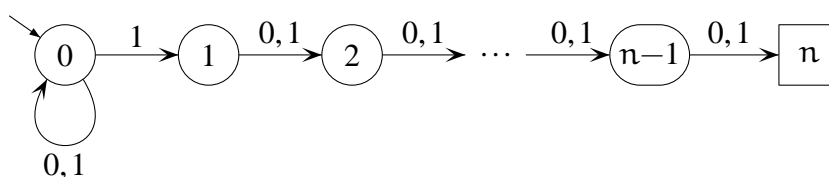


In Satz 2.15 (Seite 30) haben wir die Äquivalenz von DFA und NFA nachgewiesen. Durch die Potenzmengenkonstruktion angewandt auf gegebenen NFA  $\mathcal{M}$  kann ein DFA  $\mathcal{M}_{\text{det}}$  entstehen, der exponentiell größer als  $\mathcal{M}$  ist, selbst dann, wenn man alle unerreichbaren Zustände entfernt. Insbesondere ist auch der zeitliche Aufwand für die Konstruktion exponentiell in der Größe des NFA. Das exponentielle Blowup beim Übergang von einem NFA zu einem äquivalenten DFA lässt sich jedoch im allgemeinen nicht verhindern, da es Sprachen gibt, für die die Darstellung durch einen NFA sehr viel effizienter als durch einen DFA ist. Als Beispiel hierzu betrachten wir die Sprachfamilie  $(L_n)_{n \geq 1}$ , wobei

$$\begin{aligned} L_n &= \{ w \in \{0, 1\}^* : \text{das } n\text{-letzte Zeichen von } w \text{ ist eine "1"} \} \\ &= \{ x1y : x, y \in \{0, 1\}^*, |y| = n-1 \} \end{aligned}$$

Die Sprache  $L_n$  wird durch folgenden NFA mit  $n+1$  Zuständen der folgenden Form akzeptiert:



Wir zeigen nun mit Hilfe des Satzes von Myhill & Nerode, dass jeder DFA für  $L_n$  mindestens exponentiell viele Zustände hat.

**Lemma 2.45.** *Jeder DFA für  $L_n$  hat mindestens  $2^n$  Zustände.*

*Beweis.* Es genügt zu zeigen, dass der Nerode-Index der Sprachen  $L_n$  mindestens exponentiell ist. Genauer gilt:

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

Hieraus folgt dann, dass  $|\Sigma^*/L_n| \geq |\{0, 1\}^n| = 2^n$ .

Wir zeigen nun, dass für je zwei Wörter  $x, y \in \{0, 1\}^n$  mit  $x \neq y$  gilt  $x \not\sim_L y$ . 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  $a_i = 1$  und  $b_i = 0$ . Dann ist  $x0^{i-1} \in L_n$  und  $y0^{i-1} \notin L_n$ , da  $a_i$  bzw.  $b_i$  das  $n$ -letzte Zeichen von  $x0^{i-1}$  bzw.  $y0^{i-1}$  ist. Also ist  $x \not\sim_{L_n} y$ . □

**Der Minimierungsalgorithmus.** Der Ausgangspunkt ist ein DFA  $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ . 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 DFA mit minimaler Anzahl an Zuständen. Aus Satz 2.43 (Seite 65) folgt, dass wir einen zum Minimalautomaten  $\mathcal{M}_L$  isomorphen DFA konstruieren müssen. Die Idee des Minimierungsalgorithmus besteht darin, sukzessive Zustände in  $\mathcal{M}$  zu identifizieren, die mit genau denselben Wörtern einen Endzustand erreichen. D.h., wir bilden den Quotienten-DFA unter der wie folgt definierten Äquivalenzrelation  $\equiv$  auf den Zuständen von  $\mathcal{M}$ .

**Definition 2.46 (Quotienten-DFA  $\mathcal{M}/\equiv$ ).** Sei  $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$  ein totaler DFA. Sei  $\equiv$  die wie folgt definierte Äquivalenzrelation auf dem Zustandsraum  $Q$  von  $\mathcal{M}$ :

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

Der Quotienten-DFA von  $\mathcal{M}$  unter  $\equiv$  ist wie folgt definiert:

$$\mathcal{M}/\equiv \stackrel{\text{def}}{=} (Q/\equiv, \Sigma, \delta', [q_0]_{\equiv}, F'),$$

wobei  $F' \stackrel{\text{def}}{=} \{ [q]_{\equiv} : q \in F \}$  und  $\delta'([q]_{\equiv}, a) \stackrel{\text{def}}{=} [\delta(q, a)]_{\equiv}$ . ■

Bevor wir fortfahren, müssen wir uns überlegen, dass  $\equiv$  tatsächlich eine Äquivalenzrelation ist und dass die Übergangsfunktion des Quotienten-DFA *wohldefiniert* ist. Ersteres ist offensichtlich. Für die Wohldefiniertheit der Übergangsfunktion  $\delta'$  ist zu zeigen, dass für alle  $q, p \in Q$  und  $a \in \Sigma$  gilt:

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

Dies ist wie folgt einsichtig. Wir setzen  $q \equiv p$  voraus. Zu zeigen ist nun, dass die Zustände  $q' = \delta(q, a)$  und  $p' = \delta(p, a)$  unter  $\equiv$  zueinander äquivalent sind. Hierzu betrachten wir ein beliebiges Wort  $z \in \Sigma^*$ . Dann gilt  $\delta(q', z) = \delta(q, az)$  und  $\delta(p', z) = \delta(p, az)$ . Hieraus folgt:

$$\begin{aligned} \delta(q', z) \in F & \quad \text{gdw} \quad \delta(q, az) \in F & \quad (\text{da } \delta(q', z) = \delta(q, az)) \\ & \quad \text{gdw} \quad \delta(p, az) \in F & \quad (\text{da } q \equiv p) \\ & \quad \text{gdw} \quad \delta(p', z) \in F & \quad (\text{da } \delta(p', z) = \delta(p, az)) \end{aligned}$$

**Lemma 2.47 (Korrektheit des Quotienten-DFA).**  $\mathcal{M}/\equiv$  und  $\mathcal{M}$  sind äquivalent.

*Beweis.* Zunächst zeigen wir, dass der Übergang von  $\mathcal{M}$  zum Quotienten-DFA die akzeptierte Sprache unverändert lässt. D.h.,  $\mathcal{L}(\mathcal{M}/\equiv) = \mathcal{L}(\mathcal{M})$ . Im Folgenden schreiben wir kurz  $[q]$  anstelle von  $[q]_{\equiv}$ . Zunächst überlegen wir uns, dass

$$q \in F \quad \text{gdw} \quad [q] \in F'$$

Die Richtung “ $\implies$ ” folgt sofort aus der Definition von  $F'$ . Für die Richtung “ $\impliedby$ ” nehmen wir an, dass  $[p] \in F'$ . 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  $z = a_1 \dots a_n \in \Sigma^*$  und  $q_0 q_1 \dots q_n$  der zugehörige Lauf in  $\mathcal{M}$ .  $[q_0] [q_1] \dots [q_n]$  ist dann der Lauf für  $z$  im Quotienten-DFA  $\mathcal{M}/\equiv$ . Dies kann durch Induktion nach der Wortlänge  $n$  gezeigt werden. Hieraus folgt nun die gewünschte Eigenschaft:

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

□

Im Beweis des vorangegangenen Lemmas haben wir die Beobachtung gemacht, dass Läufe in  $\mathcal{M}$  zu Läufen im Quotienten-DFA geliftet werden können, indem man von der Zustandsfolge  $q_0 q_1 \dots q_n$  zu der Folge der Äquivalenzklassen  $[q_0]_{\equiv} [q_1]_{\equiv} \dots [q_n]_{\equiv}$  übergeht. Für die erweiterte Übergangsfunktion im Quotienten-DFA folgt hieraus:

$$\delta'([q_0]_{\equiv}, z) = [\delta(q_0, z)]_{\equiv} \quad \text{für alle Wörter } z \in \Sigma^*$$

Mit der Annahme, dass alle Zustände  $q$  in  $\mathcal{M}$  vom Anfangszustand  $q_0$  erreichbar sind, also die Form  $q = \delta(q_0, z)$  für ein Wort  $z$  haben, gilt nun auch, dass alle Zustände des Quotienten-DFA vom Anfangszustand  $[q_0]_{\equiv}$  erreichbar sind.

**Satz 2.48 (Isomorphie des Quotienten-DFA und Minimalautomaten).**  $\mathcal{M}/_{\equiv}$  und der Minimalautomat von  $\mathcal{L}(\mathcal{M})$  sind isomorph.

*Beweis.* Sei  $L = \mathcal{L}(\mathcal{M})$ . Wir zeigen, dass die Nerode-Äquivalenz  $\sim_L$  mit der DFA-Äquivalenz  $\sim_{\mathcal{M}/_{\equiv}}$  des Quotienten-DFA übereinstimmt. Hierzu ist die gegenseitige Verfeinerung beider Äquivalenzrelationen zu zeigen:

(1)  $\sim_{\mathcal{M}/_{\equiv}}$  ist feiner als  $\sim_L$

(2)  $\sim_L$  ist feiner als  $\sim_{\mathcal{M}/_{\equiv}}$

Aussage (1) ist eine Folgerung der vorangegangenen Ergebnisse. Da  $\mathcal{M}/_{\equiv}$  die Sprache  $L$  akzeptiert (Lemma 2.47, Seite 68), ist somit die induzierte DFA-Äquivalenz  $\sim_{\mathcal{M}/_{\equiv}}$  eine Verfeinerung von  $\sim_L$  (Lemma 2.38, Seite 63).

Nun zum Beweis von Aussage (2). Zu zeigen ist, dass die Nerode-Äquivalenz  $\sim_L$  eine Verfeinerung der DFA-Äquivalenz  $\sim_{\mathcal{M}/_{\equiv}}$  ist. Wie zuvor schreiben wir  $[q]$  anstelle von  $[q]_{\equiv}$ . Seien  $x, y \in \Sigma^*$  mit  $x \sim_L y$ . Dann gilt für alle Wörter  $z \in \Sigma^*$ :

$$\begin{array}{ccc} xz \in L & \text{gdw} & yz \in L \\ \text{und somit: } \delta(q_0, xz) \in F & \text{gdw} & \delta(q_0, yz) \in F \\ \uparrow & & \uparrow \\ \delta(\delta(q_0, x), z) & & \delta(\delta(q_0, y), z) \end{array}$$

Mit  $p_x = \delta(q_0, x)$  und  $p_y = \delta(q_0, y)$  gilt also:

$$\delta(p_x, z) \in F \quad \text{gdw} \quad \delta(q_0, xz) \in F \quad \text{gdw} \quad \delta(q_0, yz) \in F \quad \text{gdw} \quad \delta(p_y, z) \in F$$

Da diese Aussage für beliebiges  $z$  gilt, folgt  $p_x = \delta(q_0, x) \equiv \delta(q_0, y) = p_y$ . Für die erweiterte Übergangsfunktion im Quotienten-DFA ergibt sich hieraus:

$$\delta'([q_0], x) = [\delta(q_0, x)] = [\delta(q_0, y)] = \delta'([q_0], y)$$

Also sind  $x$  und  $y$  DFA-äquivalent bezogen auf den Quotienten-DFA, d.h.,  $x \sim_{\mathcal{M}/_{\equiv}} y$ .

Die Voraussetzung, dass die Übergangsfunktion von  $\mathcal{M}$  total ist und dass alle Zustände  $q \in Q$  vom Anfangszustand  $q_0$  erreichbar sind (d.h., die Form  $q = \delta(q_0, z)$  für ein  $z \in \Sigma^*$  haben), überträgt sich auf den Quotienten-DFA (siehe oben). Die Isomorphie von  $\mathcal{M}/_{\equiv}$  und  $\mathcal{M}_L$  ist daher eine Folgerung aus der Eindeutigkeit des Minimalautomaten, siehe Teil (a) von Satz 2.43 auf Seite 65.  $\square$

Der Minimierungsalgorithmus startet mit einem totalen DFA  $\mathcal{M}$  ohne unerreichbare Zustände und berechnet nun die Äquivalenzrelation  $\equiv$ . Durch Zustandsaggregation (d.h., Zusammenfassen aller Zustände einer  $\equiv$ -Äquivalenzklasse) erhält man dann den Quotienten-DFA, also einen zum Minimalautomaten isomorphen DFA. Die Grobidee für die Berechnung von  $\equiv$  besteht darin, sukzessive zweistellige Relationen

$$\mathcal{R}_0 \supseteq \mathcal{R}_1 \supseteq \mathcal{R}_2 \supseteq \dots \supseteq \equiv$$

zu berechnen. Die Relationen  $\mathcal{R}_i$  werden also zunehmend feiner und erfüllen die Eigenschaft:

$$\text{Aus } \langle q, q' \rangle \in Q^2 \setminus \mathcal{R}_i \text{ folgt } q \not\equiv q'.$$

Die initiale Relation  $\mathcal{R}_0$  ist eine Äquivalenzrelation, die alle Endzustände und alle Zustände in  $Q \setminus F$  identifiziert. Hier wird die Tatsache ausgenutzt, dass Endzustände niemals zu einem Zustand äquivalent sein können, der kein Endzustand ist. Im  $(i+1)$ -ten Iterationsschritt entfernen wir aus der Relation  $\mathcal{R}_i$  ein Zustandspaar  $\langle q, q' \rangle$ , so dass

$$\langle \delta(q, a), \delta(q', a) \rangle \notin \mathcal{R}_i \text{ für ein } a \in \Sigma,$$

und erhalten somit die Relation  $\mathcal{R}_{i+1}$ . In diesem Fall gilt  $\delta(q, a) \not\equiv \delta(q', a)$ . Also gibt es ein Wort  $z \in \Sigma^*$ , so dass

$$\delta(q, az) = \delta(\delta(q, a), z) \in F \text{ und } \delta(q', az) = \delta(\delta(q', a), z) \notin F,$$

oder umgekehrt. In jedem Fall gilt dann  $q \not\equiv q'$ . Dies rechtfertigt es, das Zustandspaar  $\langle q, q' \rangle$  aus  $\mathcal{R}_i$  zu entfernen. Falls es kein solches Paar  $\langle q, q' \rangle \in \mathcal{R}_i$  und  $a \in \Sigma$  mit  $(\delta(q, a), \delta(q', a)) \notin \mathcal{R}_i$  gibt, dann hält das Verfahren an. Für die finale Relation  $\mathcal{R}_i$  gilt nun tatsächlich  $\mathcal{R}_i = \equiv$ . Dies ist wie folgt einsichtig. Die finale Relation  $\mathcal{R}_i$  hat die Eigenschaft, dass für alle Zustandspaare  $\langle q, q' \rangle \in \mathcal{R}_i$  gilt:

$$\langle \delta(q, a), \delta(q', a) \rangle \in \mathcal{R}_i \text{ für alle } a \in \Sigma \text{ und entweder } \{q, q'\} \cap F = \emptyset \text{ oder } \{q, q'\} \subseteq F.$$

Durch Induktion nach  $n$  kann nun gezeigt werden, dass für alle  $\langle q, q' \rangle \in \mathcal{R}_i$  und alle Wörter  $z$  mit  $|z| = n$  gilt:

$$\delta(q, z) \in F \text{ gdw } \delta(q', z) \in F$$

Also ist  $q \equiv q'$  für alle  $\langle q, q' \rangle \in \mathcal{R}_i$ . Die finale Relation  $\mathcal{R}_j$  ist also feiner als die Äquivalenz  $\equiv$ . Wie oben erwähnt gilt andererseits, dass  $\equiv$  feiner als  $\mathcal{R}_i$  ist. Daher stimmt die finale Relation  $\mathcal{R}_i$  mit  $\equiv$  überein.

Wir formulieren den Algorithmus (der in der Literatur häufig als *table filling algorithm* bezeichnet wird) mit einer zweidimensionalen Tabelle, die die *ungeordneten* Zustandspaare verwaltet. Siehe Algorithmus 2. Für eine Implementierung kann man eine feste Nummerierung  $q_0, q_1, \dots, q_n$  der Zustände zugrundelegen und die Tabelle so erstellen, dass genau die geordneten Zustandspaare  $\langle q_i, q_k \rangle$  mit  $i > k$  dargestellt sind. Für alle ungeordneten Zustandspaare  $\langle q_i, q_k \rangle$  ist in der Tabelle entweder eine Markierung oder kein Eintrag. Die Markierungen deuten an, für welche Zustandspaare  $\langle q_i, q_k \rangle$  *bereits nachgewiesen* ist, dass  $q_i \not\equiv q_k$ . In der  $j$ -ten Iteration besteht die Relation  $\mathcal{R}_j$  genau aus allen Paaren  $\langle q_i, q_k \rangle$ , für die *kein* Eintrag in der Tabelle ist.

Es ist klar, dass die Anzahl an Iterationen durch  $\mathcal{O}(|Q|^2)$  beschränkt ist. Mit einigen Tricks, die hier nicht besprochen werden, lässt sich das Verfahren jedoch effizienter implementieren.

---

**Algorithmus 2** Minimierungsalgorithmus für DFA (table filling algorithm)
 

---

Erstelle eine Tabelle für alle ungeordneten Zustandspaare  $\langle q_i, q_k \rangle$ .

Markiere in der Tabelle alle Paare  $\langle q_i, q_k \rangle$  mit  $\{q_i, q_k\} \cap F \neq \emptyset$  und  $\{q_i, q_k\} \setminus F \neq \emptyset$ .

**WHILE** es gibt ein unmarkiertes Paar  $\langle q_i, q_k \rangle$  und ein  $a \in \Sigma$ ,  
 so dass  $\delta(q_i, a) \neq \delta(q_k, a)$  und das Paar  $\langle \delta(q_i, a), \delta(q_k, a) \rangle$  markiert ist **DO**  
 markiere  $\langle q_i, q_k \rangle$

**OD**

Bilde maximale Mengen paarweise unmarkierter Zustände.

---

**Beispiel 2.49 (Minimierungsalgorithmus).** Der Minimierungsalgorithmus angewandt auf den DFA  $\mathcal{M}$  links in Abbildung 23 markiert zunächst die Paare  $\langle q_4, q_0 \rangle$ ,  $\langle q_4, q_1 \rangle$ ,  $\langle q_4, q_2 \rangle$  und  $\langle q_4, q_3 \rangle$ , da nur  $q_4$  ein Endzustand ist. Dann werden – in irgendeiner Reihenfolge – die Paare  $\langle q_1, q_0 \rangle$ ,  $\langle q_2, q_1 \rangle$ ,  $\langle q_3, q_0 \rangle$  und  $\langle q_3, q_2 \rangle$  markiert, da jeweils nur ein Zustand jedes Paares einen Übergang zum Endzustand besitzt. Folgende Tabelle fasst das Ergebnis der Markierungsschritte zusammen.

$q_1$	X			
$q_2$		X		
$q_3$	X		X	
$q_4$	X	X	X	X
	$q_0$	$q_1$	$q_2$	$q_3$

Letztendlich sind also  $q_0$  und  $q_2$  äquivalente Zustände, sowie auch  $q_1$  und  $q_3$ . Der Quotienten-DFA ergibt sich nun, indem  $q_0, q_2$  bzw.  $q_1, q_3$  zu je einem Zustand zusammengefasst werden. Man erhält den Automaten  $\mathcal{M}/\equiv$  rechts in Abbildung 23. ■

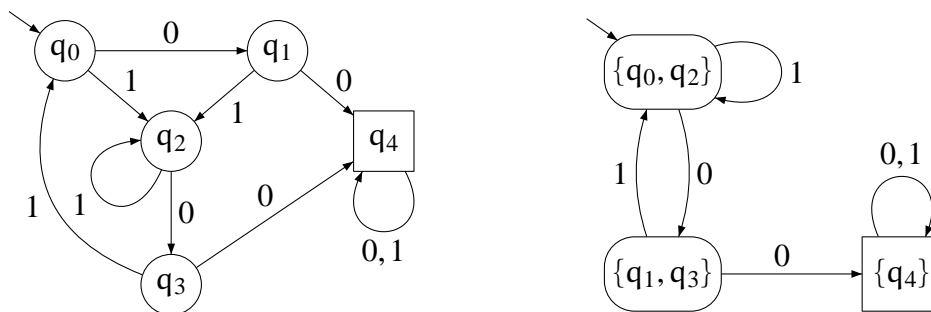


Abbildung 23: DFA  $\mathcal{M}$  und der Quotienten-DFA  $\mathcal{M}/\equiv$

## Zusammenfassung

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

**Satz 2.50 (Charakterisierung regulärer Sprachen).** Sei  $L$  eine Sprache. Dann sind folgende Aussagen äquivalent:

- (a)  $L$  ist regulär, d.h.  $L = \mathcal{L}(G)$  für eine reguläre Grammatik  $G$ .
- (b)  $L = \mathcal{L}(\mathcal{M})$  für einen DFA  $\mathcal{M}$ .
- (c)  $L = \mathcal{L}(\mathcal{M})$  für einen NFA  $\mathcal{M}$ .
- (d)  $L = \mathcal{L}(\mathcal{M})$  für einen  $\varepsilon$ -NFA  $\mathcal{M}$ .
- (e)  $L = \mathcal{L}(\alpha)$  für einen regulären Ausdruck  $\alpha$ .
- (f) Der Nerode-Index von  $L$  ist endlich.

In diesem Sinn sind die genannten Formalismen reguläre Grammatiken, DFA, NFA,  $\varepsilon$ -NFA und reguläre Ausdrücke gleichwertig. Darüberhinaus haben wir *konstruktive* Methoden angegeben, wie man die Formalismen ineinander überführen kann. Die meisten der Konstruktionen sind effizient. Ausnahmen hiervon ist die Determinisierung (und Komplementierung) von NFA mit der Potenzmengenkonstruktion sowie die Konstruktion regulärer Ausdrücke für gegebenen endlichen Automaten. Weiter hatten wir gesehen, dass die Klasse der regulären Sprachen unter allen gängigen Operatoren für formale Sprachen abgeschlossen ist und dass einige grundlegende algorithmische Problemstellungen recht einfach mit endlichen Automaten gelöst werden können.