Prof. Dr. André Schulz

Kurs 01659

Grundlagen der Theoretischen Informatik

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Kurseinheit 2

Reguläre Sprachen I: Endliche Automaten und Reguläre Ausdrücke

In dieser Kurseinheit lernen wir mit den endlichen Automaten ein erstes Berechnungsmodell kennen. Dieses Modell orientiert sich am Rechnen mit begrenztem Speicher. Wir werden sehen, dass wir nur sehr einfache Probleme mit einem endlichen Automaten lösen können. Trotzdem ist der endliche Automat ein sehr wichtiges Modell, denn er bildet die Grundlage für weitere Modelle und findet als Modellierungswerkzug in der gesamten Informatik vielfältige Anwendungsmöglichkeiten.

2.1 Der Deterministische Endliche Automat

Wir beginnen mit einer informellen Beschreibung des Modells des endlichen Automaten. Genauer gesagt, handelt es sich hierbei um das Modell des *deterministischen* endlichen Automaten, den wir kurz DEA nennen (oder auch nur Automat, wenn keine Gefahr der Verwechslung zu anderen Modellen besteht). Mit einem DEA können wir das Wortproblem von bestimmten formalen Sprachen lösen (in der letzten Kurseinheit wurde besprochen, dass alle Entscheidungsprobleme als Wortprobleme formuliert werden können). In diesem Sinne verarbeitet der Automat ein Wort (die Eingabe) und gibt uns dann die Antwort, ob das Wort aus der zugehörigen Sprache ist, oder nicht. Wir sagen in diesem Zusammenhang auch, dass der DEA das Eingabewort **akzeptiert** (wenn er das Wort der Sprache zuordnet) oder **verwirft** (wenn er das Wort als nicht zur Sprache gehörig einsortiert). Um zu einer Antwort zu gelangen, *liest* der DEA das Anfragewort zeichenweise ein. Dabei kann er immer nur auf ein Zeichen der Eingabe zugreifen. Es ist ihm zudem nicht erlaubt, bereits gelesene Zeichen der Eingabe wieder anzufragen. Ein DEA hat nur beschränkten (konstanten) Speicher. Das heißt, während der Verarbeitung kann der DEA einen von endlich vielen *Zuständen* annehmen. Die eigentliche Berechnung wird dadurch festgelegt, wie man von einem Zustand in einen anderen Zustand gelangt. Dieser *Zustandsübergang* hängt vom aktuellen Zeichen der Eingabe ab. Am Ende, nachdem das letzte Zeichen der Eingabe gelesen wurde, können wir entscheiden, ob das Anfragewort aus der Sprache des DEAs ist. Diese Entscheidung wird vom Zustand abhängen, in dem der Automat sich am Ende

befindet.

Wir führen nun eine formale Definition des mathematischen Modells des deterministischen endlichen Automaten ein.

Definition 2.1 — Deterministischer Endlicher Automat.
Ein **deterministischer endlicher Automat (DEA)** M wird durch ein Tupel $(Q, \Sigma, \delta, q_0, F)$ dargestellt. Hierbei ist

- Q eine endliche nicht-leere Menge, genannt **Zustandsmenge**,
- Σ ein (endliches) Alphabet,
- δ eine Funktion $\delta\colon Q \times \Sigma \to Q$, genannt **Übergangsfunktion**,
- q_0 ein Element aus Q, genannt **Startzustand**,
- F eine Teilmenge von Q, genannt Menge der **akzeptierenden Zustände**.

Beispiel 2.1 Das folgende Quintupel $M_1 = (Q, \Sigma, \delta, q_0, F)$ gibt einen DEA an. Wir setzen hierbei $Q = \{q_0, q_1\}$, $\Sigma = \{\mathsf{a}, \mathsf{b}\}$ und $F = \{q_0\}$. Die Übergangsfunktion δ geben wir durch eine Tabelle an.

$q \in Q$	$x \in \Sigma$	$\delta(q, x)$
q_0	a	q_1
q_0	b	q_0
q_1	a	q_0
q_1	b	q_1

An dieser Stelle soll darauf hingewiesen werden, dass wir zwischen dem *Modell* des deterministischen endlichen Automaten und konkreten *Realisierungen* in diesem Modell, wie etwa in Beispiel 2.2 angegeben, unterscheiden. Es hat sich aber eingebürgert sowohl das Modell, als auch die Realisierungen, beides als deterministischen endlichen Automaten zu bezeichnen. Die Bedeutung ergibt sich aus dem Kontext. Trotzdem sollten Sie sich dieser Unterscheidung bewusst sein. Gleiches gilt auch für andere Modelle, die wir noch später im Kurs vorstellen werden (Kellerautomat, kontextfreie Grammatik, Turingmaschine).

Häufig werden wir eine graphische Notation namens **Zustandsdiagramm** benutzen, um einen DEA anzugeben. Aus dieser Darstellung lassen sich alle Bestandteile des Automaten leicht ablesen. Zustände werden wir als Kreise darstellen (in Ausnahmefällen als Rechtecke), die mit dem Zustand (in der Mitte) beschriftet sind. Akzeptierende Zustände heben wir zusätzlich hervor, indem wir deren Kreise mit einer doppelten Linie zeichnen. Falls $\delta(q, x) = p$, vermerken wir das, indem wir einen Pfeil einfügen, der den Zustand q mit dem Zustand p verbindet (Pfeil zeigt in Richtung p). Diesen Pfeil beschriften wir zusätzlich mit x. Es verbleibt, den Startzustand zu kennzeichnen. Dies realisieren wir, indem wir einen kleinen Pfeil an diesen Zustand anbringen. Der Pfeil zeigt auf den Startzustand, und sein Anfangspunkt ist mit keinem Zustand verbunden. Die Abbildung 2.1 zeigt noch einmal die Grundelemente der graphischen Darstellung. Das Zustandsdiagramm des DEAs aus dem Bespiel 2.1 ist in Abbildung 2.2 zu sehen.

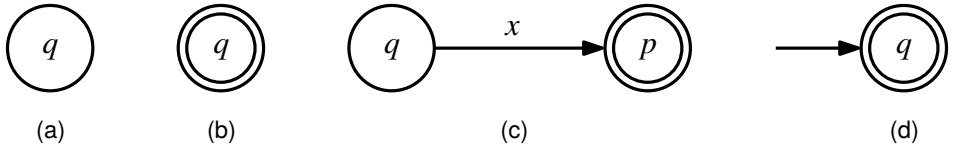

Abbildung 2.1: Bestandteile der graphischen Notation eines DEAs: (a) verwerfender Zustand, (b) akzeptierender Zustand, (c) Zustandsübergang, (d) Startzustand.

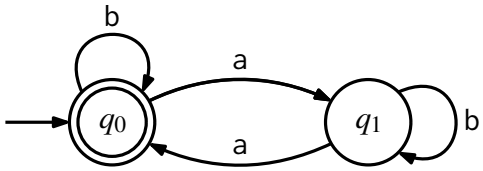

Abbildung 2.2: Zustandsdiagramm des DEAs M_1 aus Beispiel 2.1.

Die Übergangsfunktion ist das „Herzstück“ des DEAs. Es handelt sich hierbei um eine Funktion, deren Definitionsbereich Paare bestehend aus einem Zustand und einem Zeichen sind. Sie gibt also für einen Zustand und ein Zeichen einen neuen Zustand an. Diesen Zustand bezeichnen wir als **Folgezustand**. Wie bereits beschreiben, befindet sich der Automat während der Berechnung immer in einem Zustand. Zu Beginn der Berechnung ist dies der Startzustand. Während der Berechnung liest er die Eingabe Zeichen für Zeichen und gleicht seinen Zustand ab. Dazu nutzt er die Übergangsfunktion δ. Wenn q den aktuellen Zustand bezeichnet und x das nächste Zeichen der Eingabe ist, dann gibt $\delta(q, x)$ den Folgezustand an. Nachdem alle Zeichen der Eingabe gelesen wurden, befindet sich der DEA in einem Zustand. Ist dieser Zustand ein akzeptierender Zustand, wird das Eingabewort akzeptiert, ansonsten verworfen. Die Folge der Zustände, die der Automat während der Berechnung angenommen hat, bezeichnen wir als seinen **Lauf** für die gewählte Eingabe. Wir sprechen auch von einem w-Lauf, wenn der Lauf sich auf die Eingabe w bezieht. Ein Lauf ist eine **akzeptierender Lauf**, wenn er in einem akzeptierenden Zustand endet, ansonsten nennen wir ihn **verwerfenden Lauf**. Alle Zustände, die man im Zustandsdiagramm (als gerichteter Graph interpretiert) vom Startzustand erreichen kann, nennen wir **erreichbare Zustände**. Die *nicht-erreichbaren* Zustände spielen für die Akzeptanz eines Wortes keine Rolle und können immer entfernt werden.

Die Menge aller Wörter, die der Automat akzeptiert, nennen wir die **Sprache des Automaten** oder auch die vom Automaten akzeptierte Sprache. Die Sprachen eines Automaten M notieren wir mit $L(M)$.

In Abbildung 2.3 ist ein Berechnungsablauf des Automaten M_1 aus Beispiel 2.1 exemplarisch für das Wort aba dargestellt. Man kann sich für dieses Beispiel recht leicht überlegen, welche Wörter von diesem DEA akzeptiert werden. Wir erkennen, dass es zwei Zustände q_0 und q_1 gibt, von welchen nur q_0 akzeptierend ist. Wenn ein Zeichen b von der Eingabe gelesen wird, ist der Folgezustand gleich dem ursprünglichen Zustand. Deshalb hängt es nur von den Zeichen a ab, ob ein Wort akzeptiert wird. Genauer gesagt, hängt es von der Anzahl der Zeichen a ab, da a das einzig relevante Zeichen ist. Wenn ein Zeichen a gelesen wird, wird der Zustand gewechselt, und zwar von akzeptierend zu nicht-akzeptierend, oder umgekehrt. Das bedeutet, dass es von der Parität (gerade/ungerade) der Anzahl der Zeichen

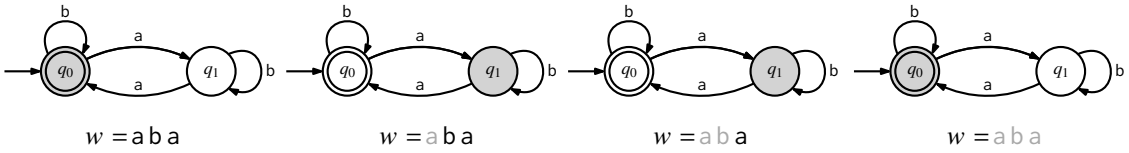

Abbildung 2.3: Ablauf der Berechnung von M_1 für die Eingabe $w = \mathsf{aba}$. Der aktuelle Zustand ist grau hinterlegt. Bereits gelesene Zeichen der Eingabe sind ebenfalls grau. Der Lauf des Automaten ist (q_0, q_1, q_1, q_0). Da der Lauf in einem akzeptierenden Zustand endet, wird die Eingabe aba akzeptiert.

a abhängt, ob die Eingabe akzeptiert wird. Wir sehen also, dass für dieses Beispiel

$$L(M_1) = \{w \in \{\mathsf{a}, \mathsf{b}\}^* \mid w \text{ enthält gerade Anzahl von as}\}$$

gilt.

Bevor wir den *Akzeptanzbegriff* des DEAs formal beschreiben, werden wir noch eine hilfreiche Notation einführen. Die Übergangsfunktion δ erlaubt uns den Folgezustand zu bestimmen, wenn wir ein Zeichen von der Eingabe gelesen haben. Oft ist es aber nützlich, den „Folgezustand“ zu beschreiben, wenn man statt eines Zeichens ein Wort liest. Dafür nutzen wir die **iterierte Übergangsfunktion** δ^*, welche direkt aus δ abgeleitet werden kann.

Definition 2.2 — Iterierte Übergangsfunktion eines DEAs.
Sei δ die Übergangsfunktion eines DEAs, dann definieren wir für alle $q \in Q$

$$\delta^0(q, \varepsilon) = q,$$

und für alle $i > 0$ und alle Wörter $w = ua \in \Sigma^i$ mit $u \in \Sigma^{i-1}$ und $a \in \Sigma$

$$\delta^i(q, w) = \delta(\delta^{i-1}(q, u), a).$$

Schließlich definieren wir die **iterierte Übergangsfunktion** $\delta^* \colon Q \times \Sigma^* \to Q$ als

$$\delta^*(q, w) := \delta^{|w|}(q, w).$$

Für den Automaten M_1 aus Beispiel 2.1 ergibt sich beispielsweise $\delta^*(q_0, \mathsf{aba}) = q_0$ und $\delta^*(q_1, \mathsf{aa}) = q_1$. Mit der iterierten Übergangsfunktion können wir nun kompakt die von einem DEA erkannte Sprache definieren.

Definition 2.3 — Sprache eines DEAs.
Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA, dann ist die von M akzeptierte Sprache definiert als

$$L(M) := \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}.$$

Die Sprachen die von einem DEA erkannt werden, haben viele nützliche Eigenschaften und bilden eine interessante Struktur. Aus diesem Grunde geben wir dieser Sprachfamilie einen Namen.

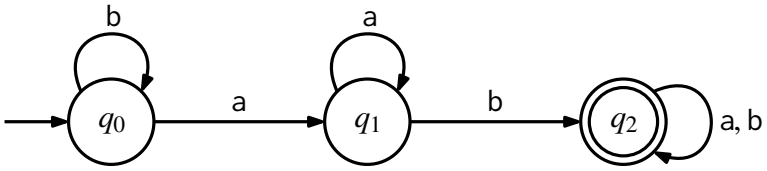

Abbildung 2.4: Der DEA M_2 zu Beispiel 2.2.

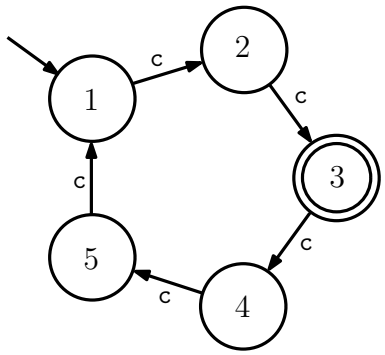

Abbildung 2.5: Der DEA M_3 zu Beispiel 2.3.

Definition 2.4 — Reguläre Sprache.
Wenn L eine Sprache ist, für die es einen DEA gibt, der L akzeptiert, nennen wir L eine **reguläre Sprache**. Wir nutzen die Bezeichnung

$$REG := \{L \mid L \text{ ist regulär}\}.$$

An dieser Stelle wollen wir noch zwei Beispiele besprechen.

Beispiel 2.2 Der DEA M_2 ist durch das Zustandsdiagramm in Abbildung 2.4 gegeben. Wir erkennen, dass es mit q_2 nur einen akzeptierenden Zustand gibt. Wenn q_2 während der Berechnung angenommen wird, verbleibt der DEA in diesem Zustand. Um nach q_2 zu gelangen, müssen wir vorher in q_1 sein, und das nächste zu lesende Zeichen muss ein b sein. Man befindet sich aber genau dann in q_1 (ohne vorher schon in q_2 zu sein), wenn als letztes Zeichen ein a gelesen wurde. Also akzeptiert M_2 alle Wörter, die als Teilwort ab erhalten. Das heißt

$$L(M_2) = \{w \in \{\mathsf{ab}\}^* \mid \mathsf{ab} \text{ ist Teilwort von } w\}.$$

Das Beispiel 2.2 gibt die erste praktische Anwendung für unser Berechnungsmodell. Die meisten Beispiele für reguläre Sprachen wirken sehr künstlich. Eigentlich gehen wir ja davon aus, dass es sich bei diesen Sprachen um Kodierungen der Ja-Instanzen von Entscheidungsproblemen handelt. Die Sprache $L(M_2)$ ist in dieser Beziehung interessant. Das zugrundeliegende Entscheidungsproblem fragt, ob ein Wort das Teilwort ab enthält. Es ist nicht schwer, den DEA umzuwandeln, sodass wir nach anderen Teilwörtern fragen können. Die Frage, ob ein Text ein Teilwort enthält, hat eine hohe praktische Relevanz (*pattern matching*). Viele Algorithmen zum Suchen von Wörtern in Texten benutzen endliche Automaten als Hilfsmittel. Zum Beispiel nutzt das Kommandozeilenprogramm *grep* einen solchen Ansatz.

Beispiel 2.3 Sei $M_3 = (\{1,2,3,4,5\}, \{\mathsf{c}\}, \delta, 1, \{3\})$ mit

$$\delta(x, \mathsf{c}) = \begin{cases} x+1 & \text{falls } x \neq 5 \\ 1 & \text{sonst.} \end{cases}$$

Das Zustandsdiagramm des Automaten ist in Abbildung 2.5 zu sehen. Wir erkennen, dass wir immer genau dann im Zustand 1 sind, wenn wir eine Anzahl von cs gelesen haben, die ein Vielfaches von 5 ist. Demnach akzeptiert M_3 genau die Wörter w mit $|w| \bmod 5 = 2$. Somit gilt

$$L(M_3) = \{\mathsf{c}^k \mid 5 \text{ teilt } k \text{ mit Rest } 2\}.$$

Test 2.1

Entwerfen Sie einen DEA, der die folgende Sprache akzeptiert:

$$L = \{w \in \{\mathtt{0},\mathtt{1}\}^+ \mid w \text{ hat unterschiedliches Anfangs- und Endzeichen}\}$$

Wir wollen nun einen ersten Satz zu den regulären Sprachen beweisen. Hierbei geht es um die Beziehung zu einer anderen Sprachklasse — den endlichen Sprachen. Wir nennen eine Sprache **endlich**, wenn sie nur endlich viele Wörter besitzt.

Satz 2.1
Jede endliche Sprache ist regulär.

Beweis. Sei $L \subseteq \Sigma^*$ eine endliche Sprache deren längstes Wort die Länge ℓ hat. Um zu zeigen, dass L regulär ist, müssen wir einen DEA M für L angeben. Wir nehmen vorerst an, dass $\Sigma = \{\mathsf{a}, \mathsf{b}\}$. Wir beschreiben M, indem wir sein Zustandsdiagramm angeben. In der Grundstruktur entspricht das Diagramm einem binären Baum der Tiefe ℓ. Von einem inneren Knoten gibt es zwei Kanten zu seinen Kindern. Eine dieser Kanten beschriften wir mit a und die andere mit b. Wir orientieren nun alle Kanten vom Vater zum Kind. Als Startzustand wählen wir die Wurzel des Baumes. Es gibt für jeden Knoten genau einen Pfad von der Wurzel. Wir benennen einen Zustand mit q_w, wenn w das Wort ist, was man lesen muss, um ihn zu erreichen. Nun machen wir genau die Zustände q_w zu akzeptierenden Zuständen, für die w ein Wort aus der Sprache L ist. Abschließend führen wir noch einen Müllzustand q_- ein (nicht-akzeptierend). Alle Übergänge die nun noch fehlen, gehen zum Müllzustand über. Abbildung 2.6 zeigt diese Konstruktion am Beispiel.

Es ist nun nicht schwer zu argumentieren, dass $M(L) = L$. Jedes Wort der Länge größer ℓ führt nach q_- und wird verworfen. Jedes andere Wort w führt zum Zustand q_w. Ist $w \in L$, dann ist $q_w \in F$ und wir akzeptieren w. Alle anderen Wörter werden verworfen.

Bei anderen Alphabeten erfolgt die Konstruktion analog. Statt eines binären Baumes nutzt man einen k-ären Baum, wobei $k = |\Sigma|$. ■

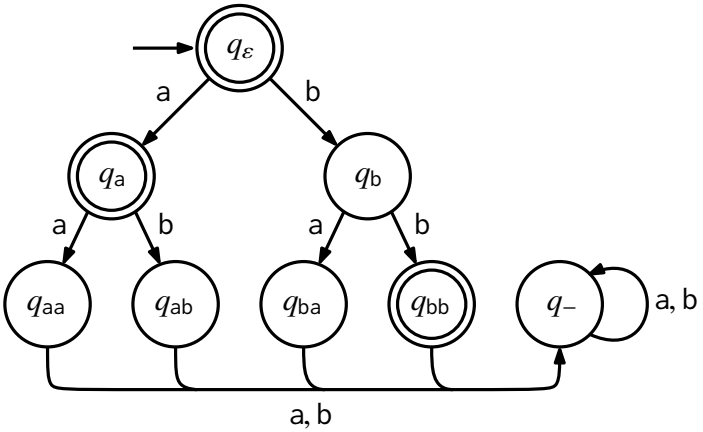

Abbildung 2.6: Konstruktion zum Beweis von Satz 2.1 für die Sprache $\{\varepsilon, \mathsf{a}, \mathsf{bb}\}$.

2.2 Nichtdeterministische Endliche Automaten

Als nächstes werden wir ein neues Berechnungsmodell einführen, welches sich an der Arbeitsweise von deterministischen endlichen Automaten anlehnt. Dies ist der sogenannte **nichtdeterministische endliche Automat (NEA)**. Wie auch der DEA arbeitet der NEA mit Zuständen, welche akzeptierend oder verwerfend sein können. Auch der NEA verarbeitet das Eingabewort zeichenweise und kann nicht auf bereits gelesene Zeichen direkt wieder zurückgreifen. Genau wie beim DEA gibt es auch eine Übergangsfunktion – diese weist jedoch jedem Zustand und Eingabezeichen nicht einen einzelnen Folgezustand zu, sondern eine Menge von Folgezuständen. In diesem Sinne gibt es nicht nur einen Lauf für jede Eingabe, sondern mitunter mehrere mögliche Läufe.

Es stellt sich natürlich die Frage, wie man damit umgeht, dass der mögliche Folgezustand nicht mehr eindeutig festgelegt ist. So könnte es durchaus sein, das bei ein und demselben Eingabewort ein Lauf in einem akzeptierenden Zustand endet, ein anderer Lauf aber in einem verwerfenden Zustand. Es ist also nicht offensichtlich, wie der Akzeptanzbegriff für NEAs gefasst ist. Unser Kriterium für die Akzeptanz eines Wortes wird anschaulich das folgende sein: **Existiert ein Lauf** vom Startzustand zu einem akzeptierenden Zustand, wird das Eingabewort akzeptiert. Dieser Akzeptanzbegriff scheint auf den ersten Blick künstlich, denn diese Art von Berechnung widerspricht unserem intuitiven Verständnis vom maschinellen Berechnen. Es wird sich jedoch zeigen, dass das Berechnungsmodell NEA seine Berechtigung hat. Durch die Nutzung des Nichtdeterminismus lassen sich viele Probleme leichter modellieren. Zusätzlich können wir durch die Verwendung von NEAs gegenüber von DEAs viele Beweise von Sätzen über reguläre Sprachen vereinfachen.

Es gibt noch einen weiteren Unterschied zwischen NEA und DEA. Bei einem DEA kann ein Zustandswechsel nur dann geschehen, wenn ein Zeichen von der Eingabe gelesen wurde. Wir erlauben beim NEA auch, den Zustand zu wechseln ohne dabei ein Zeichen zu lesen. Diese Übergänge sollen natürlich nicht beliebig stattfinden. Deshalb definieren wir sogenannte **ε-Übergänge** zwischen Zuständen. Ist ein ε-Übergang zwischen Zustand p und q vorhanden, kann man vom Zustand p in den Zustand q wechseln, ohne ein Zeichen der Eingabe zu lesen. Im Zustandsdiagramm werden solche Übergänge wie normale Übergänge eingezeichnet, statt eines Zeichen aus dem Alphabet werden sie jedoch mit ε beschriftet.

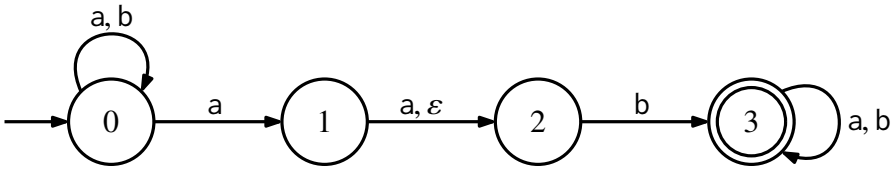

Abbildung 2.7: Zustandsdiagramm vom NEA N_1.

Bevor wir die NEAs formal definieren, erklären wir die prinzipielle Arbeitsweise eines NEAs am Beispiel. Sehen wir uns das Zustandsdiagramm von NEA N_1 in Abbildung 2.7 an. Wir erkennen an folgenden Merkmalen, dass es sich um das Diagramm eines NEAs handelt. Es gibt nicht immer genau einen möglichen Folgezustand. Zum Beispiel ist es möglich, vom Zustand 0 mit einem a sowohl zum Zustand 1 zu gelangen, als auch im Zustand 0 zu bleiben. Des Weiteren können wir beobachten, dass es vom Zustand 1 keinen möglichen Folgezustand gibt, den man mit einem b erreichen kann. Die Menge der Folgezustände kann also auch die leere Menge sein. Außerdem erkennen wir, dass der Automat einen ε-Übergang zwischen Zustand 1 und 2 aufweist. In diesem Automaten können wir vom Zustand 0 zum Zustand 3 gelangen, indem wir aab lesen. In diesem Sinne gibt es einen akzeptierenden Lauf für das Wort aab. Man kann aber auch erkennen, dass man mit demselben Wort auch einen Lauf realisieren kann, der die ganze Zeit im Zustand 0 verweilt. Ein anderes Wort mit einem akzeptierenden Lauf ist das Wort ab. Hier können wir von Zustand 0 zu Zustand 1 wechseln, indem wir ein a lesen, dann nutzen wir den ε-Übergang, um in den Zustand 3 zu gelangen, und anschließend können wir durch das Lesen des Zeichen b in den akzeptierenden Zustand 3 wechseln.

Wir werden nun das Modell NEA und den damit verbundenen Akzeptanzbegriff formal definieren. An dieser Stelle sei noch einmal daran erinnert, dass man mit der Potenzmenge $\mathcal{P}(X)$ die Menge aller Teilmengen von X bezeichnet, also $\mathcal{P}(X) := \{Y \subseteq X\}$.

Definition 2.5 — Nichtdeterministischer Endlicher Automat.
Ein **nichtdeterministischer endlicher Automat (NEA)** M wird durch ein Tupel $(Q, \Sigma, \delta, q_0, F)$ dargestellt. Hierbei ist

- Q eine endliche nicht-leere Menge, genannt **Zustandsmenge**,
- Σ ein (endliches) Alphabet,
- δ eine Funktion $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$, genannt **Übergangsfunktion**,
- q_0 ein Element aus Q, genannt **Startzustand**,
- F eine Teilmenge von Q, genannt Menge der **akzeptierenden Zustände**.

Als nächstes werden wir die iterierte Übergangsfunktion eines NEAs aus seiner Übergangsfunktion ableiten. Die iterierte Übergangsfunktion soll uns angeben, in welchem Zustand man nach dem Lesen eines Wortes *sein könnte*. Für die Einbeziehung der ε-Übergänge benötigen wir noch eine Definition. Wir wollen ausdrücken können, welche Zustände wir von p aus erreichen können, ohne ein Zeichen zu lesen. Diese Zustandsmenge notieren wir

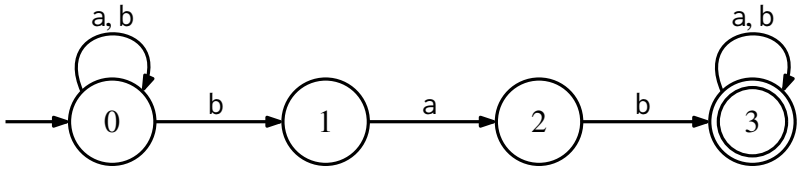

Abbildung 2.8: NEA N_2 für Beispiel 2.4.

mit $E(p)$. Es gilt also

$$E(p) := \{q \mid q \text{ ist von } p \text{ durch eine Sequenz von } \geq 0 \ \varepsilon\text{-Übergängen erreichbar}\}.$$

Für Mengen $P \subseteq Q$ definieren wir

$$E(P) := \bigcup_{p \in P} E(p).$$

Für den NEA N_1 aus Abbildung 2.7 gilt beispielsweise $E(\{0, 1\}) = \{0, 1, 2\}$.

Definition 2.6 — Iterierte Übergangsfunktion eines NEAs.
Sei δ die Übergangsfunktion eines NEAs, dann definieren wir für alle $P \subseteq Q$

$$\delta^0(P, \varepsilon) = E(P),$$

und für alle $i > 0$ und alle Wörter $w = ua \in \Sigma^i$ mit $u \in \Sigma^{i-1}$ und $a \in \Sigma$

$$\delta^i(P, w) = E(\textstyle\bigcup_{r \in \delta^{i-1}(P,u)} \delta(r, a)).$$

Schließlich definieren wir die **iterierte Übergangsfunktion** $\delta^* \colon Q \times \Sigma^* \to \mathcal{P}(Q)$ als

$$\delta^*(P, w) := \delta^{|w|}(P, w).$$

Analog zum DEA können wir die iterierte Übergangsfunktion benutzen, um die Akzeptanz eines Wortes und damit die Sprache eines NEAs zu definieren. Es sollen genau die Worte akzeptiert werden, für die es *möglich ist*, vom Startzustand durch Übergänge zu einem akzeptierenden Zustand zu gelangen.

Definition 2.7 — Sprache eines NEAs.
Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NEA, dann ist die von N akzeptierte Sprache definiert als

$$L(N) := \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}.$$

Beispiel 2.4 Als ein weiteres Beispiel sehen wir uns den in Abbildung 2.8 gezeigten NEA N_2 an. Um in den akzeptieren Zustand 3 zu kommen, muss man vorher das Teilwort bab gelesen haben. Das heißt, es können nur Wörter akzeptiert werden, die bab als Teilwort enthalten. Auf der anderen Seite gibt es für jedes Wort, welches bab als Teilwort enthält, einen akzeptierenden Lauf. Dieser verbleibt im Zustand 0 bis das Teilwort bab beginnt, dann liest er dieses Teilwort und geht dabei in den Zustand 3. Anschließend

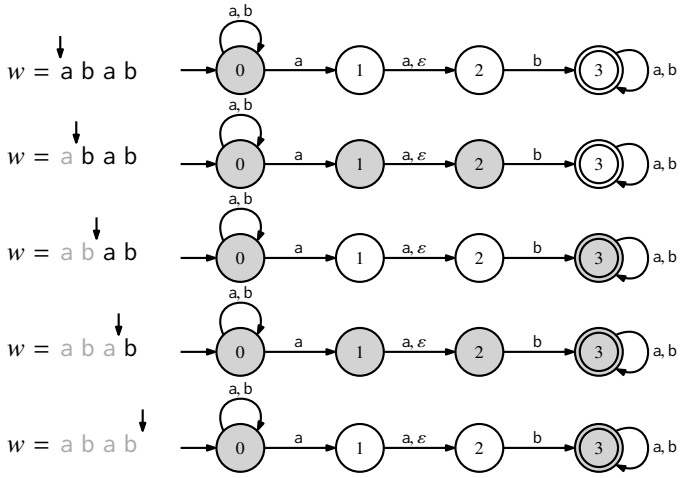

Abbildung 2.9: Mögliche Zustände (grau) des NEA N_1 aus Abbildung 2.7 bei der Verarbeitung des Eingabewortes $w = \mathsf{abab}$.

verbleibt er im Zustand 3. Wir erhalten also

$$L(N_2) := \{w \in \{\mathsf{a}, \mathsf{b}\}^* \mid w \text{ enthält } \mathsf{bab} \text{ als Teilwort}\}.$$

Wir werden uns nun ansehen, wie man (praktisch) überprüfen kann, ob ein NEA ein Wort akzeptiert oder nicht akzeptiert. Sei also ein NEA und ein Wort w gegeben, zum Beispiel der NEA N_1 aus Abbildung 2.7 und das Wort $w = \mathsf{abab}$. Wir wollen herausfinden, in welchen Zuständen man nach dem Lesen von w sein kann, wenn man vom Startzugang ausgeht, und ob einer dieser möglichen Zustände ein akzeptierender Zustand ist (wir wollen also $\delta^*(E(q_0), w) \cap F \neq \emptyset$ auswerten). Dazu werden wir das Wort w Zeichen für Zeichen verarbeiten und uns immer alle möglichen aktuellen Zustände merken. Am Anfang (ohne ein Zeichen zu lesen) können wir nur im Zustand 0 sein. Nach dem Lesen des ersten Zeichens von w (ein a) können wir sowohl im Zustand 0, 1, oder 2 sein, denn hier kommt der Nichtdeterminismus zum Tragen. Nach dem Lesen des nächsten Zeichens b können wir uns im Zustand 0 befinden (von Zustand 0 aus kommend) oder im Zustand 3 (von Zustand 2 aus kommend). Die möglichen Zustände sind also $\{0, 3\}$. Wenn wir das nächste Zeichen a lesen, kommen wir in die Zustände 0,1,2 (von Zustand 0 aus kommend) oder wir verbleiben im Zustand 3. Die Menge der möglichen Zustände ist somit $\{0, 1, 2, 3\}$. Das letzte zu lesende Zeichen ist ein b. Danach können wir uns im Zustand 0 (vom Zustand 0 aus kommend) oder im Zustand 3 (vom Zustand 2 oder 3 aus kommend) befinden. Also sind die möglichen Zustände nach dem Lesen von w gleich $\{0, 3\}$. Da in dieser Menge mit Zustand 3 ein akzeptierender Zustand enthalten ist, akzeptieren wir w. Die möglichen Zustände für dieses Beispiel sind in Abbildung 2.9 dargestellt.

Als nächstes wollen wir die Frage diskutieren, ob ein NEA mehr Sprachen erkennen kann als ein DEA. Es ist klar, dass es für jede reguläre Sprache einen NEA gibt, der diese akzeptiert, denn jeder DEA ist ein NEA, der den Nichtdeterminismus und die ε-Übergänge nicht verwendet. Wir werden aber auch zeigen, dass jede Sprache die ein NEA akzeptiert, auch von einem DEA akzeptiert wird. Dazu werden wir eine Konstruktion vorstellen, die aus einem NEA einen DEA konstruiert, der die gleiche Sprache akzeptiert. Diesen DEA nennen wir **Potenzautomat**.