

Abbildung 18: Idee für den Kleeneoperator für NFA

Für den Plusoperator kann man die Gleichung  $L^+ = L \circ L^*$  verwenden und  $+$  mittels Konkatenation und Kleeneabschluss für  $\varepsilon$ -NFA realisieren. Alternativ kann man auch einen  $\varepsilon$ -NFA  $\mathcal{M}^+$  genau wie für den Kleeneabschluss konstruieren und lediglich auf den zusätzlichen Anfangs- und Endzustand  $q_\varepsilon$  verzichten.  $\mathcal{M}^+$  geht also aus  $\mathcal{M}$  hervor, indem  $\varepsilon$ -Transitionen von allen End- zu allen Anfangszuständen eingefügt werden.

**Satz 2.23.** *Die Klasse der regulären Sprachen ist unter Vereinigung, Konkatenation, Kleeneabschluss, Komplementbildung und Durchschnitt abgeschlossen.*

### Größe der konstruierten endlichen Automaten

Die oben angegebenen Operatoren für endliche Automaten stellen zugleich algorithmische Verfahren zur Realisierung der fünf Kompositionsoperatoren für reguläre Sprachen dar, die durch endliche Automaten gegeben sind. Wir diskutieren nun die Größe der konstruierten endlichen Automaten gemessen an der Anzahl an Zuständen. Ist  $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$  ein  $\varepsilon$ -NFA (oder NFA oder DFA), so bezeichnet  $|\mathcal{M}|$  die Größe des Zustandsraums von  $\mathcal{M}$ , also

$$|\mathcal{M}| \stackrel{\text{def}}{=} |Q| = \text{Anzahl an Zustände in } \mathcal{M}$$

Die Effizienz von Algorithmen, die mit endlichen Automaten operieren, wird gemessen an der Anzahl an Zuständen plus Anzahl an Transitionen. Hierfür verwenden wir auch die Bezeichnung  $size(\mathcal{M})$ . Der exakte Wert von  $size(\mathcal{M})$  ist also  $|Q|$  plus die Anzahl an Tripeln  $(q, a, p) \in Q \times (\Sigma \cup \{\varepsilon\}) \times Q$  mit  $p \in \delta(q, a)$ . Dies entspricht der Größe von  $\mathcal{M}$  in der für Graphen bekannten Adjazenzlistendarstellung. Offenbar gilt

$$|\mathcal{M}| \leq size(\mathcal{M}) \leq |Q| + |Q|^2 \cdot (|\Sigma| + 1) = \mathcal{O}(|\mathcal{M}|^2),$$

wobei sich die Angabe der Größenordnung mit dem Operator  $\mathcal{O}$  auf ein festes Alphabet bezieht. Je nach Kontext bezeichnen wir  $|\mathcal{M}|$  oder  $size(\mathcal{M})$  als Größe des  $\varepsilon$ -NFA  $\mathcal{M}$ .

Wir diskutieren nun die Größe der endlichen Automaten, die durch den Einsatz der oben erläuterten Operatoren entstehen. Wir konzentrieren uns hier auf die Größe des Zustandsraums. Analoge Betrachtungen können für den Größenoperator  $size(\mathcal{M})$  durchgeführt werden. Die beschriebene Transformation eines  $\varepsilon$ -NFA  $\mathcal{M}$  in einen äquivalenten NFA  $\mathcal{M}'$  hat keinen Einfluss auf die Zustände. Es gilt also  $|\mathcal{M}| = |\mathcal{M}'|$ . Die Potenzmengenkonstruktion, die einen NFA  $\mathcal{M}$  in einen äquivalenten DFA  $\mathcal{M}_{\text{det}}$  überführt, verursacht ein exponentielles Blowup. Mit der im Beweis von Satz 2.15 auf Seite 30 angegebenen Definition von  $\mathcal{M}_{\text{det}}$  gilt

$$|\mathcal{M}_{\text{det}}| = 2^{|\mathcal{M}|}.$$

Oftmals sind zwar nicht alle Teilmengen  $P$  der Zustandsmenge  $Q$  von  $\mathcal{M}$  in  $\mathcal{M}_{\text{det}}$  erreichbar, jedoch werden wir später sehen, dass das exponentielle Wachstum für den Übergang von NFA zu äquivalenten DFA unvermeidbar sein kann.

Nun zu den Kompositionsoperatoren. Die Größe des Zustandsraums des NFA  $\mathcal{M}_1 \uplus \mathcal{M}_2$  für die Vereinigung ist  $|\mathcal{M}_1| + |\mathcal{M}_2|$ . Durch die Realisierung des Durchschnitts mittels des Produktoperators  $\otimes$  ergibt sich ein NFA mit  $|\mathcal{M}_1| \cdot |\mathcal{M}_2|$  Zuständen. Wendet man eine on-the-fly Konstruktion an, die nur den erreichbaren Teil des Produktautomaten konstruiert, so kann der Zustandsraum des resultierenden Automaten für die Durchschnittssprache selbstverständlich kleiner als  $|\mathcal{M}_1| \cdot |\mathcal{M}_2|$  sein. Dasselbe gilt für den modifizierten Produkt-Operator als Vereinigungsoperator für DFA. Der Komplementoperator für DFA lässt den Zustandsraum unverändert. Für jeden DFA  $\mathcal{M}$  gilt also  $|\overline{\mathcal{M}}| = |\mathcal{M}|$ . Startet man mit einem NFA  $\mathcal{M}$  und wendet zunächst die Potenzmengenkonstruktion an, um dann den Komplementbildung einzusetzen, so ist ein exponentielles Blowup möglich. Die Operationen Konkatenation und Kleeneabschluss sind für NFA oder  $\varepsilon$ -NFA effizient. Es gilt  $|\mathcal{M}_1 \circ \mathcal{M}_2| = |\mathcal{M}_1| + |\mathcal{M}_2|$  und  $|\mathcal{M}^*| = |\mathcal{M}| + 1$ .

## 2.2.2 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 2.24 (Pumping Lemma für reguläre Sprachen).** *Sei  $L \subseteq \Sigma^*$  eine reguläre Sprache. Dann gibt es eine ganze Zahl  $n \geq 1$ , so dass jedes Wort  $z \in L$  mit  $|z| \geq n$  wie folgt zerlegt werden kann:  $z = uvw$  mit Wörtern  $u, v, w \in \Sigma^*$ , so dass*

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

*Beweis.* Da  $L$  regulär ist, gibt es einen NFA  $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$  mit  $\mathcal{L}(\mathcal{M}) = L$  (Corollar 2.19 auf Seite 35). Sei  $|Q| = n$ . Wir zeigen nun, dass jedes Wort in  $\mathcal{L}(\mathcal{M})$  der Länge  $\geq n$  in Teilwörter  $u, v, w$  zerlegt werden kann, so dass die oben genannten Bedingungen (1), (2) und (3) erfüllt sind. Sei

$$z = a_1 a_2 \dots a_m \in \mathcal{L}(\mathcal{M}) \text{ mit } |z| = m \geq n$$

und sei  $q_0 q_1 \dots q_m$  ein akzeptierender Lauf für  $z$  in  $\mathcal{M}$ . Dann ist  $q_m \in F$ . Wir betrachten nur die ersten  $n+1$  Zustände dieses Laufs. Da  $\mathcal{M}$  genau  $n$  Zustände hat, gibt es Indizes  $i, j$  mit  $0 \leq i < j \leq n$ , so dass  $q_i = q_j$ . Wir zerlegen  $z$  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.

- (2) Wegen  $i < j$  gilt  $|v| \geq 1$ .

(3) Wegen  $j \leq n$  gilt  $|uv| = |a_1 \dots a_j| = j \leq n$ .

(1) Sei  $k \in \mathbb{N}$ . Wir betrachten das Wort

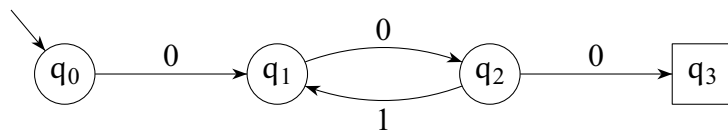
$$uv^k w = \underbrace{a_1 \dots a_i}_{=u} \underbrace{a_{i+1} \dots a_j}_{=v} \underbrace{a_{i+1} \dots a_j}_{=v} \dots \underbrace{a_{i+1} \dots a_j}_{=v} \underbrace{a_{j+1} \dots a_m}_{=w}$$

Das Wort  $uv^k w$  hat einen akzeptierenden Lauf der 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  $uv^k w \in \mathcal{L}(\mathcal{M}) = L$ . □

Wir machen uns die Aussage von Satz 2.24 an einem einfachen Beispiel klar. Wir betrachten folgenden DFA  $\mathcal{M}$  für die Sprache  $L = \{0(01)^m 00 : m \in \mathbb{N}\}$ .



Die Konstante  $n$  aus dem Pumping Lemma kann hier  $n = 4$  (Anzahl an Zustände) gewählt werden. Jedes Wort  $x \in \mathcal{L}(\mathcal{M})$  der Länge  $\geq 4$  hat die Form  $x = uvw$ , wobei

$$u = 0, \quad v = 01 \quad \text{und} \quad w = (01)^m 00$$

und (1)  $v \neq \varepsilon$ , (2)  $|uv| = 3 < 4 = n$  sowie die Pump-Eigenschaft (3)  $uv^k w = 0(01)^k w \in \mathcal{L}(\mathcal{M}) = L$  für alle  $k \in \mathbb{N}$ .

**Nicht-reguläre Sprachen.** Das Pumping Lemma kann für den Nachweis verwendet werden, dass eine Sprache nicht regulär ist. Als Beispiel betrachten wir

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

Eine informelle Erklärung, warum  $L$  nicht regulär, ergibt sich daraus, dass das Modell endlicher Automaten nicht mächtig genug ist, um  $L$  darzustellen. 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 ein endlicher Automat zwar prüfen, ob das Eingabewort von der Form  $a^n b^m$  ist. Ein endlicher Automat kann jedoch nicht die präzise Anzahl der gelesenen  $a$ 's speichern, um dann zu prüfen, ob ebensoviele  $b$ 's folgen.

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

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

Widerspruch zu Eigenschaft (3).

Ebenfalls mit Hilfe 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. Ein andere Methode, um den Nachweis zu erbringen, dass  $L'$  nicht regulär ist, geht wie folgt. Offenbar ist

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

Die Sprache  $L''$  ist regulär. Dies kann man z.B. durch die Angabe eines endlichen Automaten  $\mathcal{M}$  für  $L''$  belegen.

Hierzu genügt ein DFA mit zwei Zuständen  $q_a$  und  $q_b$ , die beide als Endzustände deklariert sind. Der Anfangszustand ist  $q_a$ . Weiter ist  $\delta(q_a, a) = q_a$ ,  $\delta(q_a, b) = \delta(q_b, b) = q_b$  und  $\delta(q_b, a) = \perp$ .

Wäre nun  $L'$  regulär, dann wäre (gemäß der Aussage von Satz 2.23, Seite 44) 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. Wir erwähnen ohne Beweis, dass es nicht-reguläre Sprachen gibt, welche das im Pumping Lemma angegebene Kriterium erfüllen.

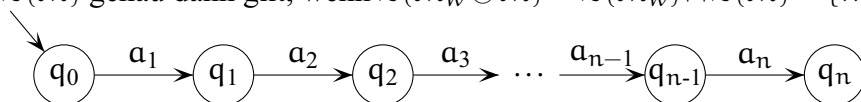
### 2.2.3 Algorithmen für endliche Automaten

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

**Das Wortproblem.** Ist  $\mathcal{M}$  ein DFA mit dem Alphabet  $\Sigma$  und  $w$  ein Wort über  $\Sigma$ , dann kann die Frage nach dem Wortproblem der akzeptierten Sprache von  $\mathcal{M}$ , d.h., "gilt  $w \in \mathcal{L}(\mathcal{M})$ ?" in linearer Zeit  $\mathcal{O}(|w|)$  beantwortet werden. Wir müssen lediglich den DFA  $\mathcal{M}$  bei Eingabe  $w$  simulieren. Für NFA kann der für  $w = a_1 a_2 \dots a_n$  relevante Teil des Potenzmengen-DFA  $\mathcal{M}_{\text{det}}$  konstruiert werden, was einer Berechnung von  $\delta(Q_0, w)$  für die erweiterte Übergangsfunktion im NFA mit der Berechnungsvorschrift

$$P_0 := Q_0 \text{ und } P_{i+1} := \bigcup_{p \in P_i} \delta(p, a_{i+1}) \text{ für } i = 0, 1, \dots, n-1.$$

Anschliessend ist zu prüfen, ob  $P_n = \delta(Q_0, w)$  wenigstens einen Endzustand von  $\mathcal{M}$  enthält. Alternativ kann man einen DFA  $\mathcal{M}_w$  für die einelementige Sprache  $\{w\}$  konstruieren (siehe Skizze unten), dann das Produkt  $\mathcal{M}_w \otimes \mathcal{M}$  bilden und anschliessend einen Nichtleerheitstest (siehe unten) auf den Produkt-NFA anwenden. Dieses Verfahren beruht auf der Beobachtung, dass  $w \in \mathcal{L}(\mathcal{M})$  genau dann gilt, wenn  $\mathcal{L}(\mathcal{M}_w \otimes \mathcal{M}) = \mathcal{L}(\mathcal{M}_w) \cap \mathcal{L}(\mathcal{M}) = \{w\} \cap \mathcal{L}(\mathcal{M}) = \{w\}$ .



**Leerheitstest.** Die Frage “gilt  $\mathcal{L}(\mathcal{M}) = \emptyset$ ?” für einen NFA (oder DFA)  $\mathcal{M}$  lässt sich mit einer Erreichbarkeitsanalyse realisieren. Wir müssen lediglich prüfen, ob einer der Endzustände von einem der Anfangszustände erreichbar ist. Dazu können wir eine Tiefen- oder Breitensuche anwenden, wahlweise “vorwärts” (d.h., mit den Anfangszuständen beginnend nach einem erreichbaren Zustand  $p \in F$  suchend entlang der Kanten, die durch die Übergangsfunktion induziert werden) oder “rückwärts” (d.h., mit den Endzuständen beginnend und nach einem Anfangszustand suchend entlang der Kanten  $(p, q)$ , falls  $p \in \bigcup_{a \in \Sigma} \delta(q, a)$ ). In beiden Fällen sind die Kosten für den Leerheitstest linear in  $size(\mathcal{M})$ .

**Universalität.** Dual zu dem Leerheitsproblem ist die Frage nach der Universalität eines endlichen Automaten. Hiermit ist die Frage “gilt  $\mathcal{L}(\mathcal{M}) = \Sigma^*$ ?” gemeint, wobei  $\Sigma$  für das Alphabet von  $\mathcal{M}$  steht. Offenbar gilt  $\mathcal{L}(\mathcal{M}) = \Sigma^*$  genau dann, wenn  $\mathcal{L}(\overline{\mathcal{M}}) = \emptyset$ . Daher kann der Universalitätstest für DFA durch Komplementierung der Endzustandsmenge (und eventuell vorangegangene Totalisierung) auf das Leerheitsproblem zurückgeführt werden.

**Inklusions- und Äquivalenztest.** Die Frage, ob zwei DFA  $\mathcal{M}_1$  und  $\mathcal{M}_2$  äquivalent sind, d.h. “gilt  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$ ?” lässt sich auf das Inklusionsproblem reduzieren, da:

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

Für den Inklusionstest können wir folgende Beobachtung ausnutzen:

$$\begin{aligned} \mathcal{L}(\mathcal{M}_1) \subseteq \mathcal{L}(\mathcal{M}_2) \quad \text{gdw} \quad \mathcal{L}(\mathcal{M}_1) \cap \overline{\mathcal{L}(\mathcal{M}_2)} &= \emptyset \\ \text{gdw} \quad \mathcal{L}(\mathcal{M}_1 \otimes \overline{\mathcal{M}_2}) &= \emptyset \end{aligned}$$

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 dem Komplementautomaten von  $\mathcal{M}_2$  und führen für diesen den Leerheitstest durch. Die Laufzeit für den Inklusions- und Äquivalenztest für gegebene DFA  $\mathcal{M}_1, \mathcal{M}_2$  ist linear in der Größe der Produkt-DFA  $\mathcal{M}_1 \otimes \overline{\mathcal{M}_2}$  bzw.  $\overline{\mathcal{M}_1} \otimes \mathcal{M}_2$ , und kann daher durch  $\Theta(size(\mathcal{M}_1) \cdot size(\mathcal{M}_2))$  angegeben werden. Hierbei steht  $size(\mathcal{M})$  für die Anzahl an Zuständen und Transitionen in  $\mathcal{M}$ . Für gegebene NFA kann der Inklusions- und Äquivalenztest mit der beschriebenen Methode und vorangegangener Determinisierung gelöst werden. Die Kosten werden dann jedoch durch die Determinisierung dominiert und sind im schlimmsten Fall exponentiell.

**Endlichkeitstest.** Das Endlichkeitsproblem für endliche Automaten fragt, ob die akzeptierte Sprache endlich ist. Das Endlichkeitsproblem kann für NFA dank folgender Beobachtung gelöst werden. Die durch einen NFA  $\mathcal{M}$  akzeptierte Sprache ist genau dann *unendlich*, wenn es einen Anfangszustand  $q_0$ , einen Endzustand  $p$  und einen Zustand  $r$  gibt, so dass  $r$  von  $q_0$  und  $p$  von  $r$  erreichbar ist und  $r$  auf einem Zyklus liegt. Diese Bedingung kann durch den Einsatz von Graphalgorithmen, welche hier nicht erläutert werden, in linearer Zeit gelöst werden.