigentlich die gleiche Idee wie beim Übergang vom DEA zur regulären Grammatik. Sei $G = (V, \Sigma, P, S)$ die reguläre Grammatik. Wir konstruieren einen NEA $A = (Z, \Sigma, \delta, z_0, E)$ wie folgt:

$$\begin{aligned} Z &= V \cup \{X\}, \\ z_0 &= S, \\ E &= \begin{cases} \{S, X\} & \text{für } S \to \varepsilon \in P, \\ \{X\} & \text{für } S \to \varepsilon \notin P, \end{cases} \end{aligned}$$

Des weiteren definieren wir die Überführungsfunktion $\delta$ durch

$$\delta(A, a) = \begin{cases} \{B \mid A \to aB \in P\} \cup \{X\} & \text{für } A \to a \in P, \\ \{B \mid A \to aB \in P\} & \text{für } A \to a \notin P, \end{cases}$$

für $A \in V$ und $a \in \Sigma$.

Jetzt könnten und müßten wir beweisen, dass $T(A) = L(G)$ gilt, worauf wir aber an dieser Stelle wiederum verzichten wollen. □

Die Abbildung 4.11 fasst die Ergebnisse der Sätze 4.28, 4.40 sowie 4.37 zusammen. Damit gilt:

**Folgerung 4.41** *Die Klasse der regulären Sprachen* (**Typ 3**) *ist gleich der Klasse der von nichtdeterministischen endlichen Automaten akzeptierten Sprachen* ($\mathcal{L}$(NEA)) *und der Klasse der von deterministischen endlichen Automaten akzeptierten Sprachen* ($\mathcal{L}$(DEA))*, also*

$$\textbf{Typ 3} = \mathcal{L}(\text{NEA}) = \mathcal{L}(\text{DEA}).$$

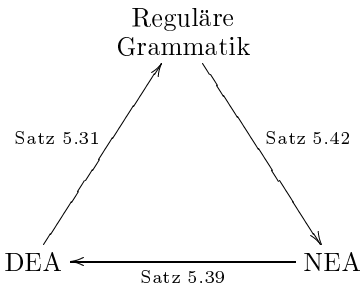


Abbildung 4.11: Beweisschema für Mechanismen zur Beschreibung regulärer Sprachen

### 4.3.3 Reguläre Ausdrücke

Nachdem wir in den vorhergehenden Kapiteln die Menge der regulären Sprachen (oder Typ-3-Sprachen) durch Grammatiken und Automaten beschrieben haben, wollen wir in diesem Kapitel eine Beschreibungsart betrachten, die algebraischer Natur ist, nämlich die regulären Ausdrücke. Obwohl die Beschreibung auf algebraische Operationen zurückgeht, also eigentlich recht mathematischer Natur ist, werden reguläre Ausdrücke in vielen Gebieten der Informatik angewendet, zum Beispiel bei der Suche in Editoren.

Die Menge der regulären Ausdrücke über einem Alphabet $\Sigma$ werden wir induktiv definieren.

**Definition 4.42 (Reguläre Ausdrücke)** *Sei $\Sigma$ ein Alphabet, dann gilt:*

(i) *$\emptyset$ ist ein regulärer Ausdruck über $\Sigma$.*

(ii) *$\varepsilon$ ist ein regulärer Ausdruck über $\Sigma$.*

(iii) *Für jedes $a \in \Sigma$ ist $a$ ein regulärer Ausdruck über $\Sigma$.*

(iv) *Wenn $\alpha$ und $\beta$ reguläre Ausdrücke über $\Sigma$ sind, so auch $\alpha\beta$, $(\alpha \mid \beta)$ und $(\alpha)^*$.*

Reguläre Ausdrücke über einem Alphabet $\Sigma$ sind also erst einmal nur Wörter spezieller Art über diesem Alphabet. Nun ordnen wir solch einem Wort eine Sprache zu. Die Definition dieser Zuordnung erfolgt wiederum rekursiv.

**Definition 4.43 (Sprache eines regulären Ausdrucks)** *Sei $\Sigma$ ein Alphabet und $\gamma$ ein regulärer Ausdruck über $\Sigma$, dann wird die von $\gamma$ beschriebene Sprache $L(\gamma) \subseteq \Sigma^*$ wie folgt definiert.*

(i) *Für $\gamma = \emptyset$ gilt $L(\gamma) = \emptyset$.*

(ii) *Für $\gamma = \varepsilon$ gilt $L(\gamma) = \{\varepsilon\}$.*

(iii) *Für $\gamma = a$ mit $a \in \Sigma$ gilt $L(\gamma) = \{a\}$.*

(iv) *Für $\gamma = \alpha\beta$ gilt $L(\gamma) = L(\alpha) \cdot L(\beta)$.*

(v) *Für $\gamma = (\alpha \mid \beta)$ gilt $L(\gamma) = L(\alpha) \cup L(\beta)$.*

(vi) *Für $\gamma = (\alpha)^*$ gilt $L(\gamma) = (L(\alpha))^*$.*

**Beispiel 4.44** Wir betrachten den regulären Ausdruck

$$(0 \mid (0 \mid 1)^*00).$$

Dann können wir die zugeordnete Sprache wie folgt gemäß der Definition bilden (hier für ein erstes

Beispiel sehr ausführlich aufgeschrieben):

$$\begin{aligned}
L((0 \mid (0 \mid 1)^*00)) &= L(0) \cup L((0 \mid 1)^*00)) \\
&= L(0) \cup (L((0 \mid 1)^*0)) \cdot L(0)) \\
&= L(0) \cup ((L((0 \mid 1)^*)) \cdot L(0)) \cdot L(0)) \\
&= L(0) \cup (((L((0 \mid 1)))^* \cdot L(0)) \cdot L(0)) \\
&= L(0) \cup (((L(0) \cup L(1))^* \cdot L(0)) \cdot L(0)) \\
&= \{0\} \cup ((((\{0\} \cup \{1\})^* \cdot \{0\}) \cdot \{0\}) \\
&= \{0\} \cup ((\{0,1\}^* \cdot \{0\}) \cdot \{0\}) \\
&= \{0\} \cup (\{0,1\}^* \cdot \{00\}),
\end{aligned}$$

das heißt, die vom regulären Ausdruck $(0 \mid (0 \mid 1)^*00)$ beschriebene Sprache ist die Menge aller Wörter über dem Alphabet $\{0,1\}$, die gleich 0 sind oder auf 00 enden.

**Bemerkung 4.45** 1. Wir vereinbaren, dass wir Klammern, die nicht notwendigerweise gebraucht werden, weglassen können. Zum Beispiel können wir statt $(\alpha \mid (\beta \mid \gamma))$ auch $(\alpha \mid \beta \mid \gamma)$ schreiben. Wir schreiben auch $L(\alpha \mid \beta)$ statt $L((\alpha \mid \beta))$ sowie $a^*$ statt $(a)^*$.

2. Wir benutzen die abkürzende Schreibweise $\alpha^n$ für $\underbrace{\alpha\alpha\ldots\alpha}_{n\text{-mal}}$.

3. Wir benutzen die abkürzende Schreibweise $\alpha^+$ für $\alpha^*\alpha$.

4. In der Literatur findet man oft auch abweichende Definitionen der regulären Ausdrücke. Zum Beispiel findet man für $(\alpha \mid \beta)$ auch $(\alpha + \beta)$ oder auch $(\alpha \cup \beta)$. Auch wird natürlich oft $\alpha \cdot \beta$ für $\alpha\beta$ zugelassen.

5. Oft wird in der Literatur zwischen regulärem Ausdruck und beschriebener Sprache nicht unterschieden, das heißt, man identifiziert einen regulären Ausdruck mit der beschriebenen Sprache.

Ich gebe noch ein paar weitere Beispiele an:

**Beispiel 4.46** Weitere Beispiele für reguläre Ausdrücke über $\Sigma = \{a, b\}$ und deren zugeordneten Sprachen sind:

$(a \mid b)^*$ beschreibt die Menge aller Wörter über dem Alphabet $\{a, b\}$.

$(a \mid b)^+$ beschreibt die Menge aller Wörter über dem Alphabet $\{a, b\}$, die nicht dem leeren Wort entsprechen.

$(a \mid b)^*aba(a \mid b)^*$ beschreibt die Menge aller Wörter über dem Alphabet $\{a, b\}$, die das Teilwort $aba$ haben.

$(a \mid b)^*a(a \mid b)^2$ beschreibt die Menge aller Wörter über dem Alphabet $\{a, b\}$, deren drittletztes Symbol ein $a$ ist.

$((a \mid b)(a \mid b))^*$ beschreibt die Menge aller Wörter über dem Alphabet $\{a, b\}$, deren Länge gerade ist.

$(b \mid \varepsilon)(ab)^*(a \mid \varepsilon)$ beschreibt die Menge aller Wörter über dem Alphabet $\{a, b\}$, die nicht das Teilwort $aa$ und nicht das Teilwort $bb$ enthalten.

Wenn man sich die regulären Ausdrücke und die beschriebenen Sprachen im obigen Beispiel anschaut, erkennt man, dass alle Sprachen vom Typ 3 sind, also regulär. Man stellt sich natürlich die Frage, ob alle durch reguläre Ausdrücke beschreibbaren Sprachen regulär sind und ob umgekehrt für jede reguläre Sprache ein regulärer Ausdruck existiert, der sie beschreibt. Die Bezeichnung *reguläre* Ausdrücke suggeriert natürlich die Antwort.

**Satz 4.47 (Kleene)** *Die Menge der durch reguläre Ausdrücke beschreibbaren Sprachen ist genau die Menge der regulären Sprachen.*

*Beweis.* Der Beweis muss in zwei Richtungen geführt werden.

*Teil 1:* Einerseits muss gezeigt werden, dass jeder reguläre Ausdruck eine reguläre Sprache beschreibt. Diesen Teil wollen wir an dieser Stelle skizzieren. Wir werden zeigen, dass zu jedem regulären Ausdruck ein NEA existiert, der genau die vom regulären Ausdruck beschriebene Sprache akzeptiert, womit wegen Folgerung 4.41 diese Sprache regulär oder vom Typ 3 ist.

Sei $\Sigma$ ein Alphabet und $\gamma$ ein regulärer Ausdruck über $\Sigma$. Wir betrachten zuerst die Fälle, in denen $\gamma$ die Form $\gamma = \emptyset$, $\gamma = \varepsilon$ oder $\gamma = a$ mit $a \in \Sigma$ hat. In Abbildung 4.12 sind NEA's


Abbildung 4.12: DEA's für $L(\emptyset)$, $L(\varepsilon)$ sowie $L(a)$ (von links nach rechts)

angegeben, die jeweils die Mengen $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$ sowie $L(a) = \{a\}$ akzeptieren.

Wir nehmen jetzt an, dass $\gamma$ ein regulärer Ausdruck ist, der per Definition schon aus regulären Ausdrücken zusammengesetzt ist. Dann gibt es für $\gamma$ die drei Fälle $\gamma = \alpha\beta$, $\gamma = (\alpha \mid \beta)$ und $\gamma = (\alpha)^*$.

Sei zunächst $\gamma$ ein regulärer Ausdruck der Form $\gamma = \alpha\beta$. Dabei können wir annehmen, dass es bereits NEA's für $L(\alpha)$ und $L(\beta)$ gibt, also dass NEA's $A_\alpha$ und $A_\beta$ existieren mit $T(A_\alpha) = L(\alpha)$ und $T(A_\beta) = L(\beta)$. Nun konstruieren wir den Automaten $A$, indem wir die Automaten $A_\alpha$ und

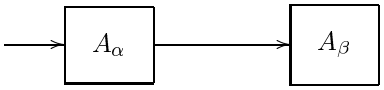


Abbildung 4.13: Schema des NEA's für $L(\alpha\beta)$

$A_\beta$ im Prinzip hintereinander schalten („in Reihe") (siehe Schema in Abbildung 4.13).

Die genaue Konstruktion für den NEA $A$ ist folgende: Die Automaten $A_\alpha = (Z_\alpha, \Sigma, \delta_\alpha, z_{\alpha 0}, E_\alpha)$ und $A_\beta = (Z_\beta, \Sigma, \delta_\beta, z_{\beta 0}, E_\beta)$ seien die Automaten mit $T(A_\alpha) = L(\alpha)$ und $T(A_\beta) = L(\beta)$. Wir konstruieren $A = (Z, \Sigma, \delta, z_0, E)$ wie folgt.

- Jeder Zustand von $A_\alpha$ und $A_\beta$ ist auch Zustand von $A$, also $Z = Z_\alpha \cup Z_\beta$.
- Der Anfangszustand von $A_\alpha$ ist auch Anfangszustand für $A$, also $z_0 = z_{\alpha 0}$.
- Die Menge der Endzustände von $A_\beta$ wird die Menge der Endzustände von $A$, also $E = E_\beta$.
- Alle Überführungen von $A_\alpha$ und $A_\beta$ gelten auch für $A$. Von allen Zuständen von $A_\alpha$, für die es Überführungen zu Endzuständen von $A_\alpha$ gibt, gibt es zusätzlich Überführungen für das gleiche Symbol zum Anfangszustand von $A_\beta$. Formal gilt für alle $z \in Z$ und $a \in \Sigma$:

$$\delta(z,a) = \begin{cases} \delta_\beta(z,a) & \text{für } z \in Z_\beta, \\ \delta_\alpha(z,a) & \text{für } z \in Z_\alpha \text{ und } \delta_\alpha(z,a) \cap E_\alpha = \emptyset, \\ \delta_\alpha(z,a) \cup \{z_{\beta 0}\} & \text{für } z \in Z_\alpha \text{ und } \delta_\alpha(z,a) \cap E_\alpha \neq \emptyset. \end{cases}$$

Für den Fall $\varepsilon \in L(\alpha)$ muss man noch Sonderregelungen treffen, darauf verzichten wir hier.

Wie man nachweisen kann, gilt dann $T(A) = L(\alpha\beta)$, das heißt $A$ akzeptiert ein Wort genau dann, wenn ein erster Teil des Wortes von $A_\alpha$ und der Rest des Wortes von $A_\beta$ akzeptiert wird.

Habe $\gamma$ nun die Form $\gamma = (\alpha \mid \beta)$. Wir setzen wieder voraus, dass die Automaten $A_\alpha = (Z_\alpha, \Sigma, \delta_\alpha, z_{\alpha 0}, E_\alpha)$ und $A_\beta = (Z_\beta, \Sigma, \delta_\beta, z_{\beta 0}, E_\beta)$ die Automaten mit $T(A_\alpha) = L(\alpha)$ und $T(A_\beta) = L(\beta)$ seien. Wir konstruieren dann den Automaten $A$ in folgender Art und Weise (wir schalten die Automaten $A_\alpha$ und $A_\beta$ im Prinzip „parallel", siehe Abbildung 4.14):

$$A = (Z_\alpha \cup Z_\beta \cup \{z_0\}, \Sigma, \delta, z_0, E_\alpha \cup E_\beta),$$

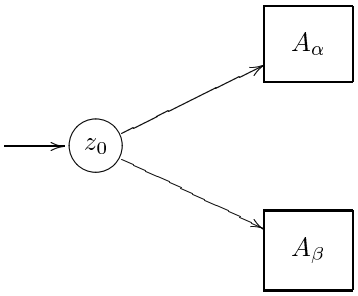


Abbildung 4.14: Schema des NEA's für $L((\alpha \mid \beta))$

die Funktion $\delta$ ist dabei wie folgt definiert.

$$\delta(z,a) = \begin{cases} \delta_\alpha(z,a) & \text{für } z \in Z_\alpha, \\ \delta_\beta(z,a) & \text{für } z \in Z_\beta, \\ \delta_\alpha(z_{\alpha 0},a) \cup \delta_\beta(z_{\beta 0},a) & \text{für } z = z_0. \end{cases}$$

Der Automat akzeptiert dann ein Wort genau dann, wenn es von $A_\alpha$ *oder* $A_\beta$ akzepterit wird. Also gilt $T(A) = A_\alpha \cup A_\beta$, das heißt $T(A) = L((\alpha \mid \beta))$.

Habe $\gamma$ nun die Form $\gamma = (\alpha)^*$. Wir setzen wiederum voraus, dass der Automat $A_\alpha = (Z_\alpha, \Sigma, \delta_\alpha, z_{\alpha 0}, E_\alpha)$ der Automat mit $T(A_\alpha) = L(\alpha)$ ist. Wir konstruieren $A$, indem wir im Prinzip

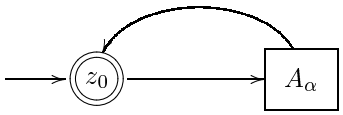


Abbildung 4.15: Schema des NEA's für $L((\alpha)^*)$

eine Schleife erzeugen (siehe Abbildung 4.15). Exakt wird dann $A = (Z, \Sigma, \delta, z_0, E)$ folgendermaßen konstruiert.

$$Z = Z_\alpha \cup \{z_0\}, \qquad E = \{z_0\}$$

und für die Überführungsfunktion $\delta$ gilt für alle $z \in Z$ und $a \in \Sigma$:

$$\delta(z,a) = \begin{cases} \delta_\alpha(z_{\alpha 0},a) & \text{für } z = z_0, \\ \delta_\alpha(z,a) & \text{für } z \in Z_\alpha \text{ und } \delta_\alpha(z,a) \cap E_\alpha = \emptyset, \\ \delta_\alpha(z,a) \cup \{z_0\} & \text{für } z \in Z_\alpha \text{ und } \delta_\alpha(z,a) \cap E_\alpha \neq \emptyset. \end{cases}$$

Zur Interpretation: Anfangszustand ist also ein neuer Zustand, der auch gleichzeitig der einzige Endzustand ist. Vom Anfangszustand gibt es dann die gleichen Überführungen wie vom Anfangszustand des Automaten $A_\alpha$. Von allen Zuständen von $A_\alpha$, für die es Überführungen zu Endzuständen von $A_\alpha$ gibt, gibt es zusätzlich Überführungen für das gleiche Symbol zum Anfangszustand $z_0$.

Damit wird das leere Wort akzeptiert (wegen $z_0 \in E$) und auch alle Wörter, die auch $A_\alpha$ akzeptiert. Weiter können wir natürlich die Schleife mehrmals durchlaufen, das heißt für die akzeptierte Sprache von $A$ gilt

$$\begin{aligned} T(A) &= \{\varepsilon\} \cup T(A_\alpha) \cup (T(A_\alpha))^2 \cup (T(A_\alpha))^3 \cup \cdots \\ &= (T(A_\alpha))^0 \cup T(A_\alpha)^1 \cup (T(A_\alpha))^2 \cup (T(A_\alpha))^3 \cup \cdots \\ &= (T(A_\alpha))^*, \end{aligned}$$

also $T(A) = (L(\alpha))^*$.

Damit hätten wir alle Fälle behandelt und haben den ersten Teil des Beweises vollendet (abgesehen von den Lücken, die wir hier nicht vollständig bewiesen haben). Das heißt, wir haben gezeigt, zu jedem regulären Ausdruck gibt es einen äquivalenten nichtdeterministischen endlichen Automaten, also ist jede von einem regulären Ausdruck beschriebene Sprache regulär.

Noch eine allgemeine Bemerkung zu den obigen Konstruktionen, natürlich müssen wir $Z_\alpha \cap Z_\beta = \emptyset$ fordern, und falls wir einen Anfangszustand $z_0$ neu hinzunehmen, darf er natürlcih in den gegebenen Automaten nicht vorkommen. Das ist aber keine Einschränkung, da wir durch einfache Umbenennungen der Zustände diese Bedingungen immer absichern können.

*Teil 2:* Im zweiten Teil des Beweises zeigen wir, dass jede reguläre Sprache von einem regulären Ausdruck beschrieben wird.

Sei $L$ eine reguläre Sprache. Dann wird $L$ von einem deterministischen endlichen Automaten $A = (Z, \Sigma, \delta, z_0, E)$ akzeptiert, wobei $\Sigma = \{a_1, a_2, \ldots, a_k\}$ gelte. Für jeden Zustand $z \in Z$ definieren wir die Wortmenge $L_z = \{w \in \Sigma^* \mid \hat{\delta}(z, w) \in E\}$. Für alle diese $L_z$ werden wir zeigen, dass sie von regulären Ausdrücken beschrieben werden können. Wegen $L = T(A) = L_{z_0}$ würde dann die Behauptung folgen.

Die Mengen $L_z$ stehen in folgenden Beziehungen zueinander.

$$\begin{aligned} L_z &= \bigcup_{i=1}^{k} \{a_i\} \cdot L_{\delta(z,a_i)} && \text{für } z \notin E \text{ und} \\ L_z &= \bigcup_{i=1}^{k} \{a_i\} \cdot L_{\delta(z,a_i)} \cup \{\varepsilon\} && \text{für } z \in E. \end{aligned} \tag{4.2}$$

Wir werden jetzt dieses System von Gleichungen, in denen die $L_z$ als Unbekannte auftreten, schrittweise auflösen, wobei wir nur die Operationen Vereinigung, Konkatenation und Iteration verwenden, so dass letztendlich nur $L_{z_0}$ stehen bleibt und somit von einem regulären Ausdruck beschrieben werden kann.

Die Tatsache, dass Unbekannte auf beiden Seiten ein und derselben Gleichung auftreten können, führt dazu, dass hier ein „normales“ Eliminieren nicht möglich ist.

Hier hilft folgendes Lemma,

**Lemma 4.48** *Für $B, C \in \Sigma^*$ gilt: Ist $\varepsilon \notin B$, so ist $L = B^* \cdot C$ die einzige Lösung der Gleichung $L = B \cdot L \cup C$.*

*Beweis.* Der Beweis erfolgt durch das Zeigen der Inklusionen $B^* \cdot C \subseteq L$ sowie $L \subseteq B^* \cdot C$.

*Teil 1:* $B^* \cdot C \subseteq L$. Wir zeigen durch Induktion über $k$, dass jedes $B^k \cdot C$ in jeder Lösung $L$ enthalten ist.

*Induktionsanfang.* Für $k = 0$ gilt:

$$B^0 \cdot C = \{\varepsilon\} \cdot C = C \subseteq B \cdot L \cup C = L.$$

*Induktionsschritt.* Mit der Induktionsvoraussetzung $B^k \cdot C \subseteq L$ schließt man

$$B^{k+1} \cdot C = B \cdot (B^k \cdot C) \subseteq B \cdot L \subseteq B \cdot L \cup C = L.$$

*Teil 2:* $L \subseteq B^* \cdot C$. Wir zeigen durch Induktion über die Länge des Wortes $w$, dass aus $w \in L$ stets $w \in B^* \cdot C$ folgt.

*Induktionsanfang.* Ist $|w| = 0$, so ist $w = \varepsilon$. Gilt $\varepsilon \in L$, so folgt aus $L = B \cdot L \cup C$ und $\varepsilon \notin B$ sofort $\varepsilon \in C$. Dann ist aber auch $\varepsilon \in B^* \cdot C$.

*Induktionsschritt.* Es sei $|w| > 0$, und nach Induktionsvoraussetzung folge für alle $v$ mit $|v| < |w|$ aus $v \in L$ stets $v \in B^* \cdot C$.

Es sei nun $w \in L = B \cdot L \cup C$. Im Falle $w \in C$ folgt sofort $w \in B^* \cdot C$. Im Falle $w \in B \cdot L$ gibt es ein $u \in B$ und ein $v \in L$ mit $w = u \cdot v$. Wegen $\varepsilon \notin B$ gilt $|u| > 0$ und folglich $|v| < |w|$. Nach Induktionsvoraussetzung haben wir damit $v \in B^* \cdot C$, und wir schließen $w = u \cdot v \in B \cdot (B^* \cdot C) \subseteq B^* \cdot C$. □

Mit diesem Lemma können wir jetzt $L_{z_0}$ bestimmen, indem wir schrittweise alle anderen Mengenvaraiblen $L_z$ wie folgt eliminieren. Kommt in der Gleichung, bei der links $L_z$ steht, auch rechts $L_z$ vor, so wird das Lemma angewendet. In jedem Fall hat man dann eine Gleichung $L_z = R$, wobei in $R$ die Variable $L_z$ nicht mehr vorkommt. Ersetzt man jetzt in allen anderen Gleichungen $L_z$ durch $R$, so ist $L_z$ eliminiert.

Wenn wir alle Variablen bis auf $L_{z_0}$ eliminiert haben, ist $L_{z_0}$ dargestellt als endlich oftmalige Anwendung der Operationen Konkatenation, Vereinigung und Iteration über Symbolen aus $\Sigma$. Somit kann $L_{z_0} = T(A)$ durch einen regulären Ausdruck beschrieben werden.

Somit ist der zweite Teil des Satzes von Kleene und damit auch der Satz bewiesen. □

Wir betrachten ein Beispiel für die Konstruktion eines äquivalenten regulären Ausdrucks für einen gegebenen deterministischen endlichen Automaten.

**Beispiel 4.49** In der Abbildung 4.16 ist ein ein DEA über dem Alphabet $\Sigma = \{a, b\}$ durch

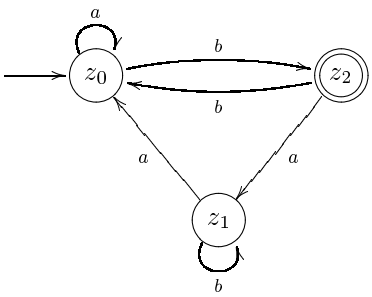


Abbildung 4.16: Ein deterministischer endlicher Automat

einen Überführungsgraphen gegeben. Das dazugehörige Gleichungssystem für die Wortmengen $L_z$ besteht somit aus den drei Gleichungen

$$\left.\begin{aligned} L_{z_0} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_2}, \\ L_{z_1} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_1}, \\ L_{z_2} &= \{a\} \cdot L_{z_1} \cup \{b\} \cdot L_{z_0} \cup \{\varepsilon\}. \end{aligned}\right\} \tag{4.3}$$

Die Variable (Wortmenge) $L_{z_2}$ in der ersten Gleichung können wir ohne Probleme eliminieren, indem wir sie durch die rechte Seite der dritten Gleichung ersetzen. Somit erhalten wir das zu (4.3) äquivalente Gleichungssystem

$$\begin{aligned} L_{z_0} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot (\{a\} \cdot L_{z_1} \cup \{b\} \cdot L_{z_0} \cup \{\varepsilon\}), \\ L_{z_1} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_1} \end{aligned}$$

oder, indem wir die rechte Seite der ersten Gleichung zusammenfassen,

$$\begin{aligned} L_{z_0} &= (\{a\} \cup \{bb\}) \cdot L_{z_0} \cup \{ba\} \cdot L_{z_1} \cup \{b\}, \\ L_{z_1} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_1}. \end{aligned}$$

Nun wenden wir auf die zweite Gleichung das Lemma 4.48 an und erhalten $L_{z_1} = \{b\}^*\{a\} \cdot L_{z_0}$. Damit können wir jetzt $L_{z_1}$ in der ersten Gleichung des Gleichungssystems ersetzen.

$$\begin{aligned} L_{z_0} &= (\{a\} \cup \{bb\}) \cdot L_{z_0} \cup \{ba\} \cdot L_{z_1} \cup \{b\} \\ &= (\{a\} \cup \{bb\}) \cdot L_{z_0} \cup \{ba\} \cdot (\{b\}^*\{a\} \cdot L_{z_0}) \cup \{b\} \\ &= (\{a\} \cup \{bb\} \cup \{ba\} \cdot \{b\}^*\{a\}) \cdot L_{z_0} \cup \{b\}. \end{aligned}$$

Nun wenden wir abermals unser Lemma an und erhalten

$$L_{z_0} = (\{a\} \cup \{bb\} \cup \{ba\} \cdot \{b\}^*\{a\})^*\{b\}.$$

Somit können wir $L_{z_0}$, also die vom gegebenen Automaten akzeptierte Sprache, durch den regulären Ausdruck

$$(a \mid bb \mid bab^*a)^*b$$

beschreiben.

Betrachten wir weiteres Beispiel.

**Beispiel 4.50** Es ist ein regulärer Ausdruck zu bestimmen, der die Menge aller Wörter über dem Alphabet $\{a, b\}$, die nicht das Teilwort *bab* enthalten, beschreibt.

Wir konstruieren zuerst den DEA, der die Menge aller Wörter über dem Alphabet $\{a, b\}$, die das Teilwort *bab* enthalten, beschreibt (siehe Abbildung 4.17). Jetzt können wir daraus sehr einfach

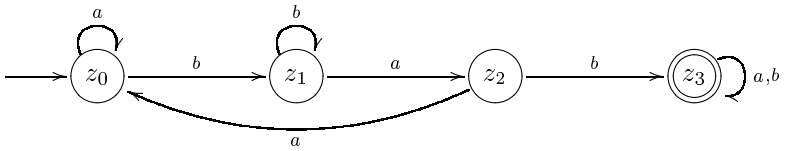


Abbildung 4.17: Ein DEA für die Menge aller Wörter über $\{a, b\}$, die das Teilwort *bab* enthalten

einen DEA konstruieren, der die Komplementärmenge akzeptiert, nämlich durch Vertauschen der akzeptierenden und nichtakzeptierenden Zustände (siehe Abbildung 4.18).

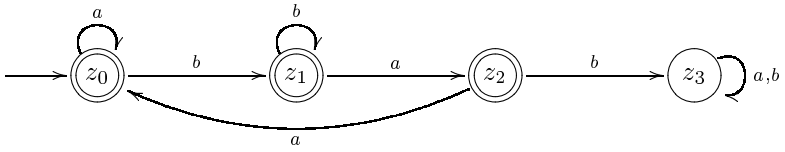


Abbildung 4.18: Ein DEA für die Menge aller Wörter über $\{a, b\}$, die nicht das Teilwort *bab* enthalten

Nun können wir für den DEA aus Abbildung 4.18 das Gleichungssystem der Wortmengen für die einzelnen Zustände aufstellen.

$$\begin{aligned}
L_{z_0} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon\},\\
L_{z_1} &= \{a\} \cdot L_{z_2} \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon\},\\
L_{z_2} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_3} \cup \{\varepsilon\},\\
L_{z_3} &= \{a\} \cdot L_{z_3} \cup \{b\} \cdot L_{z_3}.
\end{aligned}$$

Aus der vierten Gleichung erhalten wir $L_{z_3} = \{a, b\} \cdot L_{z_3}$ und durch Anwendung des Lemmas 4.48 schließlich $L_{z_3} = \{a, b\}^* \cdot \emptyset = \emptyset$. Setzen wir dies in die dritte Gleichung ein, erhalten wir folgendes neue Gleichungssystem.

$$\begin{aligned}
L_{z_0} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon\},\\
L_{z_1} &= \{a\} \cdot L_{z_2} \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon\},\\
L_{z_2} &= \{a\} \cdot L_{z_0} \cup \{\varepsilon\}.
\end{aligned}$$

Mittels der dritten Gleichung können wir $L_{z_2}$ in der zweiten Gleichung ersetzen und erhalten

$$\begin{aligned}
L_{z_1} &= \{a\} \cdot L_{z_2} \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon\}\\
&= \{a\} \cdot (\{a\} \cdot L_{z_0} \cup \{\varepsilon\}) \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon\}\\
&= \{aa\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon, a\}\\
&= \{b\} \cdot L_{z_1} \cup \{aa\} \cdot L_{z_0} \cup \{\varepsilon, a\}.
\end{aligned}$$

Wenden wir nun auf diese Gleichung unser Lemma an, erhalten wir

$$\begin{aligned} L_{z_1} &= \{b\}^* \cdot (\{aa\} \cdot L_{z_0} \cup \{\varepsilon, a\}) \\ &= \{b\}^* \cdot \{aa\} \cdot L_{z_0} \cup \{b\}^* \cdot \{\varepsilon, a\}. \end{aligned}$$

Setzen wir nun dieses Ergebnis in die Gleichung für $L_{z_0}$ ein, so haben wir

$$\begin{aligned} L_{z_0} &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot L_{z_1} \cup \{\varepsilon\} \\ &= \{a\} \cdot L_{z_0} \cup \{b\} \cdot (\{b\}^* \cdot \{aa\} \cdot L_{z_0} \cup \{b\}^* \cdot \{\varepsilon, a\}) \cup \{\varepsilon\} \\ &= (\{a\} \cup \{b\} \cdot \{b\}^* \cdot \{aa\}) \cdot L_{z_0} \cup \{b\} \cdot \{b\}^* \cdot \{\varepsilon, a\} \cup \{\varepsilon\}. \end{aligned}$$

Daraus erhalten wir nach Anwendung unseres Lemmas

$$L_{z_0} = (\{a\} \cup \{b\} \cdot \{b\}^* \cdot \{aa\})^* \cdot (\{b\} \cdot \{b\}^* \cdot \{\varepsilon, a\} \cup \{\varepsilon\}),$$

und somit beschreibt der reguläre Ausdruck

$$(a \mid bb^*aa)^*(bb^*(\varepsilon \mid a) \mid \varepsilon)$$

unsere gegebene Menge der Wörter über dem Alphabet $\{a, b\}$, die nicht das Teilwort *bab* enthalten.

### 4.3.4 Das Pumping Lemma

In diesem Kapitel behandeln wir einen Satz, mit dem man zeigen kann, dass eine Sprache *nicht* regulär ist.

**Satz 4.51 (Pumping Lemma)** *Sei $L$ eine reguläre Sprache. Dann gibt es eine Konstante $k \in \mathbb{N}$, so dass für alle Wörter $z \in L$ mit $|z| \geq k$ Wörter $u$, $v$ und $w$ existieren, so dass gilt:*

(i) $z = uvw$,

(ii) $|uv| \leq k$ *and* $|v| \geq 1$,

(iii) *für alle* $i \in \mathbb{N}$ *gilt* $uv^iw \in L$.

*Beweis.* Sei $L$ eine reguläre Sprache, dann gibt es einen DEA $A$, der $L$ akzeptiert. Sei nun $k$ die Anzahl der Zustände von $A$.

Wir betrachten ein Wort $z$, welches von $A$ akzeptiert wird und für das $|z| \geq k$ gilt. Beim Einlesen von $z$ durchläuft der DEA offensichtlich $|z| + 1 \geq k + 1$ Zustände (einschließlich des Startzustands). Da der Automat aber nur $k$ Zustände hat, muß der DEA $A$ beim Einlesen von $z$ zweimal denselben Zustand durchlaufen, also durchläuft der Automat eine Schleife. Wir bezeichnen mit $u$ den Teil des Wortes, den $A$ vor Erreichen der Schleife eingelesen hat, mit $v$ den Teil, der während der Schleife eingelesen wurde, mit $w$ den Rest des Wortes. Offensichtlich gilt $|uv| \leq k$ und $v \geq 1$.

Da wir eine Schleife durchlaufen haben, kann der Automat natürlich auch die Schleife meiden ($i = 0$) oder zweimal durchlaufen ($i = 2$) oder beliebig oft durchlaufen. Mit anderen Worten, für alle $i \geq 0$ gilt $uv^iw \in L$, was zu beweisen war. □

**Bemerkung 4.52** Schaut man sich die logische Struktur des Pumping Lemmas an, erkennt man, dass es sich um eine Implikation handelt, dass heißt, die Umkehrung *muss nicht* wahr sein. Hier ist es tatsächlich so, es gibt Sprachen, die nicht regulär sind, für die aber die Behauptungen des Pumping Lemmas erfüllt sind (siehe Abbildung 4.19). Damit eignet sich das Pumping Lemma gut, um zu zeigen, dass gewisse Sprachen *nicht regulär* sind, man kann aber auf *keinen Fall* zeigen, dass gewisse Sprachen *regulär* sind!!!.

**Beispiel 4.53** Wir betrachten die Sprache

$$L = \{a^n b^n \mid n \geq 1\},$$

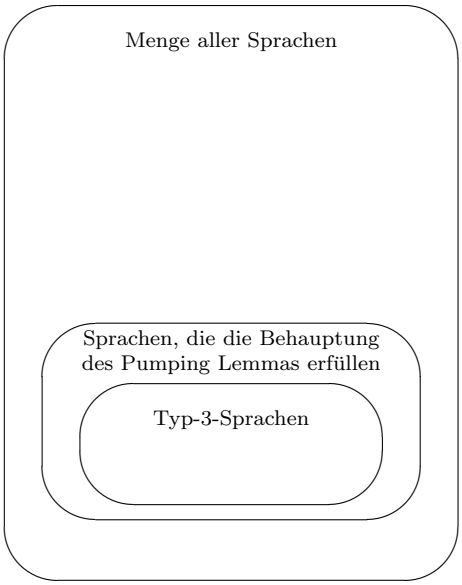


Abbildung 4.19: Hierarchie mit Menge der Sprachen, die die Behauptung des Pumping Lemmas erfüllen

von der wir schon länger vermuten, dass sie nicht regulär ist. Das wollen wir jetzt mit dem Pumping Lemma durch die indirekte Beweismethode beweisen.

Dazu nehmen wir an, $L$ sei regulär. Dann gibt es laut Pumping Lemma eine Konstante $k$, so dass für alle Wörter $z \in L$ mit $|z| \geq k$ Wörter $u$, $v$ und $w$ existieren, so dass gilt:

(i) $z = uvw$,

(ii) $|uv| \leq k$ and $|v| \geq 1$,

(iii) für alle $i \in \mathbb{N}$ gilt $uv^i w \in L$.

Betrachten wir speziell das Wort $z = a^k b^k$, dann gilt offensichtlich $|z| = 2\,k \geq k$, also gibt es eine Zerlegung $z = a^k b^k = uvw$ mit $|uv| \leq k$ and $|v| \geq 1$. Daraus ergibt sich:

$$\begin{aligned} uv &= a^r \quad \text{für } r \leq k, \\ w &= a^{k-r} b^k, \\ v &= a^s \quad \text{für } 1 \leq s \leq r \leq k. \end{aligned}$$

Nach (iii) ist jedes Wort $uv^i w \in L$, also auch

$$uv^2 w \in L. \tag{4.4}$$

Andererseits gilt

$$uv^2 w = uvvw = a^r a^s a^{k-r} b^k = a^{k+s} b^k$$

mit $s \geq 1$, also $k + s \neq k$, das heißt

$$uv^2 w \notin L, \tag{4.5}$$

was einen Widerspruch zu 4.4 darstellt. Folglich war unsere Annahme ($L$ regulär) falsch, also kann $L = \{a^n b^n \mid n \geq 1\}$ *nicht regulär* sein.

**Beispiel 4.54** Mit Hilfe des Pumping Lemmas kann man auch die Nichtregularität der Sprachen

$L = \{a^{n^2} \mid n \geq 1\}$,
$L = \{a^{2^n} \mid n \geq 1\}$,
$L = \{a^p \mid p \text{ Primzahl}\}$,
$L = \{a^n b^n c^n \mid n \geq 1\}$,
$L = \{ww^R \mid w \in \{a, b\}^*\}$,
$L = \{ww \mid w \in \{a, b\}^*\}$,
$L = \text{EXPR}$

und vielen anderen zeigen.

### 4.3.5 Abschlusseigenschaften

Wir wollen in diesem Kapitel untersuchen, unter welchen Operationen die Menge der regulären abgeschlossen ist. Eine Menge $M$ ist *unter der Operation* $\circ$ *abgeschlossen*, falls für alle $a, b \in M$ auch $(a \circ b) \in M$ gilt.

**Satz 4.55** *Die Menge der regulären Sprachen ist unter den Operationen*

(i) *Vereinigung,*

(ii) *Durchschnitt,*

(iii) *Komplement,*

(iv) *Produkt (Konkatenation) und*

(v) *Kleene-Stern*

*abgeschlossen.*

*Beweis.* (i), (iv) und (v): Der Abschluss unter den Operationen Vereinigung, Produkt und Kleene-Stern ist eine direkte Folgerung aus der Definition der regulären Ausdrücke, da mit zwei gegebenen regulären Ausdrücken $\alpha$ und $\beta$ auch $\alpha\beta$, $(\alpha|\beta)$ sowie $(\alpha)^*$ reguläre Ausdrücke sind und somit die beschriebenen Sprachen regulär sind.

(iii): Der Abschluss unter der Operation Komplementbildung wird folgendermaßen nachgewiesen. Sei $L$ eine reguläre Sprache. Dann existiert ein DEA $A = (Z, \Sigma, \delta, z_0, E)$ mit $T(A) = L$. Wir konstruieren den DEA $A' = (Z, \Sigma, \delta, z_0, \Sigma \setminus E)$. Offensichtlich gilt dann $T(A') = \overline{L}$, d. h., durch das Vertauchen von akzeptierenden und nichtakzeptierenden Zuständen akzeptiert $A'$ genau die Wörter, die $A$ nicht akzeptiert hat, also das Komplemt von $L$.

(ii): Offensichtlich gilt

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

(folgt direkt aus Gleichung 1.42 auf Seite 11), damit ist durch den Abschluss unter Komplement und Vereinigung auch der Abschluss unter Durchschnitt gegeben.

Der Abschluss unter Durchschnittsbildung kann allerdings auch direkt durch Konstruktion des sogenannten *Produktautomaten* nachgewiesen werden (siehe z. B. [9]). □

### 4.3.6 Wortproblem und andere Entscheidbarkeitsprobleme

Nach Satz 4.21 ist das Wortproblem für Typ-3-Grammatiken *entscheidbar*, dann natürlich *auch für deterministische endliche Automaten*, denn wir können zu einem deterministischen endlichen Automaten eine äquivalente Typ-3-Grammatik konstruieren. Allerdings weist der Algorithmus, der im Beweis des Satzes 4.21 benutzt wurde, eine exponentielle Laufzeit aus, für praktische Anwendungen nicht brauchbar.

Wir betrachten jetzt das Wortproblem für DEA.

**Gegeben:** DEA $A = (Z, \Sigma, \delta, z_0, E)$, und Wort $w \in \Sigma^*$,

**Frage:** Gilt $w \in T(A)$?

Dann gilt.

**Satz 4.56** *Das Wortproblem für deterministische endliche Automaten ist in linearer Zeit entscheidbar.*

*Beweis.* Der Beweis ist trivial: Nach dem „Einlesen“ des Wortes $w$ wissen wir, ob $w \in T(A)$ gilt oder nicht (je nachdem ob ein akzeptierender Zustand erreicht wurde oder nicht). Und das Einlesen benötogt genau $n$ Schritte, falls das Wort $w$ die Länge $n$ hat. □

Man betrachtet noch andere Entscheidbarkeitsprobleme für deterministische endliche Automaten.

*Leerheitsproblem:*

**Gegeben:** DEA $A$.

**Frage:** Gilt $T(A) = \emptyset$?

*Endlichkeitsproblem:*

**Gegeben:** DEA $A$.

**Frage:** Ist $T(A)$ endlich?

*Schnittproblem:*

**Gegeben:** Zwei DEA's $A_1$ und $A_2$.

**Frage:** Gilt $T(A_1) \cap T(A_2) = \emptyset$?

*Äquivalenzproblem:*

**Gegeben:** Zwei DEA's $A_1$ und $A_2$.

**Frage:** Gilt $T(A_1) = (A_2)$?

Zusammenfassend gilt folgender Satz, den wir aber hier nicht beweisen wollen.

**Satz 4.57** *Das Leerheitsproblem, das Endlichkeitsproblem, das Schnittproblem sowie das Äquivalenzproblem für DEA sind entscheidbar.* □