

Reguläre Sprachen

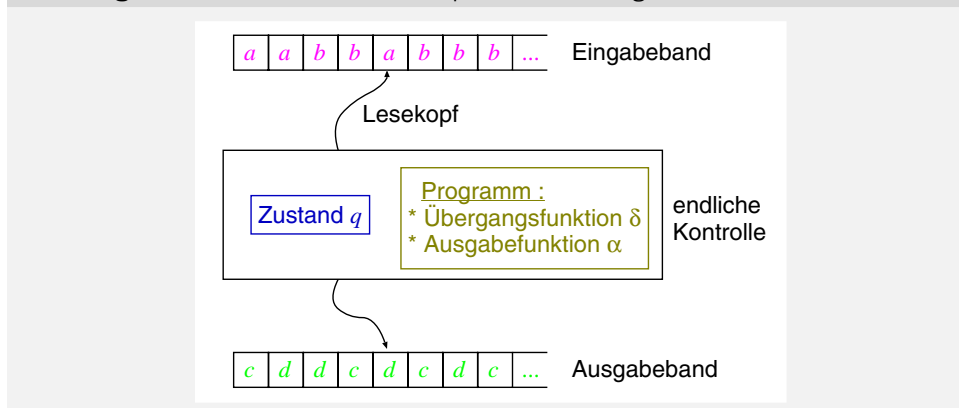
Das Konzept regulärer Sprachen (Sprachen vom Typ 3) wird in vielen Bereichen angewandt. Zu den wichtigsten zählen der Schaltkreisentwurf und die lexikalische Analyse beim Kompilieren von Programmen höherer Programmiersprachen.

Reguläre Sprachen wurden in Abschnitt 5, Seite 195 mithilfe regulärer Grammatiken definiert. In diesem Kapitel stellen wir äquivalente Formalismen für reguläre Sprachen vor: endliche Automaten, reguläre Ausdrücke und Syntaxdiagramme.

6.1 Endliche Automaten

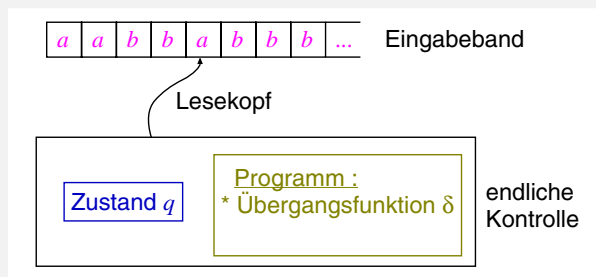
Während für die Hardwarespezifikation endliche Automaten als sequentielle Ein-/Ausgabemaschinen eingesetzt werden (Mealy and Moore-Automaten), werden sie beim Compilerbau als Sprachakzeptoren verwendet.

Abbildung 6.1.1 Endlicher Automat als sequentielle Ein-/Ausgabemaschine



Wir beschäftigen uns hier nur mit endlichen Automaten als *Sprachakzeptoren*. Endliche Automaten können als eine sehr eingeschränkte Variante von Turingmaschinen angesehen werden, die mit einem Eingabeband ausgerüstet sind, auf dem der Lesekopf nur nach rechts bewegt werden kann und denen *kein* weiteres Band zur Verfügung steht¹ (siehe Abbildung 6.1.2).

¹ Obwohl es prinzipiell möglich ist, werden wir endliche Automaten *nicht* als Spezialfall von Turingmaschinen definieren. Der Unterschied wird im Akzeptanzverhalten liegen.

Abbildung 6.1.2 Endlicher Automat als Sprachakzeptor

Ein endlicher Automat besteht also im Wesentlichen nur aus der endlichen Kontrolle (den Zuständen und der Übergangsfunktion) und dem Eingabeband. Das einzige zur Verfügung stehende Speichermedium sind die Zustände.

6.1.1 Deterministische endliche Automaten

Wir betrachten zunächst deterministische endliche Automaten, für die wir die Abkürzung DEA verwenden.² In Abschnitt 6.1.2 werden wir die nichtdeterministische Variante untersuchen.

Definition 6.1.3 [Deterministischer endlicher Automat (DEA)]

Ein DEA ist ein Tupel

$$\mathcal{M} = (Q, \Sigma, \delta, q_0, F),$$

bestehend aus

- ▶ einer endlichen Menge Q von *Zuständen*,
- ▶ einem endlichen Alphabet Σ ,
- ▶ einer partiellen Funktion $\delta : Q \times \Sigma \rightarrow Q$,
- ▶ einem *Anfangszustand* (auch *Startzustand* genannt) $q_0 \in Q$,
- ▶ einer Menge $F \subseteq Q$ von *Endzuständen* (auch *Akzeptanzzustände* genannt).

δ wird auch *Übergangsfunktion* genannt.

Initial steht das Eingabewort auf dem Eingabeband. In jedem Schritt (Konfigurationswechsel) wird der Lesekopf auf dem Eingabeband um eine Position nach rechts verschoben. Sobald das letzte Zeichen der Eingabe gelesen ist, hält die Berechnung an. Eine Berechnung ist entweder akzeptierend (wenn der Automat einen Endzustand erreicht

² Im Englischen ist die Abkürzung DFA gebräuchlich. Sie steht für »deterministic finite automaton«.

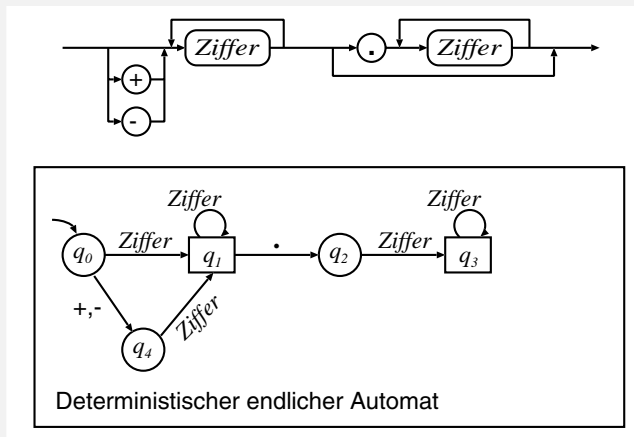
hat) oder verwerfend. Jede Berechnung, in welcher der Automat anhält, ohne das Eingabewort zu Ende gelesen zu haben, ist verwerfend. Man beachte den Unterschied des Akzeptanz- und Terminierungsverhaltens eines DEAs zu dem einer Turingmaschine. Hört die Berechnung verfrüht auf (noch bevor das Eingabewort vollständig gelesen wurde), dann akzeptiert der DEA *nicht*, unabhängig vom erreichten Zustand. Während eine Turingmaschine stets anhält, sobald ein Endzustand erreicht ist, geht die Berechnung eines DEAs in einem Endzustand weiter, sofern die Eingabe noch nicht vollständig gelesen ist.

Zur Darstellung des DEAs verwenden wir Kreise für die Zustände $q \in Q \setminus F$ und Quadrate oder Rechtecke für die Endzustände. Der Anfangszustand ist durch einen kleinen Pfeil markiert.

Beispiel 6.1.4. [DEA für Dezimalzahlen] Die folgende Abbildung 6.1.5 zeigt das Syntaxdiagramm und einen DEA für Dezimalzahlen. Zur Vereinfachung fassen wir das Symbol *Ziffer* als ein Terminalzeichen auf und verwenden das Alphabet

$$\Sigma = \{ \gg + \ll, \gg - \ll, \gg . \ll, \text{Ziffer} \} .$$

Abbildung 6.1.5 DEA mit partieller Übergangsfunktion



Totale Übergangsfunktion: Wie für DTMs können wir die Übergangsfunktion δ durch eine totale Funktion $\hat{\delta}$ ersetzen, die das Akzeptanzverhalten von \mathcal{M} nicht verändert. Dazu müssen wir lediglich \mathcal{M} um einen neuen Fangzustand p erweitern und die Übergangsfunktion wie folgt modifizieren:

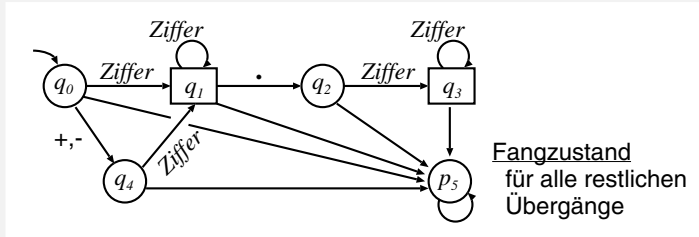
$$\hat{\delta}(q, a) = \begin{cases} \delta(q, a) & : \text{ falls } q \in Q \text{ und } \delta(q, a) \neq \perp \\ p & : \text{ sonst.} \end{cases}$$

Der neue Fangzustand ist kein Akzeptanzzustand.

Aus technischen Gründen werden wir im Folgenden häufig voraussetzen, dass ein DEA mit einer totalen Übergangsfunktion vorliegt.

Der Automat aus Beispiel 6.1.4 hat zunächst eine partielle Übergangsfunktion (z. B. ist $\delta(q_4, +) = \perp$). Er kann durch die Hinzunahme eines Fangzustands p_5 zu einem DEA mit totaler Übergangsfunktion modifiziert werden. ■

Abbildung 6.1.6 DEA mit totaler Übergangsfunktion



Wir formalisieren nun das intuitiv erläuterte Akzeptanzverhalten eines DEAs.

Definition 6.1.7 [Erweiterte Übergangsfunktion für DEAs, akzeptierte Sprache]

Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Wir erweitern δ zu einer (ebenfalls mit δ bezeichneten) partiellen Abbildung

$$\delta : Q \times \Sigma^* \rightarrow Q.$$

Sei $a \in \Sigma, x \in \Sigma^+$ und $q \in Q$. Dann ist

$$\delta(q, \varepsilon) = q, \quad \delta(q, ax) = \begin{cases} \delta(\delta(q, a), x) & : \text{ falls } \delta(q, a) \neq \perp \\ \perp & : \text{ sonst.} \end{cases}$$

Die von \mathcal{M} akzeptierte Sprache ist $\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* : \delta(q_0, w) \in F\}$.

Bemerkung und Definition 6.1.8. [Lauf] Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA und $w = a_1 a_2 \dots a_n \in \Sigma^*$.

- Ist $\delta(q_0, w) \neq \perp$ und ist

$$q_{i+1} = \delta(q_i, a_{i+1}), \quad i = 1, \dots, n - 1,$$

dann nennen wir die Zustandsfolge q_0, \dots, q_n den *Lauf* von \mathcal{M} für w . Ist $q_n \in F$, dann sprechen wir von einem akzeptierenden Lauf, andernfalls von einem verwerfenden Lauf.

- Ist $\delta(q_0, w) = \perp$, dann ist der zu w gehörende Lauf die Folge

$$q_0, q_1, \dots, q_m, \perp,$$

wobei $q_{i+1} = \delta(q_i, a_{i+1}) \in Q$, $i = 1, \dots, m-1$, und $\delta(q_m, a_{m+1}) = \perp$. Dieser ist verwerfend.

Die von \mathcal{M} akzeptierte Sprache ist also genau die Menge aller Wörter $w \in \Sigma^*$, für die der zugehörige Lauf akzeptierend ist. ■

Beispiel 6.1.9. Wir betrachten den DEA aus Beispiel 6.1.4 und das Eingabewort $w = +78.2$.

$$\delta(q_0, +78.2) = \delta(q_4, 78.2) = \delta(q_1, 8.2) = \delta(q_1, .2) = \delta(q_2, 2) = \delta(q_3, \varepsilon) = q_3.$$

Der zu w gehörende Lauf ist die Zustandsfolge

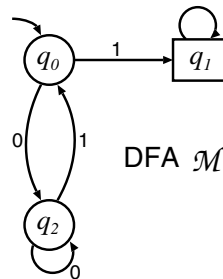
$$q_0, q_4, q_1, q_1, q_2, q_3.$$

Das Wort »+78.2« wird also akzeptiert. Entsprechend ist

$$\delta(q_0, 78 + 9) = \delta(q_1, 8 + 9) = \delta(q_1, +9) = \perp,$$

wenn wir die ursprüngliche (nicht zu einer totalen Funktion erweiterte) Übergangsfunktion betrachten.³ Das Wort »78 + 9« wird also verworfen; der zugehörige Lauf ist q_0, q_1, q_1, \perp .

Als weiteres Beispiel betrachten wir den DEA mit dem Alphabet $\Sigma = \{0, 1\}$ der folgenden Skizze.



Dieser akzeptiert genau diejenigen Wörter über $\{0, 1\}$, die entweder mit einer Eins beginnen oder das Teilwort 011 enthalten. Beispielsweise ist der zu

$$w = 01100$$

gehörende Lauf die Zustandsfolge

$$q_0, q_2, q_0, q_1, q_1, q_1.$$

Diese endet in einem Endzustand. Das Wort w wird also akzeptiert. Der zu dem Wort

$$w' = 0010$$

³ Für die totale Übergangsfunktion ist $\delta(q_0, 78 + 9) = p_5$, da der zugehörige Lauf q_0, q_1, q_1, p_5, p_5 ist.

gehörende Lauf ist die Zustandsfolge

$$q_0, q_2, q_2, q_0, q_2.$$

Diese endet in dem Zustand $q_2 \notin F$; w' wird also nicht akzeptiert. ■

Endliche Automaten und reguläre Grammatiken (1. Teil)

Wir werden sehen, dass DEAs und reguläre Grammatiken dieselbe Ausdrucksstärke haben.

Im Beweis des folgenden Lemmas geben wir ein Verfahren an, wie man zu gegebenem DEA eine reguläre Grammatik konstruieren kann. Dieses belegt, dass DEAs höchstens so ausdrucksstark wie reguläre Grammatiken sind.

Lemma 6.1.10

Zu jedem DEA \mathcal{M} gibt es eine reguläre Grammatik G mit $\mathcal{L}(\mathcal{M}) = \mathcal{L}(G)$.

Beweis: Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA mit einer totalen Übergangsfunktion. Wir fassen die Zustände als Nichtterminale auf. Die Übergangsfunktion entspricht den Regeln einer regulären Grammatik. Wir definieren das Produktionssystem der Grammatik $G = (Q, \Sigma, \mathcal{P}, q_0)$ wie folgt.

- ▶ Ist $q_0 \in F$, dann gilt $q_0 \rightarrow \varepsilon$.
- ▶ Ist $\delta(q, a) = p$, dann gilt $q \rightarrow ap$.
- ▶ Ist $\delta(q, a) = p \in F$, dann gilt $q \rightarrow a$.

Wir zeigen, dass $\mathcal{L}(G) = \mathcal{L}(\mathcal{M})$.

» \supseteq «: Sei $x = a_1 a_2 \dots a_n \in \Sigma^*$ und q_0, q_1, \dots, q_n der zu x gehörende Lauf. Dann gilt:

$$x \in \mathcal{L}(\mathcal{M}).$$

Somit ist der Lauf q_0, \dots, q_n akzeptierend und

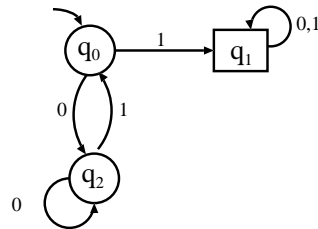
$$q_0 \Rightarrow a_1 q_1 \Rightarrow a_1 a_2 q_2 \Rightarrow \dots \Rightarrow a_1 a_2 \dots a_{n-1} a_{n-1} q_{n-1} \Rightarrow a_1 a_2 \dots a_{n-1} a_n$$

eine Ableitung von x . Also ist

$$x \in \mathcal{L}(G).$$

» \subseteq « folgt mit analogen Argumenten. □

Wir demonstrieren die im Beweis von Lemma 6.1.10 (Seite 226) angegebene Transformation an einem Beispiel. Gegeben sei folgender DEA \mathcal{M} .



Die konstruierte reguläre Grammatik ist durch die Regeln

$$S \rightarrow 0B \mid 1A \mid 1 \quad A \rightarrow 0A \mid 1A \mid 0 \mid 1 \quad B \rightarrow 1S \mid 0B$$

gegeben. Dabei identifizieren wir S mit q_0 , A mit q_1 und B mit q_2 .

6.1.2 Nichtdeterministische endliche Automaten

Wie Turingmaschinen können endliche Automaten um das Konzept von Nichtdeterminismus erweitert werden. Nichtdeterministische endliche Automaten (Abk. NEA) sind bis auf zwei Unterschiede wie DEAs definiert. Erstens lassen wir eine *Menge* von Anfangszuständen zu, unter denen eine nichtdeterministische Auswahl stattfindet. Zweitens ordnet die Übergangsfunktion jedem Paar $(q, a) \in Q \times \Sigma$ eine Menge von möglichen Folgezuständen zu.

Definition 6.1.11 [Nichtdeterministischer endlicher Automat (NEA)]

Ein NEA ist ein Tupel $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$, bestehend aus einer endlichen Menge Q von Zuständen, einem endlichen Alphabet Σ , einer Menge $F \subseteq Q$ von Endzuständen und

- ▶ einer totalen Übergangsfunktion $\delta : Q \times \Sigma \rightarrow 2^Q$,
- ▶ einer Menge $Q_0 \subseteq Q$ von Anfangszuständen.

(Zur Erinnerung: 2^Q bezeichnet die Potenzmenge von Q .)

In Anlehnung an die visuelle Darstellung von NEAs als Digraphen (eventuell mit parallelen Kanten) kann man die Übergangsfunktion eines NEAs auch als Relation $\subseteq Q \times \Sigma \times Q$ auffassen.

Definition 6.1.12 [Erweiterte Übergangsfunktion, akzeptierte Sprache]

Sei $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$ ein NEA. In Analogie zu Definition 6.1.7 (Seite 224) erweitern wir die Übergangsfunktion eines NEAs zu einer (ebenfalls mit δ bezeichneten) Abbildung

$$\delta : 2^Q \times \Sigma^* \rightarrow 2^Q.$$

Intuitiv ist $\delta(P, x)$ die Menge aller Zustände, die man mit dem Wort x von einem Zustand $p \in P$ erreichen kann. Die formale Definition ist wie folgt:

Sei $P \subseteq Q$ und $a \in \Sigma, x \in \Sigma^+$, dann ist $\delta(P, \varepsilon) = P$ und

$$\delta(P, ax) = \bigcup_{p \in P} \delta(\delta(p, a), x).$$

Die von \mathcal{M} akzeptierte Sprache ist $\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* : \delta(Q_0, w) \cap F \neq \emptyset\}$.

In Analogie zu Bemerkung 6.1.8 (Seite 224) können wir $\mathcal{L}(\mathcal{M})$ über die Läufe charakterisieren. Sei $w = a_1 a_2 \dots a_n \in \Sigma^*$ und q_0, q_1, \dots, q_m eine Zustandsfolge mit

$$q_0 \in Q_0 \text{ und } q_{i+1} \in \delta(q_i, a_{i+1}), i = 1, \dots, m - 1.$$

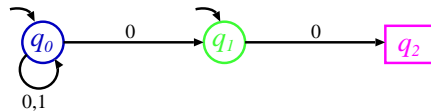
- ▶ Ist $m = n$, dann nennen wir q_0, \dots, q_n einen Lauf von \mathcal{M} für w . Dieser heißt akzeptierend, wenn $q_n \in F$; andernfalls verwerfend.
- ▶ Ist $m < n$ und $\delta(q_m, a_{m+1}) = \emptyset$, dann nennen wir

$$q_0, \dots, q_m, \perp$$

einen verwerfenden Lauf von \mathcal{M} für w .

Die von \mathcal{M} akzeptierte Sprache ist also genau die Menge aller Wörter $w \in \Sigma^*$, für die es einen akzeptierenden Lauf gibt. Im Gegensatz zu DEAs kann ein Wort w viele Läufe in einem NEA haben. Für die Akzeptanz wird lediglich gefordert, dass einer der Läufe für w akzeptierend ist.

Beispiel 6.1.13. [Läufe eines NEAs] Wir betrachten den nachstehenden NEA \mathcal{M} mit dem Alphabet $\Sigma = \{0, 1\}$ und das Wort $w = 0100$.



w hat mehrere Läufe.

q_0, q_0, q_0, q_0, q_0	nicht akzeptierend
q_0, q_0, q_0, q_0, q_1	nicht akzeptierend
q_0, q_0, q_0, q_1, q_2	akzeptierend
q_0, q_1, \perp	nicht akzeptierend
q_1, q_2, \perp	nicht akzeptierend

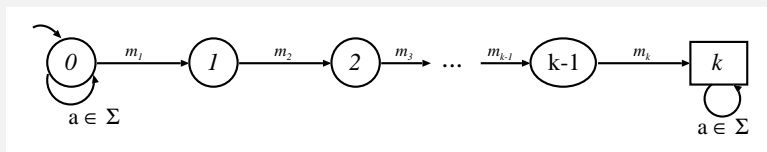
Nur die Existenz eines akzeptierenden Laufs ist entscheidend. Es gilt also $w \in \mathcal{L}(\mathcal{M})$. Man kann zeigen, dass \mathcal{M} genau diejenigen Wörter akzeptiert, die entweder 0 sind oder mit 00 enden. ■

NEAs können auch außerhalb des Übersetzerbaus und Schaltkreisentwurfs nützlich sein. Wir haben bereits gesehen, dass nichtdeterministische Algorithmen oftmals einfacher als entsprechende deterministische Algorithmen sind. In solchen Fällen, in denen ein endlicher Speicher ausreichend ist, können wir nichtdeterministische Algorithmen anhand von NEAs entwerfen (anstelle der sehr viel komplexeren NTMs).

Beispiel 6.1.14. [NEA für Mustererkennung] Wir betrachten das Problem der Mustererkennung, bei dem ein Muster $M = m_1 \dots m_k$ und ein Text $T = t_1 \dots t_n$ gegeben und gefragt ist, ob M in T vorkommt. Für eine algorithmische Lösung ist es ausreichend, nur die maximale Musterposition i , für die das Teilwort $m_1 \dots m_i$ ein Endstück des gelesenen Textteils ist, zu speichern.

Wir nehmen nun an, dass das zu suchende Muster $M = m_1 \dots m_k$ fest ist und betrachten den NEA, der genau das Wort M akzeptiert.

Abbildung 6.1.15 NEA für das Mustererkennungsproblem



Wird nun ein Text $T = t_1 t_2 \dots t_n$ über den Automaten »gescannt«, gibt es genau dann einen akzeptierenden Lauf für T , wenn M ein Teilwort von T ist. ■

Beispiel 6.1.16. [NEA für das Teilsummenproblem] Wir betrachten eine vereinfachte Variante von SUBSUM (s. Abschnitt 4.3.3, Seite 170 ff).

- ▶ Gegeben sind n Zahlen $p_1, \dots, p_n \in \{1, \dots, N\}$.
- ▶ Gefragt ist, ob es eine Teilmenge I von $\{1, \dots, n\}$ gibt, sodass

$$\sum_{i \in I} p_i = N.$$

Zu jedem $N \in \mathbb{N}$ können wir einen NEA entwerfen, der das Problem nichtdeterministisch löst. Intuitiv bearbeitet der Automat die Eingabewerte p_1, \dots, p_n der Reihe nach und entscheidet nichtdeterministisch, die i -te Zahl p_i an der Summe zu beteiligen oder nicht. Wir betrachten folgenden NEA, dessen Zustände die Zahlen

$$q \in \{0, 1, \dots, N\}$$

sind, die jeweils für den Wert der Teilsummen stehen. Der Automat verfügt über einen eindeutigen Anfangs- und Endzustand. Der Anfangszustand ist 0; dies entspricht der initialen Teilsumme 0. Der Endzustand ist N (der gewünschte Wert der Teilsumme).

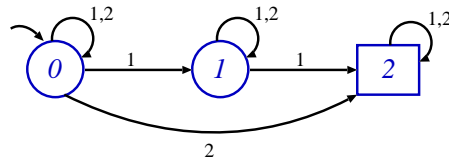
Wir definieren den NEA \mathcal{M} wie folgt:

$$\mathcal{M} = (\{0, 1, \dots, N\}, \{1, 2, \dots, N\}, \delta, \{0\}, \{N\}),$$

wobei

$$\delta(q, a) = \{q, q + a\} \cap \{0, 1, \dots, N\}.$$

Für $N = 2$ hat der NEA folgende Gestalt:



Man überlegt sich leicht, dass die von $\mathcal{L}(\mathcal{M})$ akzeptierte Sprache genau die Menge aller Zahlenfolgen p_1, \dots, p_n ist, für die es eine Teilfolge p_{i_1}, \dots, p_{i_l} gibt, deren Summe N ergibt. ■

Äquivalenz von DEAs und NEAs

Jeder DEA kann als NEA aufgefasst werden. Dies ist wie folgt einsichtig. Ist $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA, dann definieren wir einen NEA wie folgt. Sei

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

wobei

$$\delta'(q, a) = \begin{cases} \{\delta(q, a)\} & : \text{ falls } \delta(q, a) \neq \perp \\ \emptyset & : \text{ sonst.} \end{cases}$$

Offenbar ist jeder Lauf von \mathcal{M}' für w zugleich ein Lauf von \mathcal{M} in w und umgekehrt. Somit gilt

$$\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}').$$

Es stellt sich die Frage, ob NEAs mächtiger sind als DEAs; d.h. ob es Sprachen gibt, die von einem NEA, aber nicht von einem DEA akzeptiert werden können. Das folgende Lemma zeigt, dass dies nicht der Fall ist.

Definition 6.1.17 [Äquivalenz von NEAs]

Seien \mathcal{M}_1 und \mathcal{M}_2 zwei NEAs mit demselben Alphabet Σ . \mathcal{M}_1 und \mathcal{M}_2 heißen *äquivalent*, falls

$$\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2).$$

(Wir fassen DEAs als NEAs auf, sodass der Äquivalenzbegriff zugleich für DEAs definiert ist.)

Lemma 6.1.18

Zu jedem NEA gibt es einen äquivalenten DEA.

Beweis: Sei $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$ ein NEA. Wir wenden die sog. *Potenzmengenkonstruktion* an und definieren einen DEA, dessen Zustände Mengen von Zuständen in \mathcal{M} sind.

$$\mathcal{M}' = (2^Q, \Sigma, \delta', Q_0, F'),$$

wobei $F' = \{P \subseteq Q : P \cap F \neq \emptyset\}$ und

$$\delta'(P, a) = \bigcup_{p \in P} \delta(p, a).$$

Insbesondere ist $\delta'(\emptyset, a) = \emptyset$ für alle $a \in \Sigma$. Man beachte, dass die Übergangsfunktion δ' von \mathcal{M}' total ist. Wir zeigen nun, dass

$$\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}').$$

» \subseteq «: Sei $w = a_1 a_2 \dots a_n \in \mathcal{L}(\mathcal{M})$ und q_0, q_1, \dots, q_n ein akzeptierender Lauf von \mathcal{M} für w . Dann ist $q_0 \in Q_0$, $q_{i+1} \in \delta(q_i, a_{i+1})$, $i = 0, 1, \dots, n-1$, und $q_n \in F$. Sei P_0, P_1, \dots, P_n der zu w gehörende Lauf in \mathcal{M}' . Durch Induktion nach i kann man zeigen, dass

$$q_i \in P_i, \quad i = 0, 1, \dots, n.$$

Insbesondere ist $q_n \in P_n \cap F$ und daher $P_n \cap F \neq \emptyset$. Hieraus folgt $P_n \in F'$. Also ist P_0, P_1, \dots, P_n ein akzeptierender Lauf. Somit ist $w \in \mathcal{L}(\mathcal{M}')$.

» \supseteq «: Sei $w \in \mathcal{L}(\mathcal{M}')$ und $w = a_1 \dots a_n$. Weiter sei P_0, P_1, \dots, P_n der zu w gehörende Lauf in \mathcal{M}' . Dann ist $P_0 = Q_0$, $P_{i+1} = \delta'(P_i, a_{i+1})$, $i = 0, 1, \dots, n-1$, und $P_n \cap F \neq \emptyset$.

► Wir wählen einen beliebigen Zustand $q_n \in P_n \cap F$.

- ▶ Sei $q_{n-1} \in P_{n-1}$ mit $q_n \in \delta(q_{n-1}, a_n)$.
- ▶ Sei $q_{n-2} \in P_{n-2}$ mit $q_{n-1} \in \delta(q_{n-2}, a_{n-1})$.
- ▶ \vdots
- ▶ Sei $q_0 \in P_0$ mit $q_1 \in \delta(q_0, a_1)$.

Beachte: Es gilt

$$P_{n-i+1} = \delta'(P_{n-i}, a_{n-i+1}) = \bigcup_{q \in P_{n-i}} \delta(q, a_{n-i+1}).$$

Daher gibt es zu jedem Zustand $q_{n-i+1} \in P_{n-i+1}$ einen Zustand

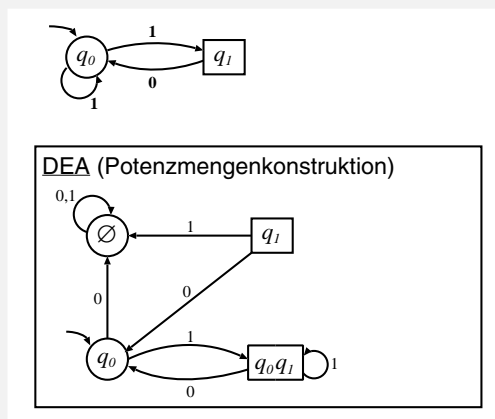
$$q_{n-i} \in P_{n-i} \quad \text{mit} \quad q_{n-i+1} \in \delta(q_{n-i}, a_{n-i+1}).$$

- ▶ Wegen $P_0 = Q_0$ ist q_0 ein Anfangszustand und q_0, q_1, \dots, q_n ein Lauf von \mathcal{M} für w .
- ▶ Wegen $q_n \in F$ ist der Lauf q_0, q_1, \dots, q_n akzeptierend.

Also ist $w \in \mathcal{L}(\mathcal{M})$. □

In Abbildung 6.1.19 betrachten wir ein einfaches Beispiel für die im Beweis von Lemma 6.1.18 angegebene Potenzmengenkonstruktion.

Abbildung 6.1.19 Potenzmengenkonstruktion



Beispielsweise kann der akzeptierende Lauf

$$q_0, q_1, q_0, q_1$$

für das Wort 1010 durch den akzeptierenden Lauf

$$\{q_0\}, \{q_0, q_1\}, \{q_0\}, \{q_0, q_1\}$$

im DEA »simuliert« werden.

Beispiel 6.1.20. [DEA für das Teilsummenproblem] Als weiteres Beispiel betrachten wir die Variante von SUBSUM, wie sie in Beispiel 6.1.16 (Seite 229) vorgestellt wurde. Wird die Potenzmengenkonstruktion auf den dort angegebenen NEA angewandt, liefert sie einen DEA \mathcal{M}' , dessen Zustände Teilmengen von $\{0, 1, \dots, N\}$ sind. Dieser hat folgende Komponenten.

$$\mathcal{M}' = \left(2^{\{0,1,\dots,N\}}, \{0, 1, \dots, N\}, \delta', \{0\}, \{N\} \right),$$

wobei

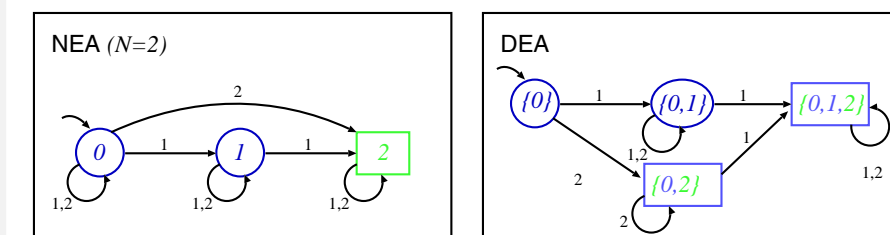
$$\delta'(P, a) = \{p \in \{0, 1, \dots, N\} : p \in P \text{ oder } p - a \in P\}.$$

Das Eingabewort $p_1 p_2 \dots p_n$ hat genau dann einen akzeptierenden Lauf in \mathcal{M}' , wenn

$$\sum_{i \in I} p_i = N$$

für eine Teilmenge I von $\{1, \dots, n\}$.

Abbildung 6.1.21 DEA für SUBSUM ($N = 2$)



Für $N = 2$ erhält man einen DEA mit vier (vom Anfangszustand $\{0\}$) erreichbaren Zuständen (siehe Abb. 6.1.21). ■

Effizienz von NEAs

Die Potenzmengenkonstruktion aus Lemma 6.1.18 (Seite 231) hat einen kleinen Haken. Sie zeigt zwar, dass man jeden NEA in einen äquivalenten DEA umwandeln kann; jedoch entsteht ein exponentiell großer DEA.⁴ Insbesondere ist auch der zeitliche Auf-

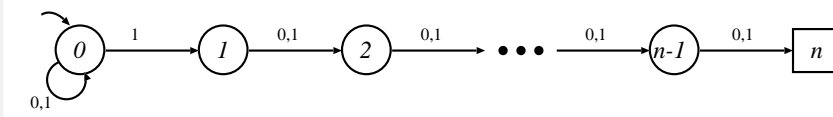
⁴ Der Zustandsraum des DEAs ist die Potenzmenge des Zustandsraums des NEAs. Möglicherweise sind zwar manche dieser Mengen (DEA-Zustände) nicht von dem Anfangszustand Q_0 erreichbar und können daher weggelassen werden; jedoch muss man im Allgemeinen mit exponentiell vielen DEA-Zuständen rechnen.

wand für die Konstruktion exponentiell in der Größe des NEAs. Dennoch lässt sich das exponentielle Blow-Up im Allgemeinen nicht verhindern, da es Sprachen gibt, für die die Darstellung durch einen NEA sehr viel effizienter als durch einen DEA ist. Hierzu betrachten wir die Sprachfamilie $(L_n)_{n \geq 1}$, wobei

$$L_n = \{w \in \{0, 1\}^* : \text{das } n\text{-letzte Zeichen von } w \text{ ist eine } \text{»1«}\}.$$

Die Sprache L_n wird durch einen NEA mit $n + 1$ Zuständen der folgenden Form akzeptiert (siehe Abbildung 6.1.22).

Abbildung 6.1.22 NEA mit $n + 1$ Zuständen



Andererseits hat jeder DEA für L_n mindestens exponentiell viele Zustände. Wir werden dieses Resultat in Abschnitt 6.4.1 beweisen. Siehe Lemma 6.4.14 (Seite 267). Wir greifen diesen Ergebnissen vor und erhalten damit den folgenden Satz:

Satz 6.1.23

Es gibt eine Folge von Sprachen L_n , $n = 1, 2, \dots$, die durch NEAs der Größe $\mathcal{O}(n)$ akzeptiert werden, während jeder DEA für L_n (mindestens) $\Omega(2^n)$ Zustände hat.

Satz 6.1.23 belegt, dass sich die exponentielle worst-case Rechenzeit der Potenzmengenkonstruktion nicht unterbieten lässt.⁵

Endliche Automaten und reguläre Grammatiken (2. Teil)

In Lemma 6.1.10 (Seite 226) haben wir gezeigt, dass die durch einen DEA akzeptierte Sprache durch eine reguläre Grammatik erzeugt wird. Die wesentliche Idee der Konstruktion einer regulären Grammatik bestand darin, die Überföhrungsfunktion als Regel einer Grammatik aufzufassen. Umgekehrt können reguläre Grammatiken über den Umweg eines NEAs in DEAs überföhrt werden. Hierzu konstruieren wir zu gegebener regulärer Grammatik einen NEA, dessen Zustände im Wesentlichen die Nichtterminale sind und dessen Übergänge durch die Regeln der Grammatik gegeben sind.

⁵ Dennoch gibt es bessere Verfahren, die nur den erreichbaren Teil des durch die Potenzmengenkonstruktion gegebenen DEAs generieren und damit *oftmals* eine bessere Laufzeit erzielen.

Lemma 6.1.24

Zu jeder regulären Grammatik G gibt es einen NEA \mathcal{M} mit $\mathcal{L}(G) = \mathcal{L}(\mathcal{M})$.

Beweis: Sei $G = (V, \Sigma, \mathcal{P}, S)$ eine ε -freie reguläre Grammatik.⁶ Wir definieren nun einen NEA \mathcal{M} wie folgt.

- ▶ Die Zustandsmenge ist $Q = V \cup \{q_F\}$, wobei $q_F \notin V$.
- ▶ Die Anfangszustandsmenge ist $Q_0 = \{S\}$.
- ▶ Die Endzustandsmenge ist abhängig davon, ob $\varepsilon \in \mathcal{L}(G)$.

$$F = \begin{cases} \{S, q_F\} & : \text{ falls } S \rightarrow \varepsilon \\ \{q_F\} & : \text{ sonst.} \end{cases}$$

Die Übergangsfunktion δ ist durch folgende beiden »Axiome« gegeben. Seien $A, B \in V$ und $a \in \Sigma$.

$$\begin{aligned} B \in \delta(A, a) & \text{ gdw } A \rightarrow aB \\ q_F \in \delta(A, a) & \text{ gdw } A \rightarrow a \end{aligned}$$

Weiter ist $\delta(q_F, a) = \emptyset$ für alle $a \in \Sigma$.

Dann gilt für alle $w = a_1 a_2 \dots a_n \in \Sigma^+$:

- $w \in \mathcal{L}(\mathcal{M})$
- gdw es existiert ein akzeptierender Lauf A_0, A_1, \dots, A_n von \mathcal{M} für w
- gdw es gibt eine Folge A_0, A_1, \dots, A_{n-1} von Variablen mit $A_0 = S, q_F \in \delta(A_{n-1}, a_n)$ und $A_{i+1} \in \delta(A_i, a_{i+1}), i = 0, \dots, n-2$
- gdw es existieren $A_0, \dots, A_{n-1} \in V$ mit $A_0 = S, A_{n-1} \rightarrow a_n$ und $A_{i+1} \rightarrow a_{i+1} A_i, i = 0, \dots, n-2,$
- gdw es existiert eine Ableitung $S \Rightarrow^* a_1 a_2 \dots a_n = w$
- gdw $w \in \mathcal{L}(G)$

Weiter gilt:

$$\varepsilon \in \mathcal{L}(G) \text{ gdw } S \rightarrow \varepsilon \text{ gdw } S \in F \text{ gdw } \varepsilon \in \mathcal{L}(\mathcal{M}).$$

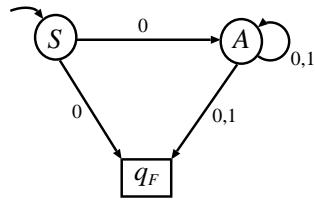
Es folgt $\mathcal{L}(G) = \mathcal{L}(\mathcal{M})$. □

⁶ Mit geringfügigen Modifikationen der in Kapitel 5 (s. Algorithmus 5.1.19, Seite 206) beschriebenen Technik kann jede reguläre Grammatik in eine äquivalente reguläre Grammatik überführt werden, die ε -frei ist.

Beispiel 6.1.25. Wir veranschaulichen die im Beweis von Lemma 6.1.24 angegebene Konstruktion exemplarisch anhand der Grammatik

$$S \rightarrow 0 \mid 0A, \quad A \rightarrow 0 \mid 1 \mid 0A \mid 1A.$$

Diese erzeugt die Sprache bestehend aus allen Wörtern $w \in \{0, 1\}^+$, die mit einer Null beginnen. Der konstruierte NEA hat die Gestalt:



■

Corollar 6.1.26

Sei $L \subseteq \Sigma^*$ eine Sprache. L ist genau dann regulär, wenn es einen DEA (oder NEA) \mathcal{M} mit $L = \mathcal{L}(\mathcal{M})$ gibt.

Corollar 6.1.26 ergibt sich unmittelbar aus den beschriebenen Transformationen:

$$\text{DEA} \Rightarrow \text{reguläre Grammatik} \Rightarrow \text{NEA} \Rightarrow \text{DEA}$$

6.2 Eigenschaften regulärer Sprachen

Bevor wir weitere Charakterisierungen regulärer Sprachen angeben, weisen wir einige Eigenschaften regulärer Sprachen nach, die sich aus der Darstellbarkeit durch endliche Automaten ergeben.

6.2.1 Konstruktion endlicher Automaten

Reguläre Sprachen sind unter allen gängigen Verknüpfungsoperatoren (Vereinigung, Durchschnitt, Konkatenation, Kleeneabschluss und Komplementbildung) abgeschlossen. Wir erläutern, wie sich diese Operationen mit endlichen Automaten realisieren lassen.

Vereinigung: Seien $\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, Q_{0,1}, F_1)$ und $\mathcal{M}_2 = (Q_2, \Sigma, \delta_2, Q_{0,2}, F_2)$ zwei NEAs mit $Q_1 \cap Q_2 = \emptyset$.⁷ Die Grundidee zur Bildung des NEAs für die Vereinigung besteht darin, für ein gegebenes Eingabewort w nichtdeterministisch zu entscheiden, mit

⁷ Man kann die Forderung $Q_1 \cap Q_2 = \emptyset$ durch die Bedingung, dass $\delta_1(q, a) = \delta_2(q, a)$ für alle $q \in Q_1 \cap Q_2$, ersetzen.

welchem der beiden Automaten \mathcal{M}_1 oder \mathcal{M}_2 die Worterkennung durchgeführt wird. Wir definieren

$$\mathcal{M}_1 \uplus \mathcal{M}_2 = (Q_1 \cup Q_2, \Sigma, \delta, Q_{0,1} \cup Q_{0,2}, F_1 \cup F_2),$$

wobei

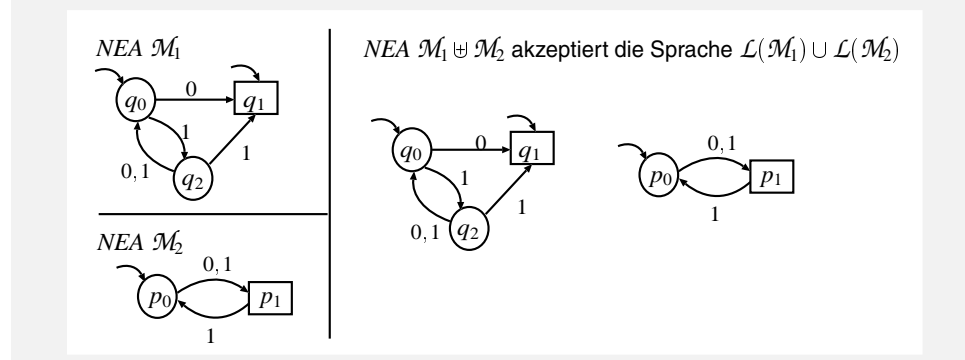
$$\delta(q, a) = \begin{cases} \delta_1(q, a) & : \text{ falls } q \in Q_1 \\ \delta_2(q, a) & : \text{ sonst.} \end{cases}$$

Dann gilt

$$\mathcal{L}(\mathcal{M}_1 \uplus \mathcal{M}_2) = \mathcal{L}(\mathcal{M}_1) \cup \mathcal{L}(\mathcal{M}_2).$$

Die Kosten für die Konstruktion von $\mathcal{M}_1 \uplus \mathcal{M}_2$ sind $\mathcal{O}((|Q_1| + |Q_2|)|\Sigma|)$. Man beachte, dass $\mathcal{M}_1 \uplus \mathcal{M}_2$ kein DEA ist; auch dann nicht, wenn \mathcal{M}_1 und \mathcal{M}_2 deterministisch sind.

Abbildung 6.2.1 Vereinigung von NEAs



Durchschnitt: Seien $\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1)$ und $\mathcal{M}_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$ zwei NEAs. Der *Produktautomat* $\mathcal{M}_1 \times \mathcal{M}_2$ unterliegt der Vorstellung, dass \mathcal{M}_1 und \mathcal{M}_2 parallel geschaltet werden. Ist w das Eingabewort, dann starten wir die synchrone Bearbeitung des Wortes w durch \mathcal{M}_1 und \mathcal{M}_2 . Sobald einer der Automaten frühzeitig verwirft, dann auch der Produktautomat. Nur wenn beide Automaten akzeptieren, dann akzeptiert auch der Produktautomat.

Wir definieren

$$\mathcal{M}_1 \times \mathcal{M}_2 = (Q_1 \times Q_2, \Sigma, \delta, Q_{0,1} \times Q_{0,2}, F_1 \times F_2),$$

wobei

$$\delta(\langle q_1, q_2 \rangle, a) = \{ \langle p_1, p_2 \rangle : p_1 \in \delta_1(q_1, a), p_2 \in \delta_2(q_2, a) \}.$$

Dann gilt

$$\mathcal{L}(\mathcal{M}_1 \times \mathcal{M}_2) = \mathcal{L}(\mathcal{M}_1) \cap \mathcal{L}(\mathcal{M}_2).$$

Wird diese Konstruktion für zwei DEAs $\mathcal{M}_1, \mathcal{M}_2$ durchgeführt, dann entsteht ein DEA, dessen Übergangsfunktion durch die Formel

$$\delta(\langle q_1, q_2 \rangle, a) = \begin{cases} \langle \delta_1(q_1, a), \delta_2(q_2, a) \rangle & : \text{ falls } \delta_1(q_1, a) \neq \perp \text{ und } \delta_2(q_2, a) \neq \perp \\ \perp & : \text{ sonst} \end{cases}$$

gegeben ist. Der Produktautomat von DEAs ist also wieder ein DEA.

Die Konstruktion des Produktautomaten (für NEAs oder DEAs) kann in Zeit

$$\mathcal{O}(|Q_1| \cdot |Q_2| \cdot |\Sigma|)$$

durchgeführt werden.

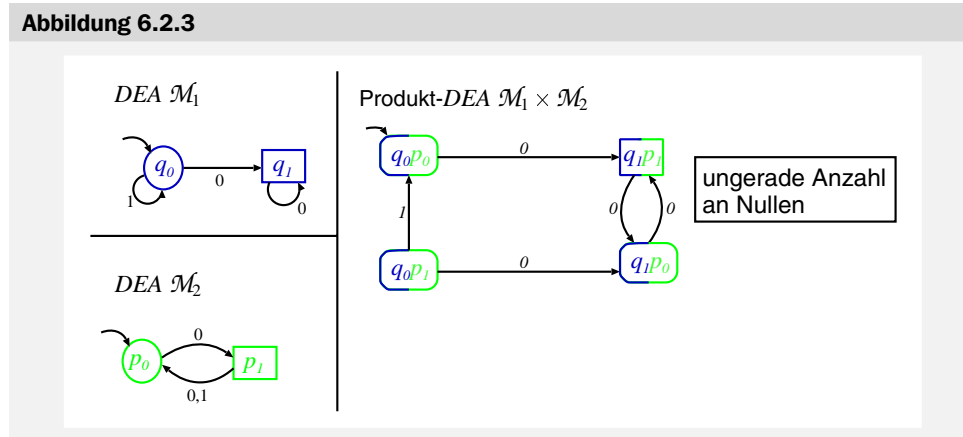
Beispiel 6.2.2. Als Beispiel (siehe Abb. 6.2.3) betrachten wir die zwei folgenden DEAs \mathcal{M}_1 und \mathcal{M}_2 , welche die Sprachen

$$\mathcal{L}(\mathcal{M}_1) = \{1^n 00^m : m, n \geq 0\}$$

$$\mathcal{L}(\mathcal{M}_2) = \{0x_1 0x_2 0 \dots 0x_k 0 : k \geq 0, x_1, \dots, x_k \in \{0, 1\}\}$$

akzeptieren.

Abbildung 6.2.3



Der Produktautomat hat vier Zustände; diese sind Paare, bestehend aus einem Zustand von \mathcal{M}_1 und \mathcal{M}_2 . Der Anfangszustand ist $\langle q_0, p_0 \rangle$. Der Endzustand ist $\langle q_1, p_1 \rangle$.

Man überzeugt sich leicht davon, dass die akzeptierte Sprache genau die Menge aller Wörter 0^{2k+1} , $k \in \mathbb{N}$, ist. Tatsächlich ist die Durchschnittssprache $\mathcal{L}(\mathcal{M}_1) \cap \mathcal{L}(\mathcal{M}_2)$ genau die Menge aller Wörter w , die zugleich die Gestalt $1^n 00^m$ und $0x_1 0 \dots 0x_k 0$, $x_i \in \{0, 1\}$, haben. Also

$$n = 0, x_1 = \dots = x_k = 0 \text{ und } m = 2k$$

und somit $1^n 00^m = 0^{2k+1}$. ■

Komplement: Für den Komplementoperator gehen wir von einem DEA $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ mit einer totalen Übergangsfunktion aus und konstruieren einen DEA für die Komplementsprache

$$\overline{\mathcal{L}(\mathcal{M})} = \Sigma^* \setminus \mathcal{L}(\mathcal{M}).$$

Wir erhalten einen DEA für $\overline{\mathcal{L}(\mathcal{M})}$, indem wir die Endzustandsmenge komplementieren.

$$\overline{\mathcal{M}} = (Q, \Sigma, \delta, q_0, Q \setminus F)$$

Dann ist $\overline{\mathcal{M}}$ ein DEA mit

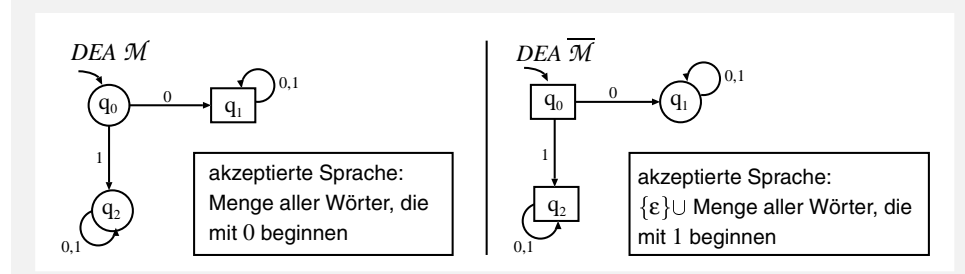
$$\mathcal{L}(\overline{\mathcal{M}}) = \overline{\mathcal{L}(\mathcal{M})}.$$

Man beachte die wesentliche Annahme, dass der vorliegende DEA eine totale Übergangsfunktion hat, da alle Wörter w , für die \mathcal{M} eine »vorzeitig abbrechende« (verwerfende) Berechnung hat, von $\overline{\mathcal{M}}$ akzeptiert werden müssen.

Die Konstruktion des Komplementautomaten erfordert lediglich $\mathcal{O}(|Q|)$ Schritte.

Als Beispiel betrachten wir einen DEA, der alle Wörter über $\{0, 1\}$ akzeptiert, die mit 0 beginnen (siehe Abbildung 6.2.4).

Abbildung 6.2.4 Komplementbildung für DEAs



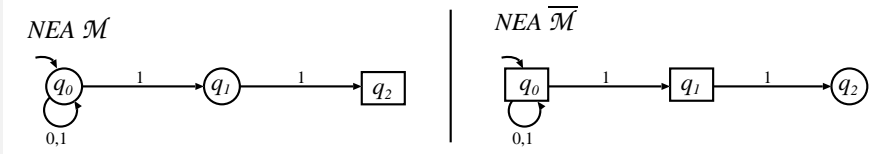
Man beachte, dass die entsprechende Konstruktion für einen NEA fehlschlagen kann. Dieser Sachverhalt ist wegen der Asymmetrie von Akzeptanz und Verwurf in nichtdeterministischen Maschinenmodellen nicht verwunderlich.⁸ Wir machen uns die Aussage an folgendem Beispiel klar (siehe Abbildung 6.2.5, Seite 240).

Der skizzierte NEA \mathcal{M} akzeptiert genau diejenigen Wörter, die das Teilwort 11 enthalten. Der Komplementautomat $\overline{\mathcal{M}}$ akzeptiert jedoch alle Wörter in $\{0, 1\}^*$ (da q_0, q_0, \dots, q_0 stets ein akzeptierender Lauf ist) und nicht nur diejenigen Wörter, die keine zwei Einsen hintereinander enthalten.

Liegt ein NEA vor, dann kann man mit der im Beweis von Lemma 6.1.18 (Seite 231) vorgestellten Potenzmengenkonstruktion einen äquivalenten DEA konstruieren und auf diesen den Komplementoperator anwenden.

⁸ Ein NEA akzeptiert genau dann, wenn es einen akzeptierenden Lauf gibt. Es kann jedoch auch verwerfende Läufe für das betreffende Eingabewort geben. Andererseits wird ein Eingabewort w nur dann von einem NEA verworfen, wenn *alle* Läufe für w verwerfend sind.

Abbildung 6.2.5 Fehlgeschlagene Komplementierung für NEAs



Vereinigung für DEAs: Wir haben erwähnt, dass die oben angegebene Konstruktion für die Vereinigung zunächst einen NEA liefert; auch wenn zwei DEAs verknüpft werden. Liegen zwei DEAs $\mathcal{M}_1, \mathcal{M}_2$ vor, für die ein DEA für die Sprache $\mathcal{L}(\mathcal{M}_1) \cup \mathcal{L}(\mathcal{M}_2)$ erstellt werden soll, so könnte man zwar den Operator \uplus anwenden und dann die Potenzmengenkonstruktion durchführen; jedoch gibt es einen einfacheren Weg, mit dem die exponentiellen Kosten für die Potenzmengenkonstruktion umgangen werden können.

Hierzu wenden wir die de Morgansche Regel

$$\mathcal{L}(\mathcal{M}_1) \cup \mathcal{L}(\mathcal{M}_2) = \overline{\overline{\mathcal{L}(\mathcal{M}_1)} \cap \overline{\mathcal{L}(\mathcal{M}_2)}}$$

an, die es erlaubt, die Vereinigung auf die Komplement- und Durchschnittsbildung zurückzuführen. Die Aussage, dass der DEA

$$\mathcal{M} = \overline{\overline{\mathcal{M}_1} \times \mathcal{M}_2}$$

genau die Vereinigungssprache akzeptiert, kann man sich intuitiv wie folgt klarmachen. Die Endzustände von \mathcal{M} sind genau diejenigen Zustandspaare $\langle q_1, q_2 \rangle$, für die wenigstens einer der beiden Zustände q_1 oder q_2 ein Endzustand ist. Somit akzeptiert \mathcal{M} genau dann, wenn wenigstens einer der Läufe in \mathcal{M}_1 oder \mathcal{M}_2 akzeptierend ist. Tatsächlich ist der Umweg über die Komplementierung unnötig, da es genügt, den Produktautomaten von \mathcal{M}_1 und \mathcal{M}_2 zu bilden und diesen mit der Endzustandsmenge

$$F = \{ \langle q_1, q_2 \rangle : q_1 \in F_1 \vee q_2 \in F_2 \}$$

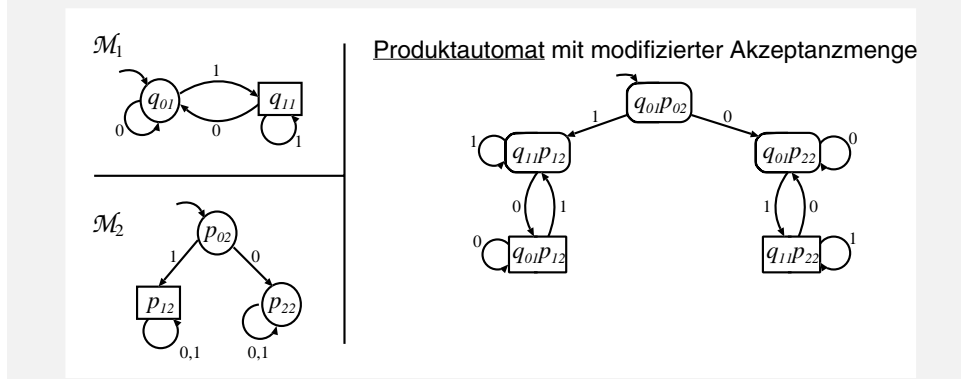
zu versehen. Dabei müssen wir voraussetzen, dass \mathcal{M}_1 und \mathcal{M}_2 jeweils mit einer totalen Übergangsfunktion ausgestattet sind.

Diese Überlegungen zeigen, dass auch die Bildung eines DEAs für die Vereinigung in Zeit

$$\mathcal{O}(|Q_1| \cdot |Q_2| \cdot |\Sigma|)$$

möglich ist. Als Beispiel betrachten wir die in Abbildung 6.2.6 skizzierten DEAs \mathcal{M}_1 und \mathcal{M}_2 .

\mathcal{M}_1 akzeptiert genau diejenigen Wörter $w \in \{0, 1\}^*$, die mit Eins enden. Die von \mathcal{M}_2 akzeptierte Sprache besteht aus allen Wörtern $w \in \{0, 1\}^*$, die mit einer Eins beginnen.

Abbildung 6.2.6 Produktautomat mit modifizierter Akzeptanzmenge

Tatsächlich akzeptiert der DEA

$$\overline{\mathcal{M}_1} \times \overline{\mathcal{M}_2}$$

genau die Sprache aller Wörter, die mit einer Eins beginnen oder enden.

NEAs mit ε -Übergängen

Als technisches Hilfsmittel zur Behandlung von Konkatenation und Kleeneabschluss betrachten wir eine Erweiterung von NEAs, in denen ε -Übergänge möglich sind. ε -Übergänge sind *spontane Zustandsveränderungen*, die unabhängig vom Zeichen unter dem Lesekopf stattfinden und die die Position des Lesekopfs unverändert lassen. Zu jedem Zeitpunkt können beliebig viele ε -Übergänge stattfinden.

Definition 6.2.7 [NEAs mit ε -Übergängen]

Ein ε -erweiterter NEA ist ein Tupel $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$, dessen Komponenten Q , Σ , Q_0 und F wie in einem NEA definiert sind und dessen Übergangsfunktion eine Funktion des Typs

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$$

ist.

Wir formalisieren das Verhalten ε -erweiterter NEAs durch die Erweiterung der Übergangsfunktion zu einer Abbildung

$$\delta_\varepsilon : 2^Q \times \Sigma^* \rightarrow 2^Q.$$

Sei $P \subseteq Q$. Wir setzen

$$\delta(P, \varepsilon) = \bigcup_{p \in P} \delta(p, \varepsilon).$$

Zunächst definieren wir den ε -Abschluss $\Delta(P)$. Dieser enthält genau diejenigen Zustände, die von P durch beliebig viele (0 oder mehrere) ε -Übergänge erreicht werden können.

$$\Delta(P) = \bigcup_{i \geq 0} \Delta_i(P), \quad \text{wobei } \Delta_0(P) = P \text{ und } \Delta_{i+1}(P) = \Delta_i(P) \cup \delta(\Delta_i(P), \varepsilon).$$

$\Delta_i(P)$ ist die Menge aller Zustände $q \in Q$, die durch höchstens i ε -Übergänge von einem Zustand $p \in P$ erreichbar sind. Wir definieren nun

$$\delta_\varepsilon(P, \varepsilon) = \Delta(P), \quad \delta_\varepsilon(P, ax) = \bigcup_{q \in \Delta(P)} \delta_\varepsilon(\delta(q, a), x),$$

wobei $a \in \Sigma$ und $x \in \Sigma^*$. Ist $P = \{p\}$ einelementig, dann schreiben wir auch $\delta_\varepsilon(p, a)$ anstelle von $\delta_\varepsilon(\{p\}, a)$. Entsprechend steht $\Delta(p)$ für $\Delta(\{p\})$.

Es gilt $q \in \delta_\varepsilon(P, a)$ genau dann, wenn es einen Zustand $p \in P$ gibt, von dem der Automat durch beliebig viele (0 oder mehrere) ε -Übergänge, gefolgt von einem a -Übergang und beliebig vielen ε -Übergängen, in den Zustand q wechseln kann. Weiter gilt

$$q \in \delta_\varepsilon(P, a_1 \dots a_n)$$

genau dann, wenn es eine Folge q_0, q_1, \dots, q_n von Zuständen gibt, sodass

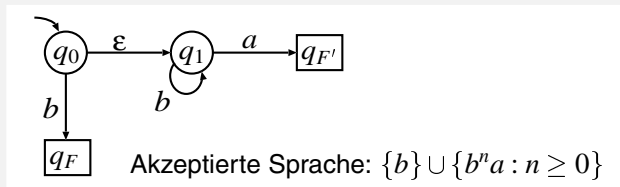
- ▶ $q_0 \in P$ und
- ▶ $q_{i+1} \in \delta_\varepsilon(q_i, a_{i+1}), i = 0, 1, \dots, n$.

Die durch einen ε -erweiterten NEA \mathcal{M} akzeptierte Sprache ist

$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* : \delta_\varepsilon(Q_0, w) \cap F \neq \emptyset\}.$$

Als Beispiel betrachten wir einen ε -erweiterten NEA, der die Sprache $L = \{b\} \cup \{b^n a : n \in \mathbb{N}\}$ akzeptiert.

Abbildung 6.2.8 NEA mit ε -Übergängen



Offenbar kann jeder NEA (und damit auch jeder DEA) als ε -erweiterter NEA interpretiert werden. Wir müssen dazu lediglich die Übergangsfunktion δ eines NEAs durch

$\delta(q, \varepsilon) = \emptyset$ erweitern. ε -erweiterte NEAs sind daher mindestens so ausdrucksstark wie NEAs oder DEAs. Das folgende Lemma belegt, dass auch die Umkehrung gilt; d.h. dass jede von einem ε -erweiterten NEA akzeptierte Sprache durch einen NEA (oder DEA) repräsentiert werden kann.

Lemma 6.2.9

Zu jedem ε -erweiterten NEA gibt es einen NEA \mathcal{M}' mit $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}')$.

Beweis: Sei $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$ ein ε -erweiterter NEA. Wir definieren einen NEA

$$\mathcal{M}' = (Q, \Sigma, \delta', \Delta(Q_0), F)$$

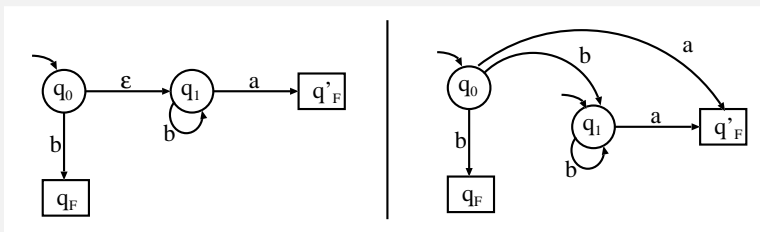
wie folgt.

$$\delta'(q, a) = \delta_\varepsilon(q, a)$$

für alle $q \in Q$ und $a \in \Sigma$. Es ist leicht zu sehen, dass $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}')$. \square

Wir demonstrieren die in Lemma 6.2.9 angegebene Konstruktion an einem Beispiel.

Abbildung 6.2.10 Äquivalenter NEA ohne ε -Übergänge



Die Erweiterung von NEAs durch ε -Übergänge ermöglicht recht einfache Verknüpfungen zur Realisierung von Konkatenation und Kleeneabschluss. In beiden Fällen gehen wir von NEAs (mit oder ohne ε -Übergängen) aus und konstruieren einen NEA mit ε -Übergängen. Dieser kann mit der im Beweis von Lemma 6.2.9 (Seite 243) angegebenen Methode zu einem gewöhnlichen NEA transformiert werden.

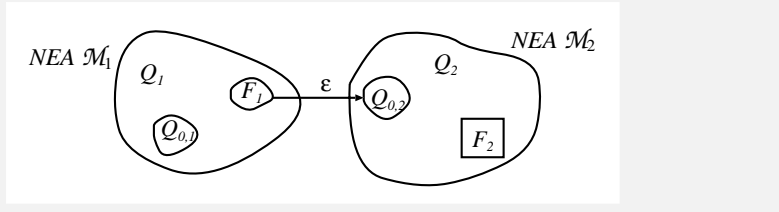
Konkatenation und Kleeneabschluss

Konkatenation: Seien $\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, Q_{0,1}, F_1)$ und $\mathcal{M}_2 = (Q_2, \Sigma, \delta_2, Q_{0,2}, F_2)$ zwei NEAs (mit oder ohne ε -Übergängen). Wir können o.E. annehmen, dass $Q_1 \cap Q_2 = \emptyset$. Der ε -erweiterte NEA

$$\mathcal{M}_1 \circ \mathcal{M}_2 = (Q_1 \cup Q_2, \Sigma, \delta, Q_{0,1}, F_2)$$

verbindet die Endzustände von \mathcal{M}_1 mit den Anfangszuständen von \mathcal{M}_2 durch einen ε -Übergang.

Abbildung 6.2.11 Konkatenation zweier NEAs



Wir formalisieren dies durch folgende Definition der Übergangsfunktion δ :

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & : \text{ falls } q \in Q_1 \text{ und } (q \notin F_1 \text{ oder } a \neq \varepsilon) \\ \delta_2(q, a) & : \text{ falls } q \in Q_2 \\ Q_{0,2} \cup \delta(q, \varepsilon) & : \text{ falls } q \in F_1 \text{ und } a = \varepsilon \end{cases}$$

Dabei ist $a \in \Sigma \cup \{\varepsilon\}$. Die Konstruktion von $\mathcal{M}_1 \circ \mathcal{M}_2$ ist in Zeit $\mathcal{O}((|Q_1| + |Q_2|) \cdot |\Sigma|)$ möglich.

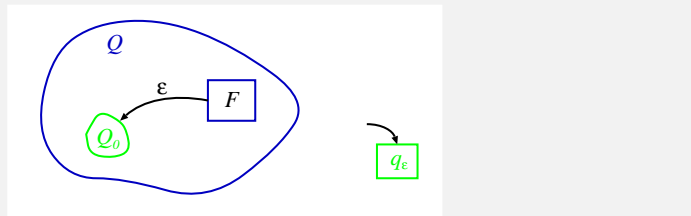
Kleeneabschluss: Sei $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$ ein NEA mit oder ohne ε -Übergängen. Sei $q_\varepsilon \notin Q$.

Wir definieren einen ε -erweiterten NEA

$$\mathcal{M}^* = (Q \cup \{q_\varepsilon\}, \Sigma, \delta^*, Q_0 \cup \{q_\varepsilon\}, F \cup \{q_\varepsilon\}),$$

dessen Übergangsfunktion δ^* die Endzustände über einen ε -Übergang mit den Anfangszuständen verbindet.

Abbildung 6.2.12 Kleeneabschluss eines NEAs



Falls $a \in \Sigma$ und $q \in Q$, so ist $\delta^*(q, a) = \delta(q, a)$. Die ε -Übergänge der Zustände $q \in Q$ sind durch

$$\delta^*(q, \varepsilon) = \begin{cases} Q_0 \cup \delta(q, \varepsilon) & : \text{ falls } q \in F \\ \delta(q, \varepsilon) & : \text{ falls } q \in Q \setminus F \end{cases}$$

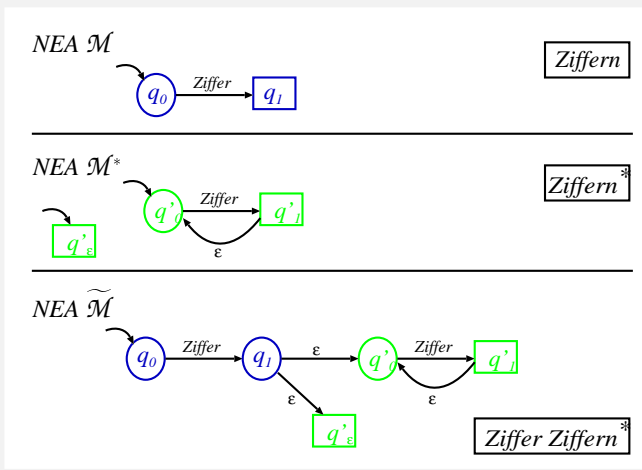
gegeben. Der Spezialzustand q_ε wird benötigt, um sicherzustellen, dass das leere Wort akzeptiert wird.⁹ Wir definieren q_ε als einen Zustand ohne Übergänge, also $\delta^*(q_\varepsilon, a) = \emptyset$ für alle $a \in \Sigma \cup \{\varepsilon\}$. Es ist leicht zu sehen, dass

$$\mathcal{L}(\mathcal{M}^*) = \mathcal{L}(\mathcal{M})^*.$$

Für die Konstruktion von \mathcal{M}^* sind $\mathcal{O}(|Q| \cdot |\Sigma|)$ Schritte ausreichend.

Beispiel 6.2.13. Als Beispiel betrachten wir die Konstruktion eines NEAs für die Menge aller nicht leeren Ziffernfolgen.

Abbildung 6.2.14 Beispiel zu Konkatination und Kleeneabschluss



Die Konstruktion eines NEAs mit ε -Übergängen für die Sprache

$$\mathcal{L}(\mathcal{M})^+ = \mathcal{L}(\mathcal{M}) \circ \mathcal{L}(\mathcal{M})^*$$

kann man (wie in obigem Beispiel) durch Anwenden des Operators für den Kleeneabschluss, gefolgt vom Konkatinationsoperator, vornehmen. Tatsächlich ist jedoch eine einfachere Konstruktion möglich. Dazu definieren wir einen ε -erweiterten NEA

$$\mathcal{M}^+ = (Q, \Sigma, \delta^+, Q_0, F)$$

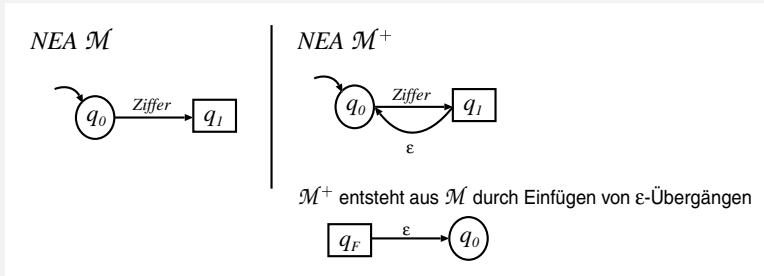
wie folgt. Intuitiv entsteht \mathcal{M}^+ aus \mathcal{M} , indem von allen Endzuständen ein ε -Übergang zu den Anfangszuständen eingefügt wird. Falls $a \in \Sigma$, so ist $\delta^+(q, a) = \delta(q, a)$. Die ε -Übergänge sind durch

$$\delta^+(q, \varepsilon) = \begin{cases} Q_0 \cup \delta(q, \varepsilon) & : \text{ falls } q \in F \\ \delta(q, \varepsilon) & : \text{ sonst} \end{cases}$$

⁹ Man kann auf q_ε verzichten, falls $\varepsilon \in \mathcal{L}(\mathcal{M})$. Ist $\varepsilon \in \mathcal{L}(\mathcal{M})$, dann ist $\mathcal{L}(\mathcal{M})^+ = \mathcal{L}(\mathcal{M})^*$ und man kann auf die einfachere Konstruktion \mathcal{M}^+ (siehe unten) zurückgreifen.

gegeben. Offenbar gilt dann $\mathcal{L}(\mathcal{M}^+) = \mathcal{L}(\mathcal{M})^+$. Als Beispiel betrachten wir nochmals die Konstruktion eines NEAs für die Sprache der nicht leeren Ziffernfolgen (siehe Abbildung 6.2.15). ■

Abbildung 6.2.15 Vereinfachte Konstruktion eines NEAs für $\mathcal{L}(\mathcal{M})^+$



Die Ergebnisse dieses Abschnitts liefern folgenden Satz.

Satz 6.2.16

Die Klasse der regulären Sprachen ist unter Vereinigung, Konkatenation, Kleeneabschluss, Komplementbildung und Durchschnitt abgeschlossen.

6.2.2 Algorithmen für den Nachweis von Eigenschaften regulärer Sprachen

In engem Zusammenhang zur Konstruktion von endlichen Automaten für gegebene reguläre Sprachen stehen Analysealgorithmen, bei denen Verfahren für das Wortproblem und der Äquivalenztest zu den wichtigsten Fragestellungen zählen.

Das Wortproblem: Ist \mathcal{M} ein DEA mit dem Alphabet Σ und w ein Wort über Σ , dann kann die Frage

$$\text{»Gilt } w \in \mathcal{L}(\mathcal{M})? \text{«}$$

in linearer Zeit $\mathcal{O}(|w|)$ beantwortet werden. Wir müssen lediglich den DEA \mathcal{M} bei Eingabe w simulieren.

Äquivalenztest: Die Frage, ob zwei DEAs \mathcal{M}_1 und \mathcal{M}_2 äquivalent sind, d.h.

$$\text{»Gilt } \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)? \text{«},$$

lässt sich auf das Inklusionsproblem reduzieren:

Es gilt

$$\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \text{ gdw } \mathcal{L}(\mathcal{M}_1) \subseteq \mathcal{L}(\mathcal{M}_2) \text{ und } \mathcal{L}(\mathcal{M}_2) \subseteq \mathcal{L}(\mathcal{M}_1).$$

Für den Inklusionstest können wir folgende Beobachtung ausnutzen:

$$\mathcal{L}(\mathcal{M}_1) \subseteq \mathcal{L}(\mathcal{M}_2) \text{ gdw } \mathcal{L}(\mathcal{M}_1) \cap \overline{\mathcal{L}(\mathcal{M}_2)} = \emptyset \text{ gdw } \mathcal{L}(\mathcal{M}_1 \times \overline{\mathcal{M}_2}) = \emptyset$$

Zur Beantwortung der Frage »Gilt $\mathcal{L}(\mathcal{M}_1) \subseteq \mathcal{L}(\mathcal{M}_2)$?« können wir also wie folgt verfahren. Wir bilden den Produktautomaten aus \mathcal{M}_1 und aus dem Komplementautomaten von \mathcal{M}_2 und führen für diesen den Leerheitstest durch.

Leerheitstest: Die Frage

$$\text{»Gilt } \mathcal{L}(\mathcal{M}) = \emptyset? \text{«}$$

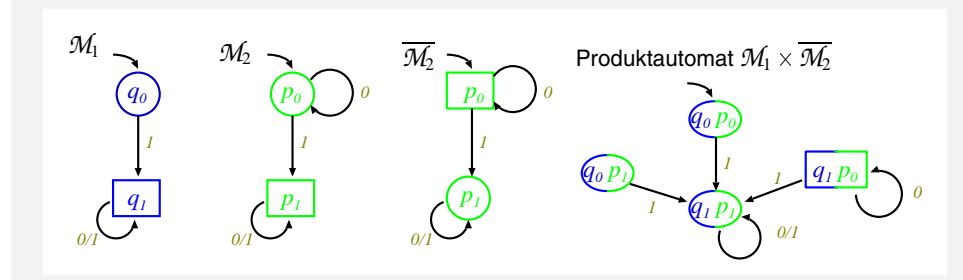
für einen DEA (oder NEA) \mathcal{M} lässt sich mit einer Erreichbarkeitsanalyse realisieren. Wir müssen lediglich prüfen, ob einer der Endzustände vom Anfangszustand (oder einem der Anfangszustände) erreichbar ist. Dazu können wir eine Tiefen- oder Breitensuche anwenden. Der zu Grunde liegende Digraph ist $\mathcal{G} = (Q, E)$, wobei

$$(q, q') \in E \text{ genau dann, wenn } \delta(q, a) = q' \text{ für ein } a \in \Sigma.$$

Es ist klar, dass der zeitliche Aufwand für den Leerheitstest $\mathcal{O}(|Q| \cdot |\Sigma|)$ ist. Die Laufzeit für den Inklusions- und Äquivalenztest wird durch die Bildung des Produktautomaten und der Erreichbarkeitsanalyse dominiert. Daher können der Inklusions- und Äquivalenztest in Zeit $\mathcal{O}(|Q_1| \cdot |Q_2| \cdot |\Sigma|)$ durchgeführt werden.

Wir illustrieren nun das für den Inklusionstest skizzierte Verfahren am Beispiel eines DEAs \mathcal{M}_1 für die Sprache $\{1x : x \in \{0, 1\}^*\}$ und eines DEAs \mathcal{M}_2 für die Sprache aller Wörter $w \in \{0, 1\}^*$, die mindestens eine Eins enthalten (vgl. Abbildung 6.2.17).

Abbildung 6.2.17 Produktautomat $\mathcal{M}_1 \times \overline{\mathcal{M}_2}$



Der Produktautomat $\mathcal{M}_1 \times \overline{\mathcal{M}_2}$ hat vier Zustände. Der Endzustand $\langle q_1, p_0 \rangle$ ist nicht vom Anfangszustand $\langle q_0, p_0 \rangle$ erreichbar. Daher ist

$$\mathcal{L}(\mathcal{M}_1 \times \overline{\mathcal{M}_2}) = \emptyset.$$

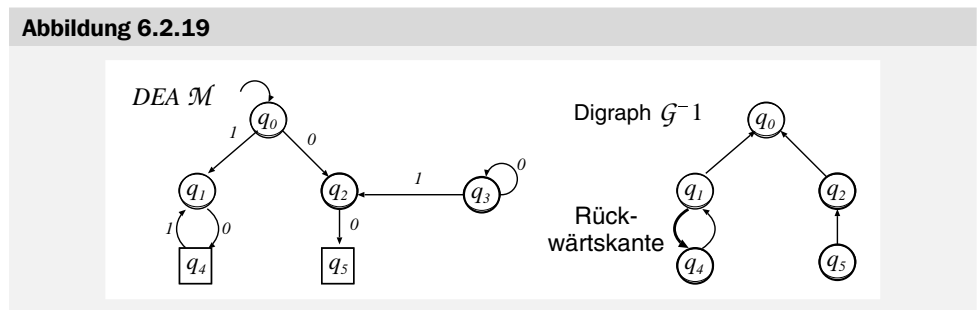
Wir erhalten das offenkundige Resultat $\mathcal{L}(\mathcal{M}_1) \subseteq \mathcal{L}(\mathcal{M}_2)$.

Endlichkeitstest: Um die Frage zu beantworten, ob die durch einen DEA \mathcal{M} akzeptierte Sprache endlich ist, müssen wir prüfen, ob von dem Anfangszustand ein Zyklus erreich-

bar ist, von dem wiederum ein Pfad zu einem akzeptierender Zustand führt. Letztendlich können wir hierzu wohlbekannte Graphalgorithmen anwenden.

1. Zunächst entfernen wir sämtliche vom Anfangszustand q_0 nicht erreichbaren Zustände. (Zum Beispiel können wir mit einer Tiefen- oder Breitensuche die von q_0 erreichbaren Zustände berechnen.)
2. Dann bilden wir den inversen Graphen \mathcal{G}^{-1} , der aus \mathcal{G} entsteht, indem alle Kanten umgedreht werden.¹⁰
3. Der Digraph \mathcal{G}^{-1} enthält genau dieselben Zyklen, durchläuft sie aber in entgegengesetzter Richtung. Wir prüfen nun, ob ein Endzustand in \mathcal{G}^{-1} einen Zyklus erreichen kann. Hierzu können wir bekannte DFS-Kantenklassifizierungsalgorithmen anwenden. Ein Zyklus liegt genau dann vor, wenn es Rückwärtskanten gibt.

Beispiel 6.2.18. Wir verdeutlichen diese Aussage an einem Beispiel: Gegeben sei folgender DEA \mathcal{M} für die Sprache $\{00\} \cup \{(10)^n : n \geq 1\}$ (vgl. Abbildung 6.2.19).



Im ersten Schritt stellen wir fest, dass der Zustand q_3 nicht von q_0 erreichbar ist. Wir entfernen den Knoten q_3 und analysieren den Digraphen \mathcal{G}^{-1} .

Die Tiefensuche mit dem Endzustand q_5 gestartet, liefert keine Rückwärtskanten, wohl aber die mit q_4 gestartete Tiefensuche. Hieraus folgt die Unendlichkeit der Sprache $\mathcal{L}(\mathcal{M})$. ■

Es ist klar, dass der zeitliche Aufwand für das skizzierte Verfahren durch die Erreichbarkeitsanalysen in \mathcal{G} und \mathcal{G}^{-1} dominiert wird. Daher ist die Laufzeit dieses Verfahrens linear in der Anzahl an Kanten und Knoten in \mathcal{G} (bzw. \mathcal{G}^{-1}) und kann mit $\mathcal{O}(|Q| \cdot |\Sigma|)$ nach oben abgeschätzt werden.

¹⁰ Dabei ist $\mathcal{G} = (Q, E)$ wie oben, also $(q, q') \in E$, falls $\delta(q, a) = q'$ für ein $a \in \Sigma$. Die Annahme, dass alle vom Anfangszustand q_0 nicht erreichbaren Zustände eliminiert wurden, stellt sicher, dass jeder Zustand $q \in Q$ über einen Pfad in \mathcal{G} von q_0 erreichbar ist. Der Digraph \mathcal{G}^{-1} hat (wie \mathcal{G}) die Knotenmenge Q und ist mit der Kantenrelation $E^{-1} = \{(q', q) : (q, q') \in E\}$ ausgerüstet.

Wir fassen zusammen:

Satz 6.2.20

Seien $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$, $\mathcal{M}_i = (Q_i, \Sigma, \delta_i, q_{0,i}, F_i)$, $i = 1, 2$, DEAs und $w \in \Sigma^*$.

- (a) Man kann in Zeit $\mathcal{O}(|Q| \cdot |\Sigma|)$ entscheiden, ob $\mathcal{L}(\mathcal{M}) = \emptyset$.
- (a) Man kann in Zeit $\mathcal{O}(|Q| \cdot |\Sigma|)$ entscheiden, ob $\mathcal{L}(\mathcal{M})$ endlich ist.
- (c) Man kann in Zeit $\mathcal{O}(|w|)$ entscheiden, ob $w \in \mathcal{L}(\mathcal{M})$.
- (d) Man kann in Zeit $\mathcal{O}(|Q_1| \cdot |Q_2| \cdot |\Sigma|)$ prüfen, ob $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$.

6.2.3 Das Pumping Lemma für reguläre Sprachen

Der folgende Satz (in der Literatur als Pumping Lemma bekannt) präsentiert ein notwendiges Kriterium für reguläre Sprachen und kann für den Nachweis genutzt werden, dass eine Sprache *nicht* regulär ist.

Satz 6.2.21 [Pumping Lemma für reguläre Sprachen]

Sei $L \subseteq \Sigma^*$ eine reguläre Sprache. Dann gibt es eine ganze Zahl $n \geq 1$, sodass jedes Wort $x \in L$ mit $|x| \geq n$ wie folgt zerlegt werden kann: $x = uvw$ mit Wörtern $u, v, w \in \Sigma^*$, sodass

- (1) $|v| \geq 1$
- (2) $|uv| \leq n$
- (3) $uv^k w \in L$ für alle $k \in \mathbb{N}$.

Beweis: Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA mit $\mathcal{L}(\mathcal{M}) = L$ (Corollar 6.1.26) und $|Q| = n$. Wir zeigen nun, dass jedes Wort in $\mathcal{L}(\mathcal{M})$ der Länge $\geq n$ zerlegt werden kann, sodass die Bedingungen (1), (2) und (3) erfüllt sind.

Sei $x = a_1 a_2 \dots a_m \in \mathcal{L}(\mathcal{M})$ mit $|x| = m \geq n$. Sei q_0, q_1, \dots, q_m der zu x gehörende Lauf von \mathcal{M} . Dann ist

$$q_m \in F.$$

Wir betrachten nur die ersten $n + 1$ Zustände des Laufs. Da \mathcal{M} genau n Zustände hat, gibt es Indizes i, j mit $0 \leq i < j \leq n$, sodass $q_i = q_j$. Wir zerlegen x wie folgt:

$$u = a_1 a_2 \dots a_i, \quad v = a_{i+1} a_{i+2} \dots a_j, \quad w = a_{j+1} a_{j+2} \dots a_m.$$

Wir weisen nun die geforderten Eigenschaften (1), (2) und (3) nach.

- (1) Wegen $i < j$ gilt $|v| \geq 1$.
- (2) Wegen $j \leq n$ gilt $|uv| = |a_1 \dots a_j| = j \leq n$.
- (3) Sei $k \in \mathbb{N}$. Wir betrachten das Wort

$$x' = uv^k w = \underbrace{a_1 \dots a_i}_{=u} \underbrace{a_{i+1} \dots a_j}_{=v} \underbrace{a_{i+1} \dots a_j}_{=v} \dots \underbrace{a_{i+1} \dots a_j}_{=v} \underbrace{a_{j+1} \dots a_m}_{=w} .$$

Der zu x' gehörende Lauf hat die Form

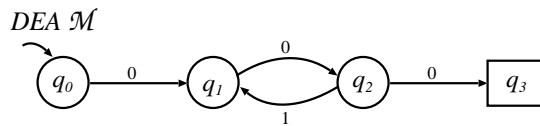
$$\begin{aligned} & q_0, \dots, q_i, q_{i+1}, \dots, q_j, q_{i+1}, \dots, q_j, \dots, q_{i+1}, \dots, q_j, q_{j+1}, \dots, q_m \\ & = q_0, \dots, q_i, (q_{i+1}, \dots, q_j)^k, q_{j+1}, \dots, q_m . \end{aligned}$$

Dabei benutzen wir die Tatsache, dass $q_i = q_j$. Wegen $q_m \in F$ folgt $x' \in \mathcal{L}(\mathcal{M}) = L$. □

Beispiel 6.2.22. Wir machen uns die Aussage von Satz 6.2.21 an einem einfachen Beispiel klar. Wir betrachten einen DEA mit vier Zuständen, der die Sprache

$$L = \{0(01)^n 00 : n \in \mathbb{N}\}$$

akzeptiert.



Jedes Wort $x \in \mathcal{L}(\mathcal{M})$ der Länge ≥ 4 hat die Form

$$x = 0(01)^{n+1}00, \text{ wobei } n \in \mathbb{N}.$$

Mit $u = 0$, $v = 01$ und $w = 00$ erhält man $uv^k w \in \mathcal{L}(\mathcal{M}) = L$ für alle $k \in \mathbb{N}$. ■

Nicht reguläre Sprachen

Das Pumping Lemma kann für den Nachweis verwendet werden, dass eine Sprache nicht regulär ist.¹¹

¹¹ Es gibt jedoch Sprachen, die zwar nicht regulär sind, aber dennoch die im Pumping Lemma angegebene Bedingung erfüllen. S. unten.

Als Beispiel betrachten wir

$$L = \{a^n b^n : n \in \mathbb{N}_{\geq 1}\}.$$

Zunächst machen wir uns intuitiv klar, weshalb L nicht regulär ist. Dies liegt im Wesentlichen daran, dass endliche Automaten keinen unbeschränkten Speicher haben, sondern lediglich die Zustände zur Speicherung von Daten verwenden können. Endliche Automaten können so konzipiert werden, dass sie bis 2, 3 oder bis zu einem anderen *festen* (von der Eingabe unabhängigen) Wert k »zählen« können; jedoch sind sie nicht in der Lage, von der Eingabe abhängige Werte zu speichern. Beim Einlesen der Eingabe kann der endliche Automat zwar prüfen, ob das Eingabewort von der Form $a^n b^m$ ist, er ist aber nicht in der Lage, die Information über die Anzahl der gelesenen a 's und b 's zu verwalten.

Wir weisen nun formal nach, dass L nicht regulär ist. Hierzu zeigen wir, dass die im Pumping Lemma angegebene Bedingung verletzt ist. Wir nehmen an, L wäre regulär und führen diese Annahme zu einem Widerspruch. Sei n die Zahl aus dem Pumping Lemma und sei $x = a^n b^n$. Weiter seien u, v und w Teilworte von x mit $x = uvw$ und den Eigenschaften (1), (2) und (3) aus dem Pumping Lemma.

Da $|uv| \leq n$ ist, besteht v nur aus a 's.

Etwa $v = a^l$. Dann ist $l = |v| \geq 1$. Somit ist

$$uv^2w = uvvw = a^{n+l}b^n \notin L.$$

Das ist ein Widerspruch zu Eigenschaft (3).

Ebenfalls mithilfe des Pumping Lemmas kann man zeigen, dass auch die Sprache L' aller Wörter $w \in \{a, b\}^*$, die ebenso viele a 's wie b 's enthalten, *nicht* regulär ist. Eine andere Methode, um den Nachweis zu erbringen, dass L' nicht regulär ist, ist diese. Offenbar ist

$$L = L' \cap \{a^n b^m : n, m \geq 0\}.$$

Die Sprache $L'' = \{a^n b^m : n, m \geq 0\}$ ist regulär. Dies kann man z. B. durch die Angabe eines DEAs \mathcal{M} für L'' belegen. Hierzu genügen zwei Zustände q_a und q_b . q_a ist der Startzustand, die Endzustandsmenge ist $\{q_a, q_b\}$. Der Startzustand q_a wird genau dann verlassen, wenn ein b gelesen wird. Im Zustand q_b können beliebig viele b 's gelesen werden. Sobald das Eingabezeichen jedoch ein a ist, hält der DEA verwerfend an.

Wäre nun L' regulär, dann wäre (gemäß der Aussage von Satz 6.2.16, Seite 246) auch die Sprache $L = L' \cap L''$ regulär. Dies ist nicht der Fall, wie wir oben gesehen haben.

Die Aussage des Pumping Lemmas stellt eine notwendige Bedingung für reguläre Sprachen dar. Es gibt jedoch nicht reguläre Sprachen, die das im Pumping Lemma angegebene Kriterium erfüllen. Beispielsweise ist die Sprache

$$L' = \{b^m : m \geq 0\} \cup \{a^n b^{m^2} : m, n \geq 1\}$$

nicht regulär, obwohl sie die im Pumping Lemma angegebene Eigenschaft hat (vgl. Übungsaufgabe 6.4).

6.3 Reguläre Ausdrücke

Satz 6.2.16 (Seite 246) kann dahingehend verschärft werden, dass die Klasse der regulären Sprachen über Σ die kleinste Klasse ist, die unter den Operationen Vereinigung, Konkatenation und Kleeneabschluss abgeschlossen ist und welche die Sprachen \emptyset , $\{\varepsilon\}$ und $\{a\}$, $a \in \Sigma$ enthält.¹² Um diese Aussage zu belegen, betrachten wir einen weiteren Formalismus, der sehr intuitive Schreibweisen für reguläre Sprachen ermöglicht.

Definition 6.3.1 [Syntax regulärer Ausdrücke]

Sei Σ ein Alphabet. Die Menge der regulären Ausdrücke über Σ ist durch folgende induktive Definition gegeben.

1. \emptyset und ε sind reguläre Ausdrücke.
2. Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck.
3. Mit α und β sind auch $(\alpha\beta)$, $(\alpha + \beta)$ und (α^*) reguläre Ausdrücke.
4. Nichts sonst ist ein regulärer Ausdruck.

Die Klammern werden oftmals weggelassen, wobei der Verknüpfungsoperator $+$ die schwächste Priorität hat und der Sternoperator am stärksten bindet. Zum Beispiel steht $a\varepsilon + bc^*$ für $((a\varepsilon) + (bc^*))$. Anstelle von $\alpha + \beta$ wird häufig auch $\alpha|\beta$ geschrieben.

Anstelle der induktiven Definition kann man auch auf folgende kontextfreie Grammatik zurückgreifen, die die regulären Ausdrücke generiert:

$$\alpha \rightarrow \emptyset \mid \varepsilon \mid a \mid \alpha\alpha \mid \alpha + \alpha \mid \alpha^*$$

Reguläre Ausdrücke stehen für Sprachen. Intuitiv sind ε und a Kurzschreibweisen für die jeweils einelementigen Sprachen $\{\varepsilon\}$ und $\{a\}$. Das Symbol $+$ steht für die Vereinigung, die Schreibweise $\alpha\beta$ deutet Konkatenation an. Der Sternoperator repräsentiert den Kleeneabschluss.

Definition 6.3.2 [Semantik regulärer Ausdrücke]

Die zu einem regulären Ausdruck α gehörende Sprache $\mathcal{L}(\alpha)$ ist wie folgt definiert:

$$\begin{array}{ll} \mathcal{L}(\emptyset) &= \emptyset & \mathcal{L}(\varepsilon) &= \{\varepsilon\} \\ \mathcal{L}(a) &= \{a\} & \mathcal{L}(\alpha\beta) &= \mathcal{L}(\alpha) \circ \mathcal{L}(\beta) \\ \mathcal{L}(\alpha + \beta) &= \mathcal{L}(\alpha) \cup \mathcal{L}(\beta) & \mathcal{L}(\alpha^*) &= \mathcal{L}(\alpha)^* \end{array}$$

Zwei reguläre Ausdrücke α_1, α_2 heißen *äquivalent* (i. Z. $\alpha_1 \equiv \alpha_2$), wenn $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$. Wir nennen α nicht leer, wenn $\alpha \neq \emptyset$, also wenn $\mathcal{L}(\alpha) \neq \emptyset$.

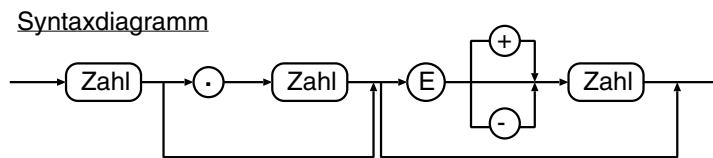
¹² Diese Beobachtung geht auf Kleene zurück und wird daher häufig als *Satz von Kleene* bezeichnet.

Beispielsweise stellt der reguläre Ausdruck

$$\alpha = \text{Ziffer Ziffer}^*$$

die Menge aller nicht leeren Ziffernfolgen dar. Dabei kann man entweder *Ziffer* als Terminalzeichen auffassen (d.h. man legt das Alphabet $\{\text{Ziffer}\}$ zu Grunde) oder fasst $\text{Ziffer} = 0|1|2|\dots|9$ als regulären Ausdruck über $\{0, 1, \dots, 9\}$ auf.

Als komplexeres Beispiel betrachten wir nichtnegative Gleitpunktzahlen, die durch das nachstehende Syntaxdiagramm spezifiziert sind.



Zum Beispiel sind »3.87E – 7«, »3.475« oder »014E74« zulässige Gleitpunktdarstellungen. Die durch das Syntaxdiagramm festgelegte Sprache wird durch den folgenden regulären Ausdruck beschrieben:

$$\text{Zahl } (\varepsilon | \cdot \text{Zahl}) (\varepsilon | \text{E}(+ | - | \varepsilon) \text{Zahl})$$

Dabei ist *Zahl* der (oben betrachtete) reguläre Ausdruck

$$\text{Ziffer Ziffer}^*.$$

Hier haben wir das Symbol »|« anstelle von »+« für den Vereinigungsoperator verwendet, um Verwechslungen mit dem Terminalzeichen »+« zu verhindern.

Konstruktion eines NEAs zu gegebenem regulären Ausdruck

Wir geben zwei Verfahren an, wie man zu gegebenem regulären Ausdruck α einen NEA konstruieren kann, der genau die Sprache $\mathcal{L}(\alpha)$ akzeptiert. Diese Verfahren liefern den Beweis für folgendes Lemma.

Lemma 6.3.3

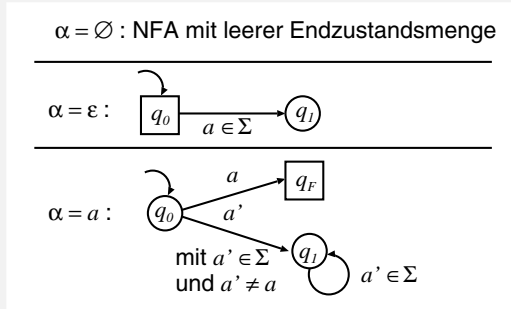
Zu jedem regulären Ausdruck α gibt es einen NEA \mathcal{M} mit $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$.

1. Verfahren: Für $\alpha = \emptyset, \varepsilon$ oder $a \in \Sigma$ geben wir explizit einen NEA \mathcal{M} mit $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\alpha)$ an (vgl. Abbildung 6.3.4).

Ist α von der Form

$$\alpha_1\alpha_2 \text{ oder } \alpha_1 + \alpha_2 \text{ oder } \alpha_0^*,$$

Abbildung 6.3.4 Beispiel für erstes Verfahren



dann konstruieren wir zuerst (rekursiv) NEAs für die Teilausdrücke α_i und wenden dann den entsprechenden Operator für NEAs an. Siehe Abschnitt 6.2.1 (Seite 236 ff).

2. Verfahren: Eine weitere Konstruktionsmöglichkeit verwendet ε -erweiterte NEAs (siehe Seite 241 ff). Den Spezialfall $\alpha \equiv \emptyset$ können wir ausklammern, da die akzeptierte Sprache eines NEAs ohne Endzustände (d.h. mit leerer Endzustandsmenge) offenbar $\emptyset = \mathcal{L}(\emptyset)$ ist.

Im Folgenden setzen wir $\alpha \neq \emptyset$ voraus. Beginnend mit einem Digraphen, bestehend aus zwei Zuständen, die über eine mit α beschriftete Kante miteinander verbunden sind, fügen wir sukzessive neue Zustände ein. Die Übergänge sind zunächst mit regulären Ausdrücken beschriftet (vgl. Abbildung 6.3.5). Diese werden sukzessive durch echte Teilausdrücke ersetzt, bis alle Kanten mit ε oder einem Terminalzeichen $a \in \Sigma$ beschriftet sind. Der resultierende Graph ist ein ε -erweiterter NEA, der genau die Sprache $\mathcal{L}(\alpha)$ akzeptiert.

Konstruktion eines regulären Ausdrucks aus einem DEA

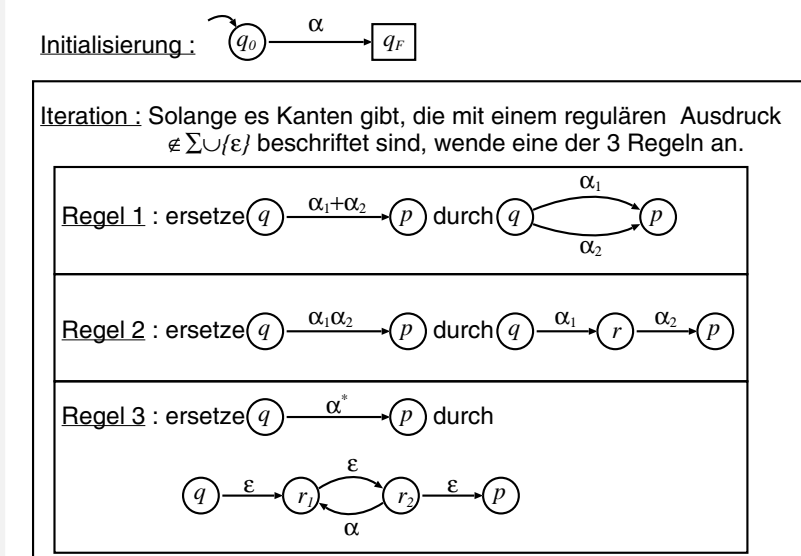
Die bisherigen Ergebnisse belegen noch nicht, dass jede reguläre Sprache durch einen regulären Ausdruck beschrieben werden kann. Wir zeigen nun, dass zu jedem DEA \mathcal{M} ein regulärer Ausdruck α konstruiert werden kann, sodass $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$. Wir werden dazu die Methode des *dynamischen Programmierens* zur Berechnung regulärer Ausdrücke für die Sprachen $\mathcal{L}_{i,j}$ aller Wörter x , sodass $\delta(q_i, x) = q_j$, anwenden.

Lemma 6.3.6

Zu jedem DEA \mathcal{M} gibt es einen regulären Ausdruck α mit $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$.

Beweis: Sei $\mathcal{M} = (Q, \Sigma, \delta, q_1, F)$ ein DEA. Sei

$$Q = \{q_1, \dots, q_n\},$$

Abbildung 6.3.5 Konstruktion eines NEAs mit ε -Übergängen

wobei q_1, \dots, q_n paarweise verschieden sind. Für $x = a_1 a_2 \dots a_m \in \Sigma^*$ sei

$$run_i(x)$$

der Lauf für x in dem DEA $\mathcal{M}_i = (Q, \Sigma, \delta, q_i, F)$. \mathcal{M}_i stimmt also mit \mathcal{M} bis eventuell auf den Anfangszustand überein. Es gilt: Ist $\delta(q_i, x) \neq \perp$, so ist $run_i(x)$ die Zustandsfolge p_0, p_1, \dots, p_m , wobei

- ▶ $p_0 = q_i$ und
- ▶ $p_{l+1} = \delta(p_l, a_{l+1})$, $l = 0, 1, \dots, m-1$.

Weiter sei $L_{i,j}^k$ die Menge aller Wörter $x \in \Sigma^*$, sodass

- ▶ $\delta(q_i, x) = q_j$ und
- ▶ $run_i(x) = p_0, p_1, \dots, p_m$, wobei $p_1, \dots, p_{m-1} \in \{q_1, q_2, \dots, q_k\}$.

In Worten: $L_{i,j}^k$ umfasst genau diejenigen Wörter x , für die der DEA \mathcal{M}_i im Zustand $p_m = q_j$ endet und – abgesehen vom Anfangszustand q_i und dem letzten Zustand $p_m = q_j$ – nur Zustände aus der Menge $\{q_1, \dots, q_k\}$ durchläuft.

Es gilt für $1 \leq i, j \leq n$:

$$L_{i,j}^0 = \{a \in \Sigma : \delta(q_i, a) = q_j\}, \quad L_{i,i}^0 = \{\varepsilon\} \cup \{a \in \Sigma : \delta(q_i, a) = q_i\}.$$

Für $k = 0, 1, \dots, n - 1$ erhalten wir:

$$L_{i,j}^{k+1} = L_{i,j}^k \cup L_{i,k+1}^k \circ \left(L_{k+1,k+1}^k \right)^* \circ L_{k+1,j}^k.$$

Anhand dieser Rekursionsformeln für die Mengen $L_{i,j}^k$ definieren wir induktiv reguläre Ausdrücke $\alpha_{i,j}^k$ mit

$$L_{i,j}^k = \mathcal{L} \left(\alpha_{i,j}^k \right).$$

Offenbar gibt es reguläre Ausdrücke $\alpha_{i,j}^0$ mit $\mathcal{L}(\alpha_{i,j}^0) = L_{i,j}^0, i, j = 1, \dots, n$.

► Ist $L_{i,j}^0 = \{a_{l_1}, a_{l_2}, \dots, a_{l_r}\} \neq \emptyset$ und $i \neq j$, so können wir reguläre Ausdrücke der Form

$$\alpha_{i,j}^0 = a_{l_1} + a_{l_2} + \dots + a_{l_r}$$

verwenden.

► Ist $i = j$ und $L_{i,i}^0$ wie oben, so setzen wir $\alpha_{i,i}^0 = \varepsilon + a_{l_1} + a_{l_2} + \dots + a_{l_r}$.

► Ist $L_{i,j}^0 = \emptyset$ und $i \neq j$, so setzen wir $\alpha_{i,j}^0 = \emptyset$.

Für $k = 0, 1, \dots, n - 1$ setzen wir

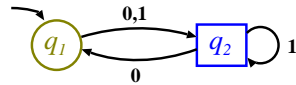
$$\alpha_{i,j}^{k+1} = \alpha_{i,j}^k + \alpha_{i,k+1}^k \left(\alpha_{k+1,k+1}^k \right)^* \alpha_{k+1,j}^k.$$

Sei $F = \{q_{l_1}, \dots, q_{l_r}\}$, wobei $1 \leq l_1 < \dots < l_r \leq n$. Es gilt offenbar:

$$\mathcal{L}(\mathcal{M}) = \bigcup_{q_j \in F} L_{1,j}^n = L_{1,l_1}^n \cup \dots \cup L_{1,l_r}^n.$$

Somit ist $\alpha = \alpha_{1,l_1}^n + \dots + \alpha_{1,l_r}^n$ ein regulärer Ausdruck mit $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$. □

Beispiel 6.3.7. Wir veranschaulichen die im Beweis von Lemma 6.3.6 (Seite 254) angegebene Methode exemplarisch für den folgenden Automaten \mathcal{M} .



Man erhält folgende regulären Ausdrücke:

$\alpha_{1,1}^0 = \varepsilon$	$\alpha_{2,2}^0 = 1 + \varepsilon$
$\alpha_{1,2}^0 = 0 + 1$	$\alpha_{2,1}^0 = 0$
$\alpha_{1,1}^1 \equiv \varepsilon$	$\alpha_{2,2}^1 = 1 + \varepsilon + 00 + 01$
$\alpha_{1,2}^1 \equiv 0 + 1$	$\alpha_{2,1}^1 \equiv 0$

und

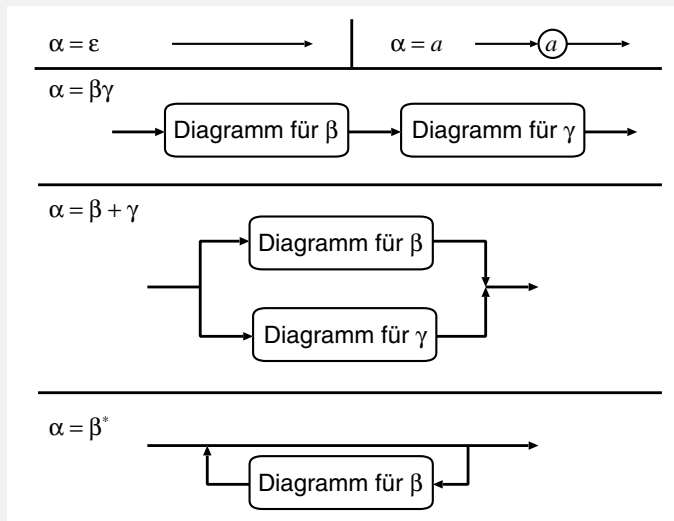
$$\begin{aligned}\alpha &= \alpha_{1,2}^2 = \alpha_{1,2}^1 + \alpha_{1,2}^1 (\alpha_{2,2}^1)^* \alpha_{2,2}^1 \\ &\equiv 0 + 1 + (0 + 1)(\varepsilon + 1 + 00 + 01)^*(\varepsilon + 1 + 00 + 01) \quad \blacksquare\end{aligned}$$

Syntaxdiagramme

Syntaxdiagramme sind eine sehr intuitive Darstellungsform der Regeln, die zur Bildung von Grundsymbolen (z. B. vom Benutzer definierte Variablennamen, Ziffern etc.) zugelassen sind. Wir haben Syntaxdiagramme bereits an mehreren Stellen verwendet und gehen davon aus, dass die Semantik den Leserinnen und Lesern intuitiv klar ist.

Syntaxdiagramme und reguläre Ausdrücke: Jeder reguläre Ausdruck $\alpha \neq \emptyset$ lässt sich grafisch durch ein Syntaxdiagramm darstellen. Das folgende Konstruktionsverfahren benutzt *strukturelle Induktion* über den syntaktischen Aufbau des Ausdrucks α (vgl. Abbildung 6.3.8).

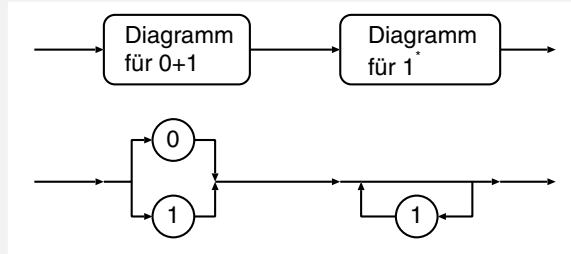
Abbildung 6.3.8 Transformationen regulärer Ausdrücke in Syntaxdiagramme



Als Beispiel betrachten wir den regulären Ausdruck $\alpha = (0 + 1)1^*$, für den das in Abbildung 6.3.9 skizzierte Syntaxdiagramm erstellt wird.

Syntaxdiagramme und endliche Automaten: Um ein Verfahren anzugeben, das ein Syntaxdiagramm in einen endlichen Automaten überführt, verwenden wir eine graphentheoretische Sicht. Wir fassen ein Syntaxdiagramm als *Digraphen* auf, dessen Knoten mit einem Symbol $a \in \Sigma$ markiert sind. Weiter arbeiten wir mit zwei zusätzlichen (im

Abbildung 6.3.9 Syntaxdiagramm für $(0 + 1)^*$



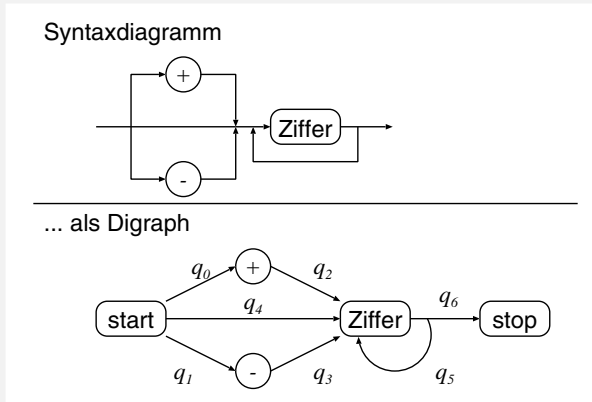
Syntaxdiagramm nicht explizit dargestellten) Knoten, die für den Eingang bzw. Ausgang stehen und nicht markiert sind. Diese nennen wir *start* und *stop*.

- ▶ Die Kantenmenge bezeichnen wir mit E , die Knotenmenge mit N .
- ▶ Wir schreiben $\ell(v)$ für die Markierung eines Knotens $v \in N \setminus \{start, stop\}$.

Für jeden Knoten u des Syntaxdiagramms sei $Post(u)$ die Menge aller Knoten w , die von u über eine Kante erreichbar sind (also die Menge der direkten Nachfolger von u), d.h. es gilt $v \in Post(u)$ genau dann, wenn $\langle u, v \rangle \in E$. Entsprechend ist $Pre(v) = \{u \in N : \langle u, v \rangle \in E\}$.

Der zu einem Syntaxdiagramm gehörende NEA: Wir ordnen dem Syntaxdiagramm zunächst einen NEA zu, der sich im Wesentlichen daraus ergibt, dass wir den zum Syntaxdiagramm *dualen Graphen* bilden. Das heißt wir fassen die Kanten des Syntaxdiagramms als Zustände und die Knoten (ausgenommen die Spezialknoten *start* und *stop*) des Syntaxdiagramms als Kanten auf (vgl. Abbildung 6.3.10).

Abbildung 6.3.10



- Die Anfangszustände des NEAs sind die von dem Knoten *start* ausgehenden Kanten $\langle start, v \rangle$.
- Die Endzustandsmenge besteht aus allen Kanten $\langle v, stop \rangle$, die in den Zustand *stop* führen.

Der zum Syntaxdiagramm (N, E, ℓ) gehörende NEA ist

$$\mathcal{M} = (E, \Sigma, \delta, E_0, F)$$

mit

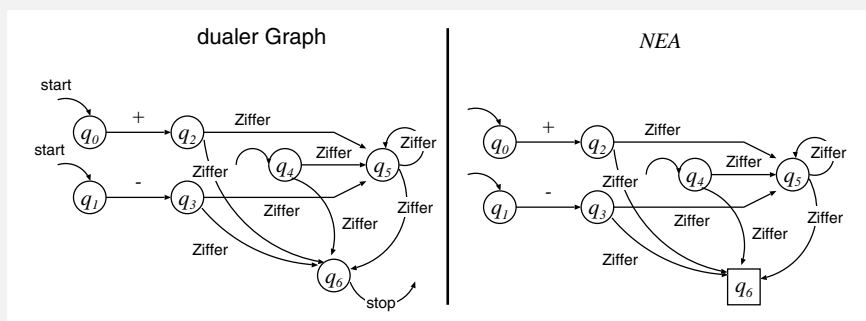
$$E_0 = \{\langle start, v \rangle : v \in Post(start)\},$$

$$F = \{\langle v, stop \rangle : v \in Pre(stop)\},$$

wobei die Übergangsfunktion δ wie folgt definiert ist: Für $a \in \Sigma$ ist

$$\delta(\langle u, v \rangle, a) = \{\langle v, w \rangle : w \in Post(v), \ell(v) = a\}.$$

Abbildung 6.3.11



Der so erhaltene NEA kann nun mit der Potenzmengenkonstruktion in einen DEA überführt werden (siehe Lemma 6.1.18, Seite 231).

Zusammenfassung

Unsere bisherigen Ergebnisse belegen, dass die Klasse der regulären Sprachen durch die Formalismen reguläre Grammatiken, DEAs, NEAs, reguläre Ausdrücke und Syntaxdiagramme eindeutig charakterisiert ist. In diesem Sinn sind die genannten Formalismen gleichwertig. Darüber hinaus haben wir *konstruktive* Methoden angegeben, wie man die Formalismen ineinander überführen kann.

$$\text{DEA} \implies \text{regulärer Ausdruck} \implies \text{NEA} \implies \text{DEA}$$

Weiter haben wir Transformationen

Syntaxdiagramm \implies NEA oder DEA \implies reguläre Grammatik \implies NEA

regulärer Ausdruck \implies Syntaxdiagramm

kennen gelernt.

6.4 Minimierung von endlichen Automaten

Die Transformation der Formalismen ist teilweise sehr aufwändig. Beispielsweise entstehen mit der auf dem dynamischen Programmieren beruhenden Methode zur Erstellung eines regulären Ausdrucks für einen DEA im Allgemeinen sehr lange reguläre Ausdrücke. Ebenso führt die Potenzmengenkonstruktion eines DEAs aus einem NEA oftmals zu einem sehr großen Zustandsraum. Es gibt zwar reguläre Sprachen, für die jeder DEA mindestens exponentiell größer ist als ein kleinster NEA (Satz 6.1.23, Seite 234), jedoch entstehen bei den Transformationen oftmals *unnötig große* DEAs. In diesem Abschnitt stellen wir ein Verfahren vor, das einen gegebenen DEA \mathcal{M} *minimiert*, d.h. \mathcal{M} durch einen äquivalenten DEA ersetzt, dessen Zustandsraum minimal unter allen äquivalenten DEAs ist.

6.4.1 Der Satz von Myhill & Nerode

Der Satz von Myhill & Nerode stellt eine weitere Charakterisierung regulärer Sprachen dar und liefert die Basis für den Minimierungsalgorithmus.

Definition 6.4.1 [Die Äquivalenzrelation \sim_L]

Sei $L \subseteq \Sigma^*$ eine Sprache. Die Relation \sim_L bezeichnet folgende Äquivalenzrelation auf Σ^* . Für alle $x, y \in \Sigma^*$ gilt

$$x \sim_L y \text{ gdw für alle } z \in \Sigma^* \text{ gilt: } xz \in L \iff yz \in L.$$

Wir schreiben $[x]_L$ oder kurz $[x]$ für die Äquivalenzklasse von x bzgl. \sim_L . Es gilt also

$$[x]_L = \{y \in \Sigma^* : x \sim_L y\}.$$

Entsprechend schreiben wir Σ^*/L für den Quotientenraum Σ^*/\sim_L .

Beispiel 6.4.2. Wir betrachten die reguläre Sprache $L = \mathcal{L}(0^*1^*)$. Zum Beispiel gilt

$$0 \sim_L 00 \sim_L 000 \sim_L \dots,$$

da $0^n z \in L$ genau dann, wenn $z \in \mathcal{L}(0^*1^*)$. Sämtliche Wörter $x \in \mathcal{L}(0^*)$ liegen also in derselben Äquivalenzklasse (diese ist $\mathcal{L}(0^*)$). Entsprechend gilt

$$1 \sim_L 01 \sim_L 001 \sim_L \dots,$$

da z. B. $0^n 1z \in L$ genau dann, wenn $z \in \mathcal{L}(1^*)$. Entsprechendes gilt für die Wörter der Form $0^n 1^m$, $m \geq 1$, $n \geq 0$.

Alle anderen Wörter x (in denen eine Eins vor einer Null steht) sind paarweise äquivalent, da $xz \notin L$ für alle Wörter $z \in \{0, 1\}^*$. Beispielsweise gilt

$$10 \sim_L 1010 \sim_L 1000 \sim_L 0110.$$

Diese Überlegungen belegen, dass es genau drei Äquivalenzklassen bzgl. \sim_L gibt. Der Quotientenraum bzgl. \sim_L ist

$$\Sigma^*/L = \left\{ \mathcal{L}(0^*), \mathcal{L}(0^*11^*), L \setminus (\mathcal{L}(0^*) \cup \mathcal{L}(0^*11^*)) \right\} = \{[0], [1], [10]\},$$

es gibt also genau drei Äquivalenzklassen bzgl. \sim_L . ■

Der *Index* einer Äquivalenzrelation bezeichnet die Anzahl an Äquivalenzklassen. Im Folgenden sprechen wir kurz von dem *Index von L* , um den Index von \sim_L zu bezeichnen. Der Index von L ist also genau dann endlich, wenn der Quotientenraum Σ^*/L endlich ist.

Satz 6.4.3 [Satz von Myhill & Nerode]

Sei L eine Sprache.

L ist genau dann regulär, wenn der Index von L endlich ist.

Bevor wir den Satz von Myhill & Nerode beweisen, überzeugen wir uns von der Aussage an zwei Beispielen.

- Für die reguläre Sprache $L = \mathcal{L}(0^*1^*)$ besteht Σ^*/L aus drei Elementen (siehe Beispiel 6.4.2).
- Die Sprache

$$K = \{0^n 1^n : n \geq 0\}$$

ist nicht regulär (siehe Seite 250). In der Tat sind die Wörter $0^i 1^j$ mit $0 \leq j \leq i$ paarweise nicht äquivalent bzgl. \sim_K . Zum Beispiel ist

$$0^i 1^j 1^{i-j} = 0^i 1^i \in K, \text{ aber } 0^i 1^{j+1} 1^{i-j} = 0^i 1^{i+1} \notin K.$$

(Dabei nehmen wir $0 \leq j < i$ an.)

Somit ist der Quotientenraum $\{0, 1\}^*/K$ unendlich.

Der Beweis von Satz 6.4.3 benutzt eine Hilfsaussage, die wir in den folgenden beiden Lemmas formulieren.

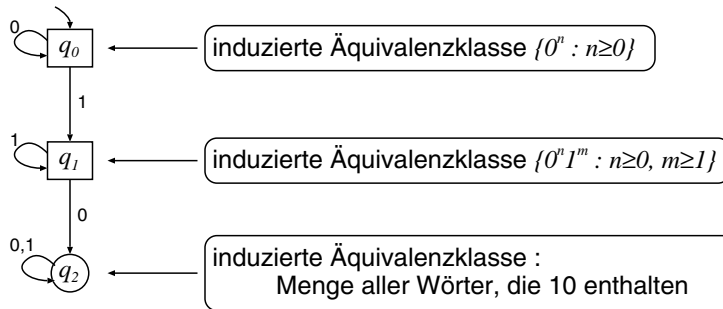
Definition 6.4.4 [Die Äquivalenzrelation $\sim_{\mathcal{M}}$]

Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Die Äquivalenzrelation $\sim_{\mathcal{M}}$ ist durch

$$x \sim_{\mathcal{M}} y \text{ gdw } \delta(q_0, x) = \delta(q_0, y)$$

definiert. Wir schreiben Σ^*/\mathcal{M} , um den Quotientenraum $\Sigma^*/\sim_{\mathcal{M}}$ zu bezeichnen, und $[x]_{\mathcal{M}}$ (statt $[x]_{\sim_{\mathcal{M}}}$) für die Äquivalenzklasse von x bzgl. $\sim_{\mathcal{M}}$.

Beispiel 6.4.5. Als Beispiel betrachten wir einen DEA \mathcal{M} , der die Sprache $\mathcal{L}(0^*1^*)$ akzeptiert.



Lemma 6.4.6

Für jeden DEA \mathcal{M} ist der Index von $\sim_{\mathcal{M}}$ endlich.

Beweis: Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Für jeden Zustand q sind die Wörter aus der Sprache

$$K_q = \{x \in \Sigma^* : \delta(q_0, x) = q\}$$

paarweise äquivalent bzgl. $\sim_{\mathcal{M}}$. Andererseits gilt $x \not\sim_{\mathcal{M}} y$ für $q \neq p, x \in K_q, y \in K_p$. Somit gilt:

$$[x]_{\mathcal{M}} = K_q, \text{ falls } \delta(q_0, x) = q.$$

Der Quotientenraum bzgl. $\sim_{\mathcal{M}}$ ist daher

$$\Sigma^*/\mathcal{M} = \{K_q : q \in Q\} \setminus \{\emptyset\}.$$

Also ist

$$|\Sigma^*/\mathcal{M}| \leq |Q| < \infty.$$

**Lemma 6.4.7**

Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA und $L = \mathcal{L}(\mathcal{M})$, dann gilt:

(a) $\sim_{\mathcal{M}}$ ist eine Verfeinerung von \sim_L , d.h. für alle $x, y \in \Sigma^*$ gilt:

Aus $x \sim_{\mathcal{M}} y$ folgt $x \sim_L y$.

(b) $|Q| \geq |\Sigma^*/\mathcal{M}| \geq |\Sigma^*/L|$.

Beweis: Es gelte $x \sim_{\mathcal{M}} y$. Weiter sei $z \in \Sigma^*$, dann gilt:

$$\delta(q_0, xz) = \delta(\delta(q_0, x), z) = \delta(\delta(q_0, y), z) = \delta(q_0, yz).$$

Hieraus folgt:

$$\begin{aligned} & xz \in L = \mathcal{L}(\mathcal{M}) \\ \text{gdw} & \quad \delta(q_0, xz) \in F \\ \text{gdw} & \quad \delta(q_0, yz) \in F \\ \text{gdw} & \quad yz \in \mathcal{L}(\mathcal{M}) = L. \end{aligned}$$

Somit gilt $x \sim_L y$. Die Relation $\sim_{\mathcal{M}}$ ist also eine Verfeinerung von \sim_L . Daher ist jede Äquivalenzklasse bzgl. $\sim_{\mathcal{M}}$ in einer Äquivalenzklasse bzgl. \sim_L enthalten.

Sei Q' die Menge aller vom Anfangszustand q_0 erreichbaren Zustände. Für $q \in Q'$ sei L_q diejenige Äquivalenzklasse bzgl. \sim_L , welche die durch q induzierte Äquivalenzklasse

$$K_q = \{x \in \Sigma^* : \delta(q_0, x) = q\}$$

bzgl. $\sim_{\mathcal{M}}$ enthält. Also:

$$K_q \subseteq L_q \in \Sigma^*/L.$$

(Man beachte, dass $K_q \neq \emptyset$ für $q \in Q'$. Weiter ist $L_q = L_p$ für $q \neq p$ möglich.) Dann gilt:

$$\Sigma^*/L = \{L_q : q \in Q'\}.$$

Also ist

$$|\Sigma^*/L| \leq |Q'| = |\Sigma^*/\mathcal{M}| < \infty.$$

(Siehe Lemma 6.4.6, Seite 262.)



Beweis von Satz 6.4.3 (Seite 261): Ist L regulär und $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA mit

$$\mathcal{L}(\mathcal{M}) = L,$$

dann gilt

$$|\Sigma^*/L| \leq |\Sigma^*/\mathcal{M}| \leq |Q| < \infty$$

(siehe Lemma 6.4.7, Seite 263).

Wir nehmen nun an, dass L eine Sprache mit endlichem Index ist. Wir zeigen, dass L regulär ist. Zunächst stellen wir fest, dass für alle $x, y \in \Sigma^*$ und $a \in \Sigma$ gilt:

(*) Aus $x \sim_L y$ folgt $xa \sim_L ya$.

Insbesondere ist $[xa]_L = [ya]_L$.

Wir definieren einen DEA

$$\mathcal{M}_L = (Q_L, \Sigma, \delta_L, q_{0,L}, F_L),$$

dessen Zustände die Äquivalenzklassen bzgl. \sim_L sind:

$$Q_L = \Sigma^*/L, \quad q_{0,L} = [\varepsilon]_L, \quad F_L = \{[x]_L : x \in L\}, \quad \delta_L([x]_L, a) = [xa]_L.$$

Die Aussage (*) stellt sicher, dass die Übergangsfunktion wohl definiert ist. Durch Induktion nach der Länge von x kann man zeigen, dass

$$\delta_L([\varepsilon]_L, x) = [x]_L$$

für alle $x \in \Sigma^*$. Weiter gilt

$$x \in L \quad \text{gdw} \quad [x]_L \in F_L.^{13}$$

Hieraus folgt:

$$\begin{aligned} & x \in \mathcal{L}(\mathcal{M}_L) \\ \text{gdw} & \quad \delta_L([\varepsilon]_L, x) \in F_L \\ \text{gdw} & \quad [x]_L \in F_L \\ \text{gdw} & \quad x \in L. \end{aligned}$$

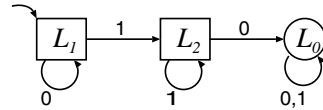
Also ist $\mathcal{L}(\mathcal{M}_L) = L$ und L regulär. □

Definition 6.4.8 [Minimalautomat]

Sei L eine reguläre Sprache. Der im Beweis von Satz 6.4.3 konstruierte DEA für L wird *Minimalautomat* von L genannt und mit \mathcal{M}_L bezeichnet.

¹³ Man beachte, dass » \Leftarrow « an der speziellen Form von \sim_L liegt. Ist $[x]_L \in F_L$, dann gibt es ein $x' \in L$, sodass $[x']_L = [x]_L$; also $x \sim_L x'$. Wir betrachten das Wort $z = \varepsilon$ und erhalten: $x = xz \in L$, da $x'z = x' \in L$.

Beispiel 6.4.9. Als Beispiel betrachten wir wieder die Sprache $L = \mathcal{L}(0^*1^*)$. Sei $L_1 = \mathcal{L}(0^*)$, $L_2 = \mathcal{L}(0^*11^*)$ und $L_0 = \{0, 1\}^* \setminus (L_1 \cup L_2)$. Der Minimalautomat für L ist:



Teil (b) des folgenden Satzes 6.4.13 zeigt, dass die Anzahl der Zustände im Minimalautomaten minimal unter allen DEAs für die betreffende Sprache ist (was die Bezeichnung »Minimalautomat« rechtfertigt). In Teil (a) zeigen wir die Eindeutigkeit des Minimalautomaten als DEA, für den die induzierte Äquivalenzrelation $\sim_{\mathcal{M}}$ mit \sim_L übereinstimmt. Tatsächlich haben wir nur Eindeutigkeit »bis auf Isomorphie«, d.h. bis auf eventuelles Umbenennen der Zustandsnamen. Auch wenn der Begriff der Isomorphie intuitiv klar ist, geben wir die präzise Definition von Isomorphie an.

Definition 6.4.10 [Isomorphie]

Zwei DEAs $\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, $\mathcal{M}_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ heißen *isomorph*, wenn es eine bijektive Abbildung $f : Q_1 \rightarrow Q_2$ gibt, sodass

- ▶ $q_2 = f(q_1)$,
- ▶ $F_2 = f(F_1)$ und
- ▶ $\delta_2(f(q), a) = f(\delta_1(q, a))$

für alle $q \in Q_1$ und $a \in \Sigma$. Die Abbildung f wird *Isomorphismus* genannt.

\mathcal{M}_1 und \mathcal{M}_2 sind also genau dann isomorph, wenn \mathcal{M}_1 und \mathcal{M}_2 bis auf die Namen der erreichbaren Zustände übereinstimmen. Intuitiv identifizieren wir isomorphe DEAs, da die Zustandsnamen völlig irrelevant für die akzeptierte Sprache sind. Offensichtlich gilt folgendes Lemma:

Lemma 6.4.11

Sind \mathcal{M}_1 und \mathcal{M}_2 isomorphe DEAs, so gilt $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$.

Bemerkung 6.4.12. Mit unserer Definition des Minimalautomaten erhalten wir eine totale Übergangsfunktion. Unter allen DEAs mit totaler Übergangsfunktion ist \mathcal{M}_L minimal, jedoch ist eine weitere Zustandsreduktion möglich, wenn man partielle Übergangsfunktionen zulässt.

Sei $L_0 = \{x \in \Sigma^* : xz \notin L \text{ für alle } z \in \Sigma^*\}$. Falls $L_0 \neq \emptyset$, so können L_0 und alle zugehörigen Kanten entfernt werden. Der resultierende Automat ist nun minimal unter allen DEAs \mathcal{M} mit $\mathcal{L}(\mathcal{M}) = L$.

Beispielsweise kann L_0 im Beispiel 6.4.9 aus dem Minimalautomaten \mathcal{M}_L für die Sprache $L = \mathcal{L}(0^*1^*)$ eliminiert werden. ■

Der folgende Satz belegt die versprochene Tatsache, dass der Minimalautomat tatsächlich minimale Größe hat.

Satz 6.4.13

Sei L eine reguläre Sprache.

(a) Der Minimalautomat \mathcal{M}_L für L ist der bis auf Isomorphie eindeutig bestimmte DEA mit folgenden drei Eigenschaften:

- ▶ $\mathcal{L}(\mathcal{M}_L) = L$
- ▶ $\sim_{\mathcal{M}_L} = \sim_L$
- ▶ Jeder Zustand ist vom Anfangszustand erreichbar.

(b) Ist $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DEA mit $\mathcal{L}(\mathcal{M}) = L$, dann ist $|Q| \geq |\Sigma^*/L|$.

Beweis: Teil (b) folgt sofort aus Lemma 6.4.7 (Seite 263). Wir zeigen Aussage (a). Zunächst zeigen wir, dass die Relationen $\sim_{\mathcal{M}_L}$ und \sim_L übereinstimmen.

Wegen Teil (a) von Lemma 6.4.7 (Seite 263) genügt es zu zeigen, dass \sim_L eine Verfeinerung von $\sim_{\mathcal{M}_L}$ ist. Im Beweis von Satz 6.4.3 (Seite 264) haben wir erwähnt, dass

$$\delta_L([\varepsilon]_L, w) = [w]_L \quad \text{für alle Wörter } w \in \Sigma^* .$$

Seien $x, y \in \Sigma^*$ und $x \sim_L y$. Es folgt

$$\delta_L([\varepsilon]_L, x) = [x]_L = [y]_L = \delta_L([\varepsilon]_L, y) .$$

Also ist $x \sim_{\mathcal{M}_L} y$.

Es folgt, dass \sim_L und $\sim_{\mathcal{M}_L}$ übereinstimmen. Weiter ist klar, dass jeder Zustand in \mathcal{M}_L vom Anfangszustand erreichbar ist. Dies folgt aus der Beobachtung:

$$\text{Ist } x \in \Sigma^*, \text{ so ist } [x]_L = \delta_L([\varepsilon]_L, x).$$

Die Aussage $\mathcal{L}(\mathcal{M}_L) = L$ wurde im Beweis von Satz 6.4.3 nachgewiesen.

Damit erfüllt \mathcal{M}_L die drei genannten Eigenschaften. Wir zeigen nun, dass jeder weitere DEA mit den genannten drei Eigenschaften zu \mathcal{M}_L isomorph ist.

Sei nun $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein weiterer DEA mit folgenden drei Eigenschaften:

- ▶ $\mathcal{L}(\mathcal{M}) = L$,
- ▶ $\sim_{\mathcal{M}} = \sim_L$,
- ▶ Alle Zustände $q \in Q$ sind vom Anfangszustand q_0 erreichbar.

Für jeden Zustand $q \in Q$ definieren wir die Sprache

$$K_q = \{x \in \Sigma^* : \delta(q_0, x) = q\}.$$

Dann ist K_q eine Äquivalenzklasse bzgl. $\sim_{\mathcal{M}} = \sim_L$.¹⁴ Also ist $K_q \in \Sigma^*/L$ ein Zustand im Minimalautomaten \mathcal{M}_L . Es ist leicht zu sehen, dass die Abbildung

$$Q \rightarrow \Sigma^*/L, q \mapsto K_q,$$

ein Isomorphismus ist. □

In Abschnitt 6.1.2 (Seite 234) haben wir erwähnt, dass die Sprachen

$$L_n = \{w \in \{0, 1\}^* : \text{das } n\text{-letzte Zeichen von } w \text{ ist eine } \text{«}1\text{«}\}$$

durch NEAs mit $n + 1$ Zuständen akzeptiert werden können, während jeder DEA für L_n mindestens exponentiell viele Zustände hat. Den Beweis der letzten Aussage können wir nun recht einfach mit dem Satz von Myhill & Nerode (Seite 261) und Satz 6.4.13 (Seite 266) führen.

Lemma 6.4.14

Jeder DEA für L_n hat mindestens 2^n Zustände.

Beweis: Hierzu zeigen wir, dass der Index der Relationen \sim_{L_n} mindestens exponentiell ist.

Je zwei Wörter $x \neq y \in \{0, 1\}^n$ sind *nicht* äquivalent bzgl. der Relation \sim_{L_n} .

Dies ist wie folgt einsichtig. Sei $x = a_1 a_2 \dots a_n$ und $y = b_1 b_2 \dots b_n$ und $x \neq y$. Dann gibt es einen Index i mit $a_i \neq b_i$; etwa $a_i = 0$ und $b_i = 1$. Dann ist

$$x0^{i-1} \notin L_n \text{ und } y0^{i-1} \in L_n,$$

da a_i bzw. b_i das n -letzte Zeichen von $x0^{i-1}$ bzw. $y0^{i-1}$ ist. Also ist $x \not\sim_{L_n} y$.

Es folgt nun, dass $|\Sigma^*/L| \geq |\{0, 1\}^n| = 2^n$. □

6.4.2 Der Minimierungsalgorithmus

Der Ausgangspunkt ist ein DEA $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$. Zur Vereinfachung nehmen wir an, dass die Übergangsfunktion total ist und dass alle Zustände $q \in Q$ vom Anfangszustand erreichbar sind. Gesucht ist ein äquivalenter DEA mit minimaler Anzahl an Zuständen. Aus Satz 6.4.13 (Seite 266) folgt, dass wir einen zum Minimalautomaten \mathcal{M}_L isomorphen DEA konstruieren müssen.

¹⁴ Beachte: Äquivalenzklassen sind (per Definition) nicht leer. In diesem Fall ist $K_q \neq \emptyset$, da q von q_0 erreichbar ist.

Die Grobidee des Minimierungsalgorithmus ist, sukzessive Zustände zu identifizieren, die mit genau denselben Wörtern einen Endzustand erreichen. Das heißt wir bilden den Quotientenautomaten bzgl. der wie folgt definierten Äquivalenzrelation \equiv auf den Zuständen von \mathcal{M} .

Definition 6.4.15 [Der Quotientenautomat]

Sei \equiv folgende Äquivalenzrelation auf Q .

$$q \equiv p \text{ gdw für alle } z \in \Sigma^* \text{ gilt: } \delta(q, z) \in F \Leftrightarrow \delta(p, z) \in F.$$

Weiter sei

$$\mathcal{M}/\equiv = (Q/\equiv, \Sigma, \delta_{\equiv}, [q_0]_{\equiv}, F_{\equiv}),$$

wobei $F_{\equiv} = \{[q]_{\equiv} : q \in F\}$ und $\delta_{\equiv}([q]_{\equiv}, a) = [\delta(q, a)]_{\equiv}$.

Bevor wir fortfahren, müssen wir uns überlegen, dass die Übergangsfunktion des Quotientenautomaten *wohl definiert* ist. Hierzu ist zu zeigen, dass für je zwei Zustände q und p und $a \in \Sigma$ gilt:

$$\text{Aus } q \equiv p \text{ folgt } \delta(q, a) \equiv \delta(p, a).$$

Dies ist wie folgt einsichtig. Es gilt

$$\delta(\delta(q, a), z) = \delta(q, az) \in F \text{ genau dann, wenn } \delta(\delta(p, a), z) = \delta(p, az) \in F.$$

Lemma 6.4.16

\mathcal{M}/\equiv und \mathcal{M} sind äquivalent.

Beweis: Zunächst zeigen wir, dass der Übergang von \mathcal{M} zum Quotientenautomaten die akzeptierte Sprache unverändert lässt. Wir müssen zeigen, dass $\mathcal{L}(\mathcal{M}/\equiv) = \mathcal{L}(\mathcal{M})$. Im Folgenden schreiben wir kurz $[q]$ anstelle von $[q]_{\equiv}$. Zunächst überlegen wir uns, dass

$$q \in F \text{ genau dann, wenn } [q] \in F_{\equiv}.$$

Die Richtung \Rightarrow folgt sofort aus der Definition von F_{\equiv} . Für die Richtung \Leftarrow nehmen wir an, dass $[p] \in F_{\equiv}$. Dann gibt es einen Endzustand $q \in F$ mit $[q] = [p]$, also $q \equiv p$. Wir betrachten das Wort $z = \varepsilon$ und erhalten $p = \delta(p, \varepsilon) \in F$, da $\delta(q, \varepsilon) = q \in F$.

Sei $x = a_1, \dots, a_n \in \Sigma^*$ und q_0, q_1, \dots, q_n der zugehörige Lauf in \mathcal{M} . $[q_0], [q_1], \dots, [q_n]$ ist dann der zu x gehörende Lauf in \mathcal{M}/\equiv . Es gilt:

$$\begin{aligned}
 & x \in \mathcal{L}(\mathcal{M}) \\
 \text{gdw} & \quad q_0, q_1, \dots, q_n \text{ ist akzeptierend} \\
 \text{gdw} & \quad q_n \in F \\
 \text{gdw} & \quad [q_n] \in F_{\equiv} \\
 \text{gdw} & \quad [q_0], [q_1], \dots, [q_n] \text{ ist akzeptierend} \\
 \text{gdw} & \quad x \in \mathcal{L}(\mathcal{M}/\equiv). \quad \square
 \end{aligned}$$

Satz 6.4.17

\mathcal{M}/\equiv und der Minimalautomat von $\mathcal{L}(\mathcal{M})$ sind isomorph.

Beweis: Sei $L = \mathcal{L}(\mathcal{M})$. Wir zeigen, dass $\sim_L = \sim_{\mathcal{M}/\equiv}$. Da \mathcal{M}/\equiv die Sprache L akzeptiert (Lemma 6.4.16, Seite 268), ist die induzierte Äquivalenz $\sim_{\mathcal{M}/\equiv}$ eine Verfeinerung von \sim_L (Lemma 6.4.7, Seite 263). Wir zeigen nun, dass \sim_L eine Verfeinerung von $\sim_{\mathcal{M}/\equiv}$ ist. Seien $x, y \in \Sigma^*$. Wir schreiben $[q]$ anstelle von $[q]_{\equiv}$.

$$\begin{aligned}
 x \sim_L y & \\
 \implies & \quad \text{für alle } z \in \Sigma^* \text{ gilt: } xz \in L \text{ gdw } yz \in L \\
 \implies & \quad \text{für alle } z \in \Sigma^* \text{ gilt: } \delta(q_0, xz) \in F \text{ gdw } \delta(q_0, yz) \in F \\
 \implies & \quad \text{für alle } z \in \Sigma^* \text{ gilt: } \delta(\delta(q_0, x), z) \in F \text{ gdw } \delta(\delta(q_0, y), z) \in F \\
 \implies & \quad \delta(q_0, x) \equiv \delta(q_0, y) \\
 \implies & \quad \delta_{\equiv}([q_0], x) = \delta_{\equiv}([q_0], y) \\
 \implies & \quad x \sim_{\mathcal{M}/\equiv} y.
 \end{aligned}$$

Die Behauptung folgt nun aus Teil (a) von Satz 6.4.13 (Seite 266). \square

Der Minimierungsalgorithmus beruht auf der Idee, den Quotientenautomaten (anstelle des Minimalautomaten) zu berechnen. Die Grobidee besteht darin, sukzessive zweistellige Relationen

$$R_0 \supseteq R_1 \supseteq R_2 \supseteq \dots \supseteq \equiv$$

zu berechnen. Die initiale Relation R_0 ist eine Äquivalenzrelation, die alle Endzustände und alle Zustände in $Q \setminus F$ identifiziert. Im $(i+1)$ -ten Iterationsschritt entfernen wir aus der Relation R_i ein Zustandspaar (q, q') , sodass

$$(\delta(q, a), \delta(q', a)) \notin R_i \text{ für ein } a \in \Sigma,$$

und erhalten somit die Relation R_{i+1} . Wenn es kein solches Paar $(q, q') \in R_i$ gibt, dann ist $R_i = \equiv$ und das Verfahren hält an.

Die Korrektheit des Verfahrens beruht auf der Isomorphie von \mathcal{M}/\equiv und dem Minimalautomaten sowie der Beobachtung, dass die Äquivalenzrelation \equiv die größte Relation mit den folgenden Eigenschaften (1) und (2) ist.

- (1) Aus $q \equiv p$ folgt: $q \in F$ gdw $p \in F$.
 (2) Für alle $a \in \Sigma$ und $q \equiv p$ gilt: $\delta(q, a) \equiv \delta(p, a)$.

Wir formulieren den Algorithmus (der in der Literatur häufig als *table filling algorithm* bezeichnet wird) mit einer zweidimensionalen Tabelle, die die *ungeordneten* Zustandspaare verwaltet.¹⁵ Siehe Algorithmus 6.4.18. Für alle Zustandspaare (q, q') ist in der Tabelle entweder eine Markierung oder kein Eintrag. Die Markierungen deuten an, für welche Zustandspaare (q, q') *bereits nachgewiesen* ist, dass

$$\delta(q, a) \equiv \delta(q', a).$$

In der i -ten Iteration besteht die Relation R_i genau aus allen Paaren (q, q') , für die *kein* Eintrag in der Tabelle ist.

Algorithmus 6.4.18 [Der Minimierungsalgorithmus]

Erstelle eine Tabelle für alle ungeordneten Zustandspaare (q, q') .
 Markiere in der Tabelle alle Zustandspaare (q, q') mit $q \in F$ und $q' \notin F$ (oder umgekehrt).
WHILE \exists unmarkiertes Paar $(q, q') \exists a \in \Sigma$, sodass $(\delta(q, a), \delta(q', a))$ markiert ist **DO**
 wähle ein solches Paar und markiere es
OD
 Bilde maximale Mengen paarweise unmarkierter Zustände.

Der skizzierte Minimierungsalgorithmus lässt sich so implementieren, dass er die Laufzeit $\mathcal{O}(|Q|^2 \cdot |\Sigma|)$ hat.

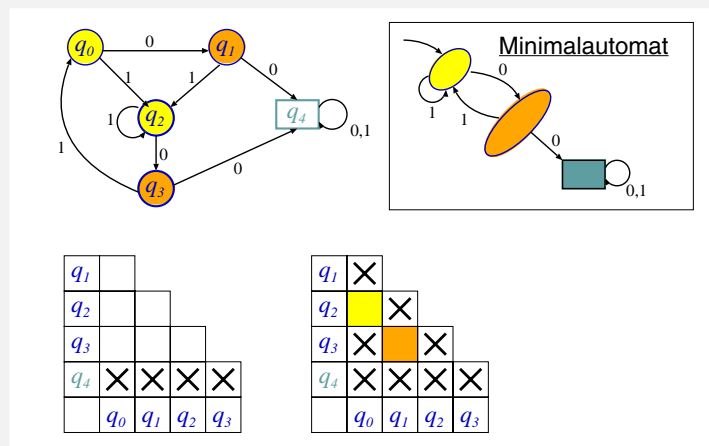
Zunächst berechnet der Minimierungsalgorithmus zwar nur die Äquivalenzklassen bzgl. \equiv ; jedoch ist aufgrund von Definition 6.4.15 (Seite 268) klar, wie die Übergangsfunktion und die Endzustandsmenge zu wählen ist, um einen minimalen DEA zu erhalten.

Beispiel 6.4.19. Wir veranschaulichen die Arbeitsweise des Minimierungsalgorithmus an einem Beispiel (siehe Abbildung 6.4.20).

Initial werden die Paare (q_4, q_0) , (q_4, q_1) , (q_4, q_2) und (q_4, q_3) markiert, da nur q_4 ein Endzustand ist. Dann werden (in irgendeiner Reihenfolge) die Paare (q_1, q_0) , (q_2, q_1) ,

¹⁵ Für eine Implementierung kann man eine feste Nummerierung q_0, q_1, \dots, q_{n-1} der Zustände zu Grunde legen und die Tabelle so erstellen, dass genau die geordneten Zustandspaare (q_l, q_j) mit $l > j$ dargestellt sind.

Abbildung 6.4.20 Table filling algorithm



(q_3, q_0) und (q_3, q_2) markiert, da jeweils nur ein Zustand jedes Paares einen Übergang zum Endzustand besitzt. ■

Zusammenfassung

Folgender Satz fasst die Charakterisierungen regulärer Sprachen, die wir in diesem Abschnitt kennen gelernt haben, zusammen.

Satz 6.4.21 [Charakterisierung regulärer Sprachen]

Sei L eine Sprache. Dann sind folgende Aussagen äquivalent:

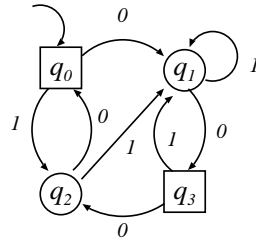
- L ist regulär, d.h. $L = \mathcal{L}(G)$ für eine reguläre Grammatik G .
- $L = \mathcal{L}(\mathcal{M})$ für einen DEA \mathcal{M} .
- $L = \mathcal{L}(\mathcal{M})$ für einen NEA \mathcal{M} .
- $L = \mathcal{L}(\alpha)$ für einen regulären Ausdruck α .
- Der Index von L ist endlich.

Darüber hinaus sind Syntaxdiagramme ein weiterer Formalismus, mit dem sich reguläre Sprachen charakterisieren lassen.

6.5 Übungen

Aufgabe 6.1 DEAs, NEAs und reguläre Grammatiken

Gegeben ist folgender DEA \mathcal{M} .



- Geben Sie eine reguläre Grammatik G mit $\mathcal{L}(G) = \mathcal{L}(\mathcal{M})$ an.
- Wenden Sie das vorgestellte Verfahren an, um aus der in (a) angegebenen regulären Grammatik einen NEA \mathcal{M}' zu konstruieren.
- Wenden Sie die Potenzmengenkonstruktion an und skizzieren Sie den resultierenden äquivalenten DEA \mathcal{M}'' .
- Wenden Sie das vorgestellte Verfahren an, um die Äquivalenz von \mathcal{M} und \mathcal{M}'' nachzuweisen.

Aufgabe 6.2 Entwurf von DEAs/NEAs

Geben Sie in (a)–(d) jeweils die präzisen Komponenten eines endlichen Automaten (wahlweise DEA, NEA oder ε -erweiterter NEA) an, der die angegebene Sprache akzeptiert.

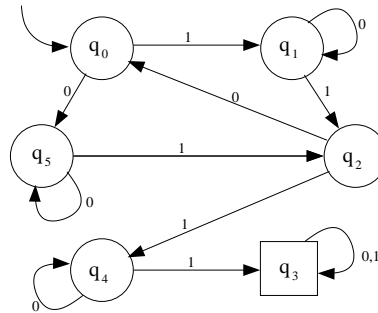
- Geben Sie einen endlichen Automaten \mathcal{M} an, dessen akzeptierte Sprache die Menge aller Wörter $w \in \{0, 1\}^*$ ist, die das Teilwort 0110 enthalten und mit einer Eins beginnen.
- Geben Sie einen endlichen Automaten \mathcal{M}' an, dessen akzeptierte Sprache die Menge aller Wörter $w \in \{0, 1, 2\}^*$ ist, die das Teilwort 0110 enthalten und mit einer Eins enden.
- Geben Sie einen endlichen Automaten \mathcal{M}'' an, dessen akzeptierte Sprache die Menge aller Wörter $w \in \{0, 1\}^*$ ist, die eine ungerade Anzahl an Einsen enthalten und für die die Anzahl an Nullen durch 3 teilbar ist.
- Geben Sie einen endlichen Automaten \mathcal{M}''' an, dessen akzeptierte Sprache die Menge

$$\mathcal{L}(\mathcal{M}''') = \{0110, 101, 11110\} \cup \{1^n : n \geq 1\}$$

ist.

Aufgabe 6.3 Minimierungsalgorithmus

Vollziehen Sie die Arbeitsweise des Minimierungsalgorithmus für DEAs (table filling algorithm) an folgendem Beispiel nach:



Skizzieren Sie anschließend den Minimalautomaten.

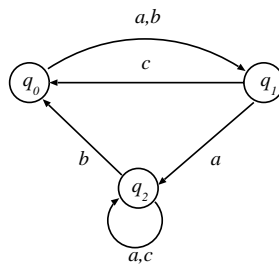
Aufgabe 6.4 Nicht reguläre Sprachen

Welche der folgenden Sprachen sind regulär? Begründen Sie Ihre Antwort.

- (a) $L_a = \{ww^R : w \in \{0, 1\}^*\}$
- (b) $L_b = \{a^n c^m b^n : n, m \geq 0\}$
- (c) $L_c = \{w \in \{0, 1\}^* : \text{Anz}(w, 0) \text{ ist gerade und } \text{Anz}(w, 1) \text{ ist durch 3 teilbar}\}$.
- (d) $L_d =$ Menge aller $w \in \{0, 1\}^*$, sodass auf jede Null eine Eins folgt.
- (e) $L_e = \{0^{n^2} : n \geq 0\}$
- (f) $\{0^m 1^n 0^{n+m} : n, m \geq 1\}$

Aufgabe 6.5 Reguläre Ausdrücke

- (a) Wenden Sie das vorgestellte Verfahren an, um aus dem folgenden DEA \mathcal{M} einen regulären Ausdruck α mit $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$ zu konstruieren.



(b) Gegeben sind die folgenden beiden regulären Ausdrücke.

▶ $\alpha_1 = (11 + 0)^*(00 + 1)^*$

▶ $\alpha_2 = (1 + 01 + 001)^*(\varepsilon + 0 + 00)$

(i) Skizzieren Sie für α_1 einen endlichen Automaten (DEA oder NEA, evt. mit ε -Übergängen), der die betreffende Sprache akzeptiert.

(ii) Skizzieren Sie ein Syntaxdiagramm für α_2 .

Aufgabe 6.6 Reguläre Ausdrücke

(a) Beweisen oder widerlegen Sie folgende Aussagen über reguläre Ausdrücke.

(i) $\alpha(\beta + \gamma)\delta \equiv \alpha\beta\delta + \alpha\gamma\delta$

(ii) $\varepsilon + \alpha^* \equiv \alpha^*$

(iii) $(\alpha\beta + \alpha)^*\alpha \equiv \alpha(\beta\alpha + \alpha)^*$

(iv) $(\alpha + \beta)^* \equiv \alpha^* + \beta^*$

(v) $\alpha(\alpha + \varepsilon)^* + \varepsilon \equiv \alpha^*$

(b) Geben Sie einen regulären Ausdruck für jede der folgenden regulären Sprachen an.

(i) L_a = Menge aller Wörter, die mit a beginnen und das Teilwort $bbca$ enthalten.

(ii) L_b = Menge aller Wörter, in denen das Teilwort $bbca$ nicht vorkommt.

(iii) L_c = Menge aller Wörter, die mit a beginnen, mit b enden und in denen das Wort $bbca$ mindestens zweimal vorkommt.

(iv) L_d = Menge aller Wörter w , die

▶ entweder mit aa beginnen und mit bb enden

▶ oder mit bb enden und in denen das Teilwort $bbca$ nicht vorkommt.

Aufgabe 6.7 Reguläre Sprachen

(a) Seien L, K reguläre Sprachen über einem Alphabet Σ . Zeigen Sie, dass

$$L/K = \{x \in \Sigma^* : xy \in L \text{ für ein } y \in K\}$$

regulär ist.

(b) Sei L eine reguläre Sprache über einem mindestens zweielementigen Alphabet Σ . Zeigen Sie, dass die folgenden Sprachen regulär sind.

(i) $L_1 = \{x \in L : \text{es gibt kein } y \in \Sigma^*, \text{ sodass } xy \in L\}$

(ii) $L_2 = \{x \in L : \text{kein echtes Präfix von } x \text{ liegt in } L\}$

(iii) $L_3 = \{x \in L : x \text{ ist Präfix eines Worts } y \in L\}$.

- (c) Seien Σ, Γ Alphabete und $h : \Sigma \rightarrow \Gamma^*$ eine Abbildung. Wir erweitern h zu einer (ebenfalls mit h bezeichneten) Abbildung

$$h : \Sigma^* \rightarrow \Gamma^*,$$

indem wir $h(\varepsilon) = \varepsilon$ und

$$h(a_1 \dots a_m) = h(a_1) \dots h(a_m)$$

setzen. Zeigen Sie:

- (i) Ist L eine reguläre Sprache über Σ^* , so ist $h(L) = \{h(w) : w \in L\}$ eine reguläre Sprache über Γ .
(ii) Ist K eine reguläre Sprache über Γ , so ist

$$h^{-1}(K) = \{w \in \Sigma^* : h(w) \in K\}$$

regulär.