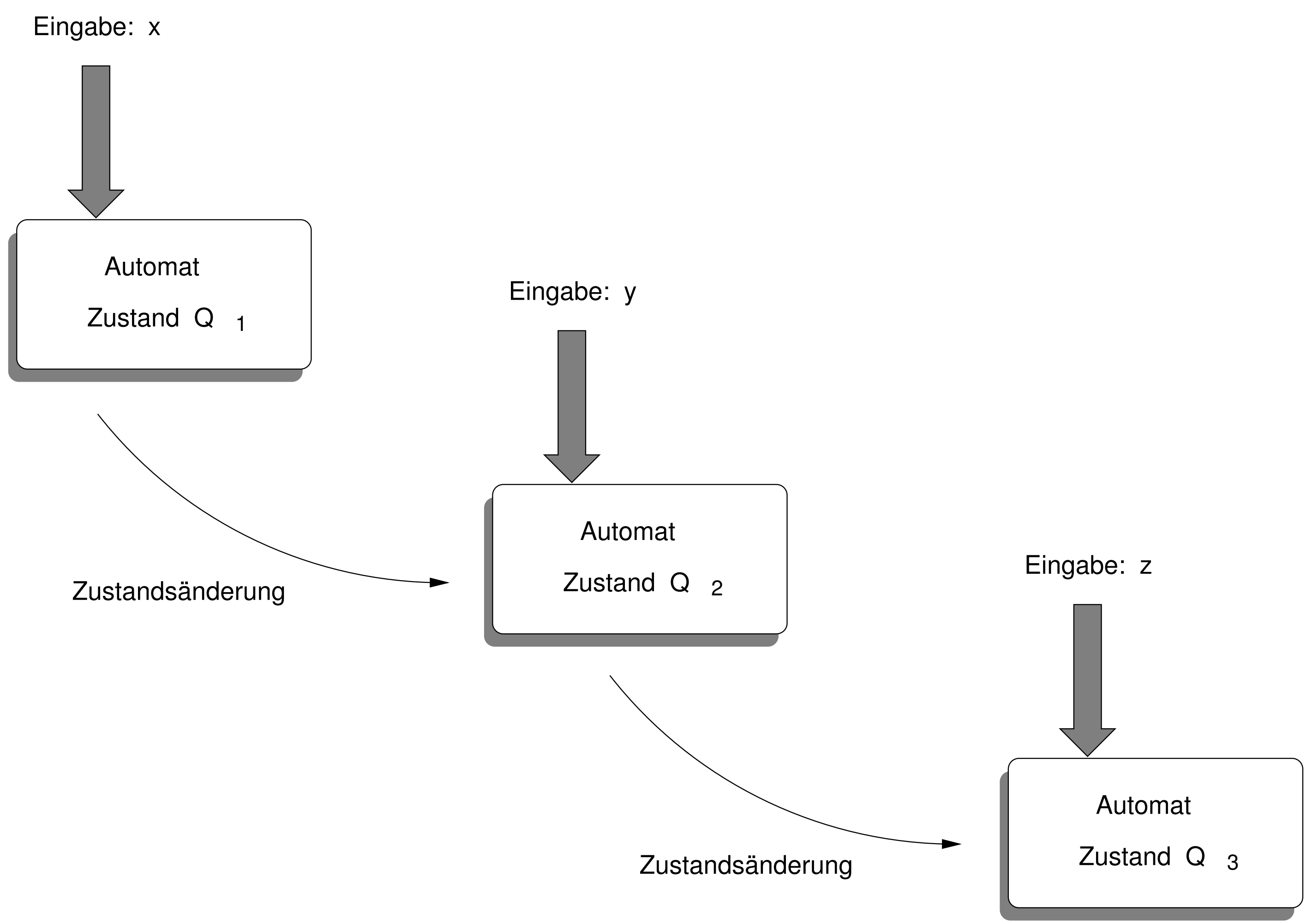
Compilerbau-Grundlagen: Endliche Automaten

Michael Jäger

17. April 2017

Automaten

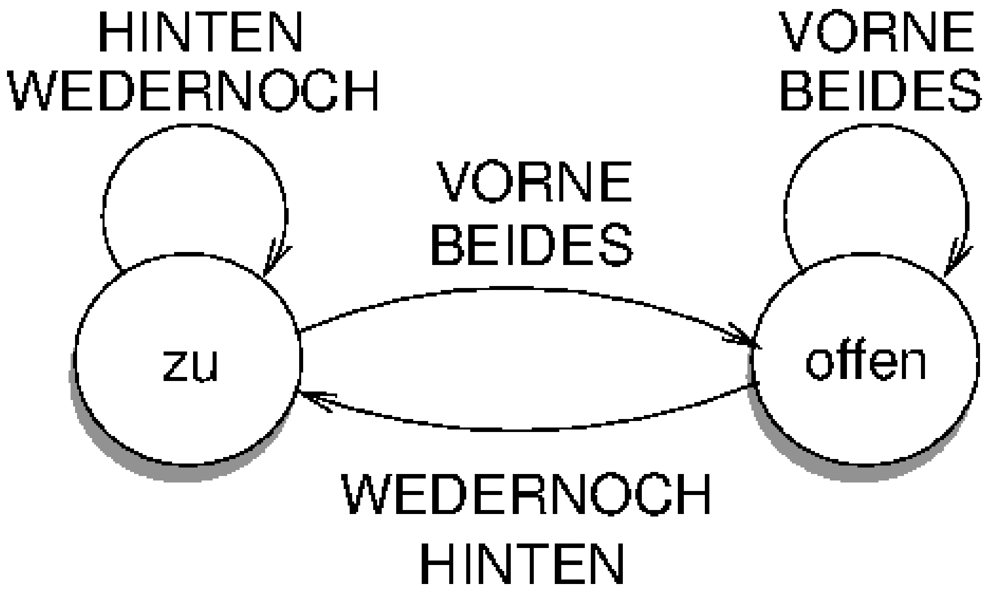

Eingabe: x
Automat
Zustand Q 1
Eingabe: y
Zustandsänderung
Automat
Zustand Q 2
Eingabe: z
Zustandsänderung
Automat
Zustand Q 3

Beispiel: Steuerung einer automatischen Tür

- 2 Detektoren: *vor* und *hinter* der Tür
- Jeder Detektor sendet binäre Information: Person vorhanden/nichts vorhanden
- Steuerung für den Durchgang in einer Richtung (von vorne nach hinten) programmiert
- 4 mögliche Eingabewerte für Controller (Paare von Sensorzuständen):

Bezeichnung	**vor Tür**	**hinter Tür**
WEDERNOCH	nichts	nichts
VORNE	Person	nichts
HINTEN	nichts	Person
BEIDES	Person	Person

Diagrammdarstellung einer Steuerung



Tabellendarstellung

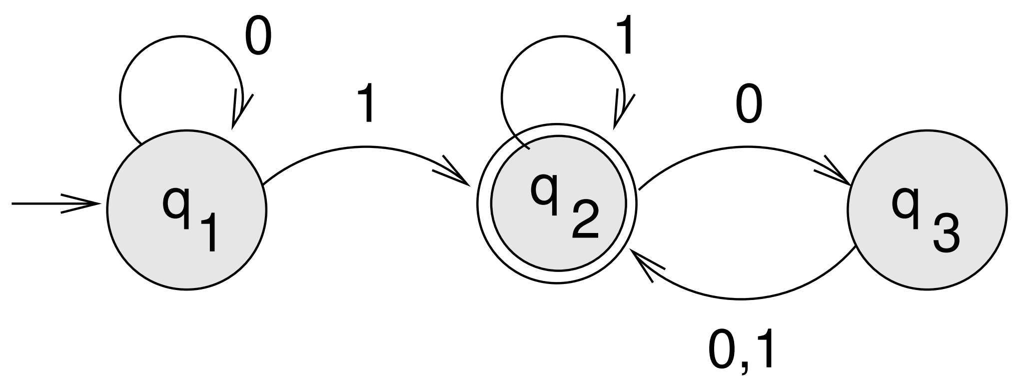
Eingabewert

		WEDERNOCH	VORNE	HINTEN	BEIDES
Ausgangszustand	ZU	ZU	OFFEN	ZU	OFFEN
	OFFEN	ZU	OFFEN	ZU	OFFEN

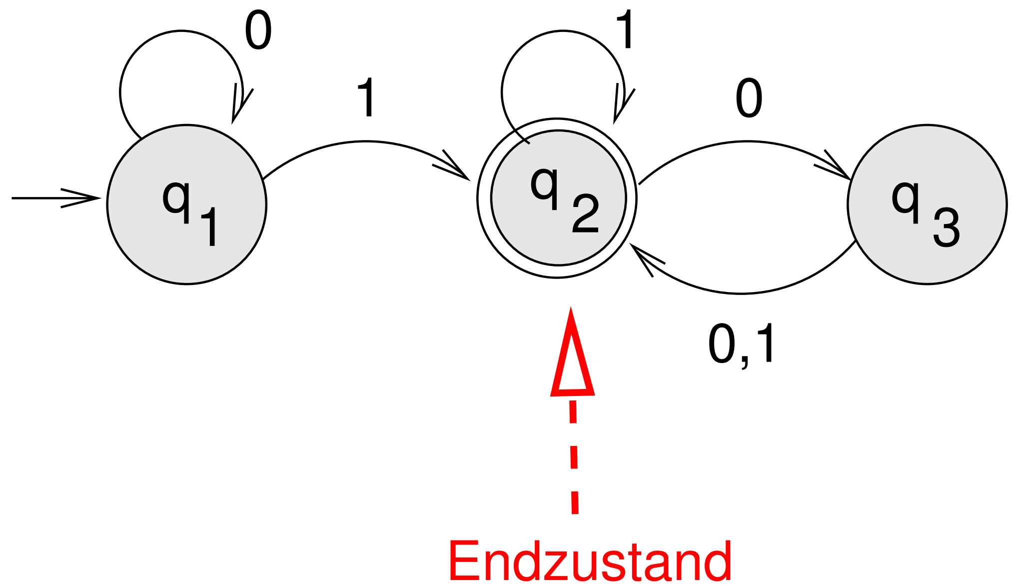
Eigenschaften der Steuerung

- Endlich viele Zustände (im Beispiel: 2)
- Zustandsübergang abhängig von Eingabewerten
- „Computer“ mit Speichergröße 1 Bit
- Steuerung für den Durchgang in einer Richtung (von vorne nach hinten) programmiert

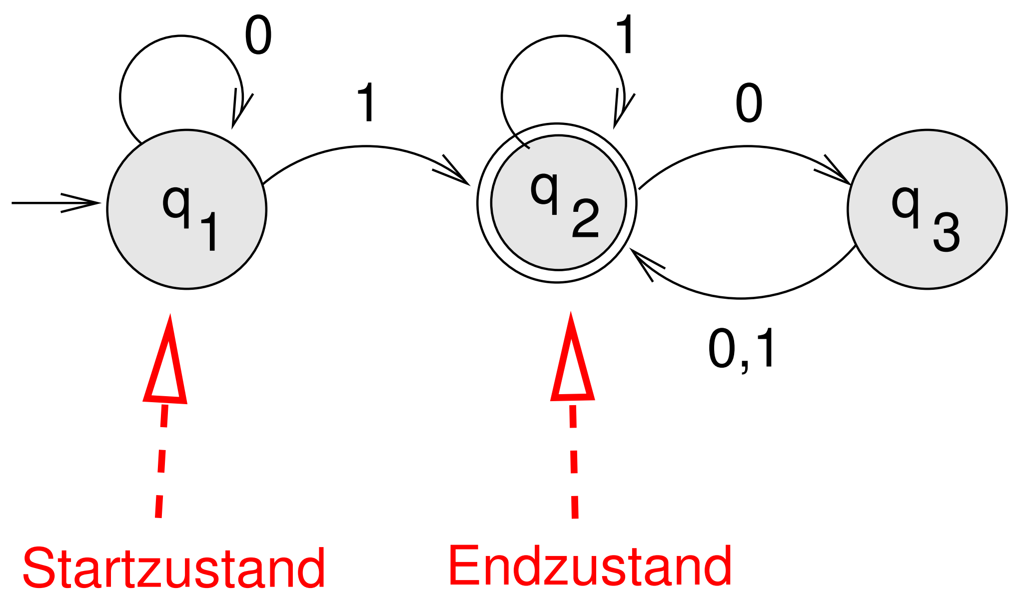
Beispiel für Zustandsübergangsdiagramm



Beispiel für Zustandsübergangsdiagramm



Beispiel für Zustandsübergangsdiagramm



Formale Definition eines Endlichen Automaten

Definition 1:

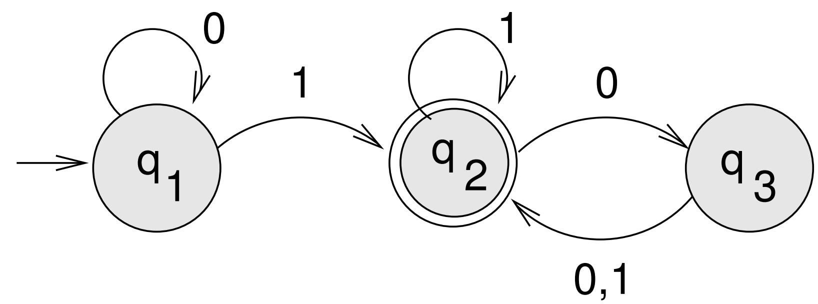
Ein Endlicher Automat (EA) ist ein 5-Tupel $(Q, \Sigma, \delta, q_0, F)$ mit folgenden Bestandteilen

1. Q ist die endliche Menge der Zustände
2. Σ ist ein Alphabet
3. $\delta : Q \times \Sigma \longrightarrow Q$ ist die Übergangsfunktion
4. $q_0 \in Q$ ist der Startzustand
5. $F \subseteq Q$ ist die Menge der Endzustände

Anmerkungen zur Definition

- Die Endzustandsmenge F kann durchaus leer sein
- δ ist *Funktion*, d.h. zu einem Zustand q und einem Eingabesymbol $a \in \Sigma$ gibt es *höchstens* einen Folgezustand q'
- Die Benennung der Zustände eines Automaten spielt keine Rolle!

Formale Definition zum Diagramm



$M = (Q, \Sigma, \delta, q_0, F)$ mit

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ als Wertetabelle:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

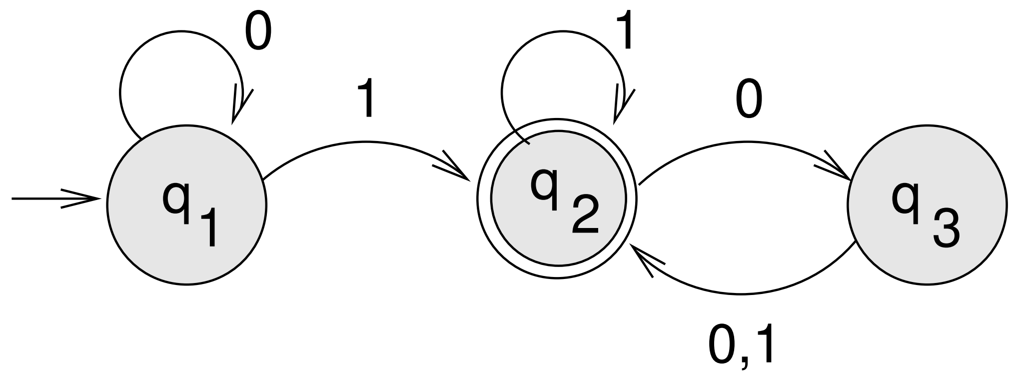
4. q_1 ist der Startzustand.
5. $F = \{q_2\}$ ist die Menge der Endzustände.

Automat und Sprache

Ein Endlicher Automat $M = (Q, \Sigma, \delta, q_0, F)$ definiert eine formale Sprache $L(M)$:

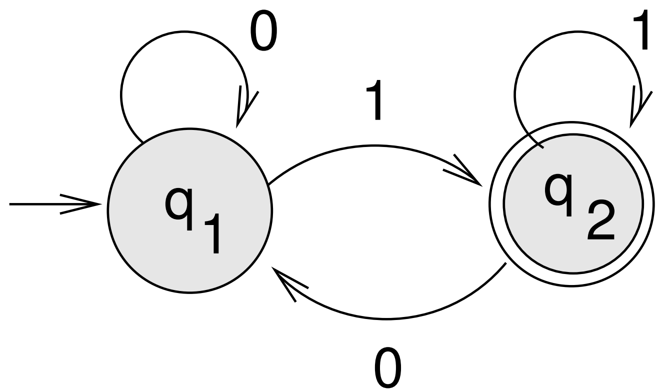
- M akzeptiert ein Wort $w = a_1, \ldots, a_n \in \Sigma^*$, g.d.w. M ausgehend von seinem Startzustand mit w als Eingabe in einen Endzustand gelangt.
- $L(M)$ ist die Menge aller von M akzeptierten Wörter.

Beispiel für akzeptierte Sprache



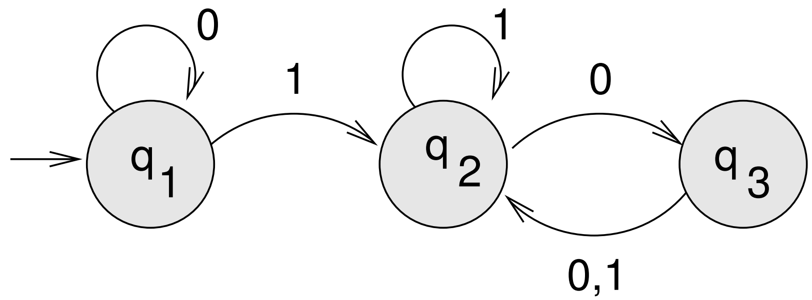
$L(M) = \{w \in \{0,1\}^* \mid w$ enthält mindestens eine 1 und hinter der letzten 1 folgt eine gerade Anzahl Nullen $\}$

EA-Beispiel 1.2



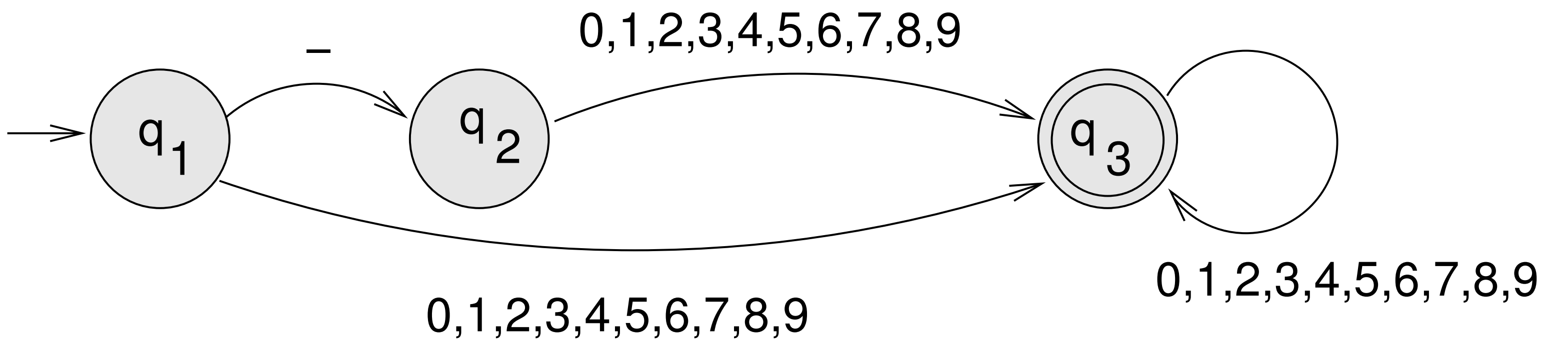
$L(M) = \{w \in \{0,1\}^* \mid w$ endet auf 1 $\}$

EA-Beispiel 1.3



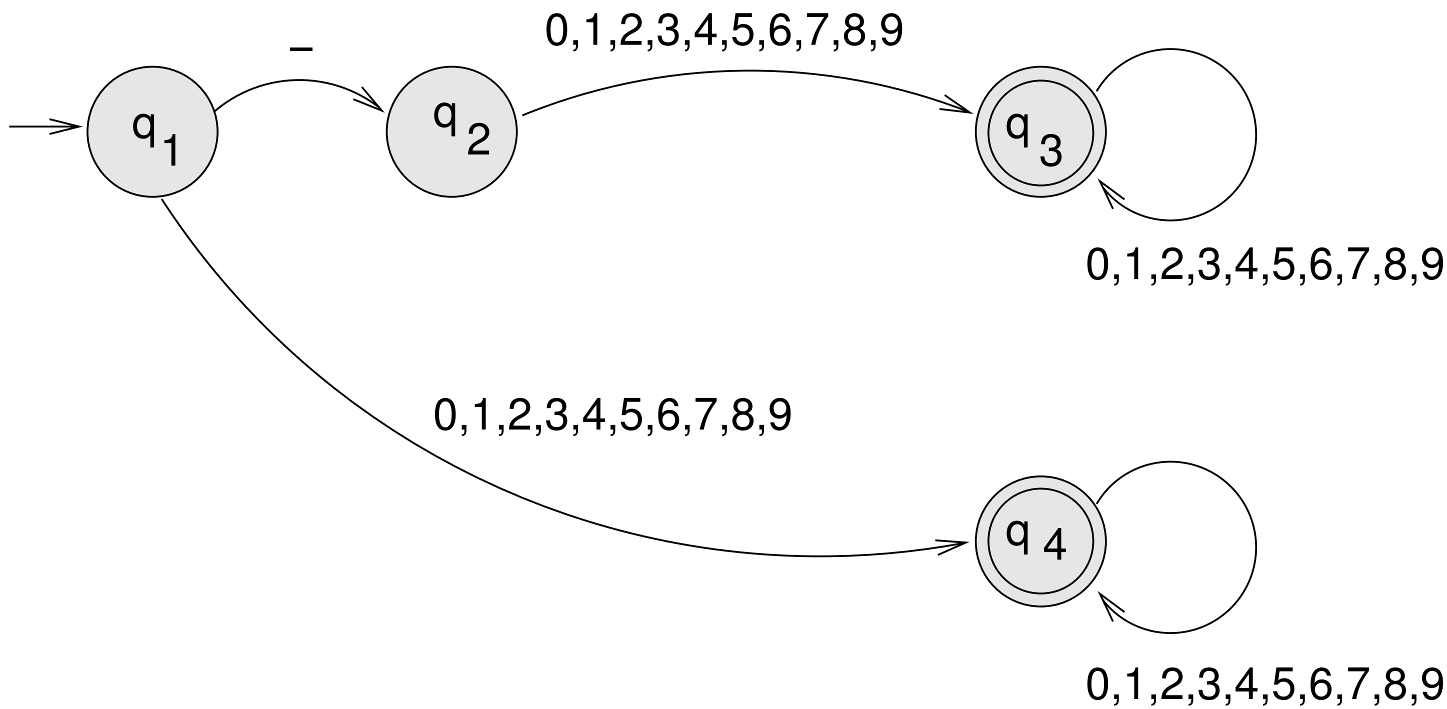
$L(M) = \emptyset$

EA-Beispiel 1.4



$L(M) =$Menge der Dezimalliterale mit optionalem negativen Vorzeichen

EA-Beispiel 1.5



$L(M) =$Menge der Dezimalliterale mit optionalem negativen Vorzeichen
Der Automat ist zu dem vorherigen äquivalent: Er akzeptiert dieselbe Sprache.

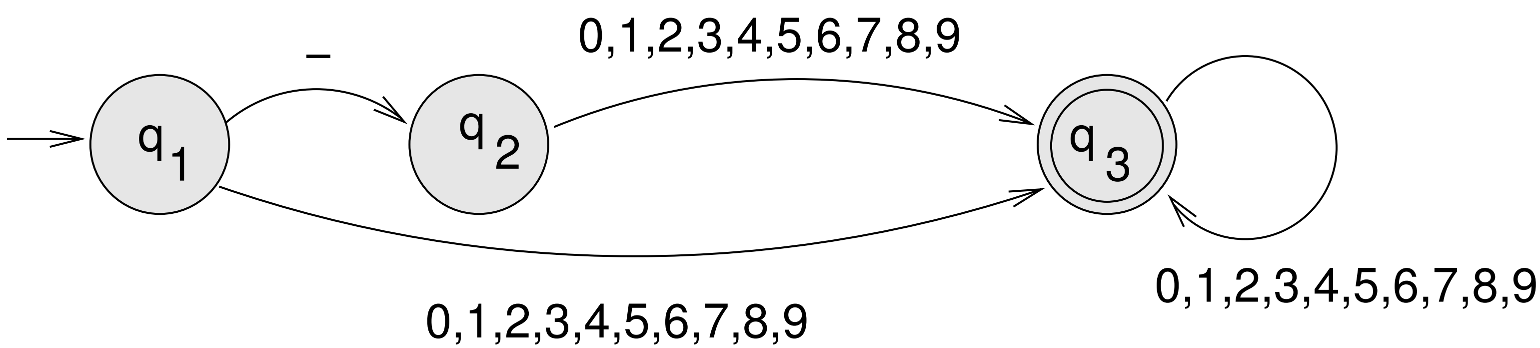
Automaten mit undefinierten Folgezuständen

- Strenggenommen definiert eine mathematische Funktion *für jedes Element* des Definitionsbereichs einen Funktionswert.

  Die Übergangsfunktion δ eines Automaten $M = (Q, \Sigma, \delta, q_0, F)$ müsste daher für jedes Paar $(q, a) \in Q \times \Sigma$ einen Folgezustand q' definieren.

- Für praktische Anwendungen ist es oft sinnvoller, für Symbole, deren Auftreten in einem bestimmten Zustand zur Ablehnung der Eingabe führt, gar keinen Folgezustand zu definieren.

  Im nachfolgenden Beispiel ist für $(q_2, -)$ und $(q_3, -)$ kein Folgezustand definiert:



- Eine Möglichkeit, dies korrekt zu formalisieren, ist das in der Informatik gebräuchliche Konzept der partiellen Funktionen: Eine partielle Funktion hat für einige Argumente des Definitionsbereichs keinen definierten Funktionswert.

  Beispiel aus der Mathematik:

  $f : \mathcal{R} \longrightarrow \mathcal{R}, f(x) = 1/x,$ f ist für $x = 0$ nicht definiert.

  Die vollständig definierten Funktionen werden demgegenüber auch als totale Funktionen bezeichnet.

- Wir lassen es zu, dass die Zustandsübergangsfunktion $\delta : Q \times \Sigma \longrightarrow Q$ eines Automaten M nur partiell definiert ist. Dies ändert nichts an der Definition der von M akzeptierten Sprache!

Vervollständigung von unvollständigen EA

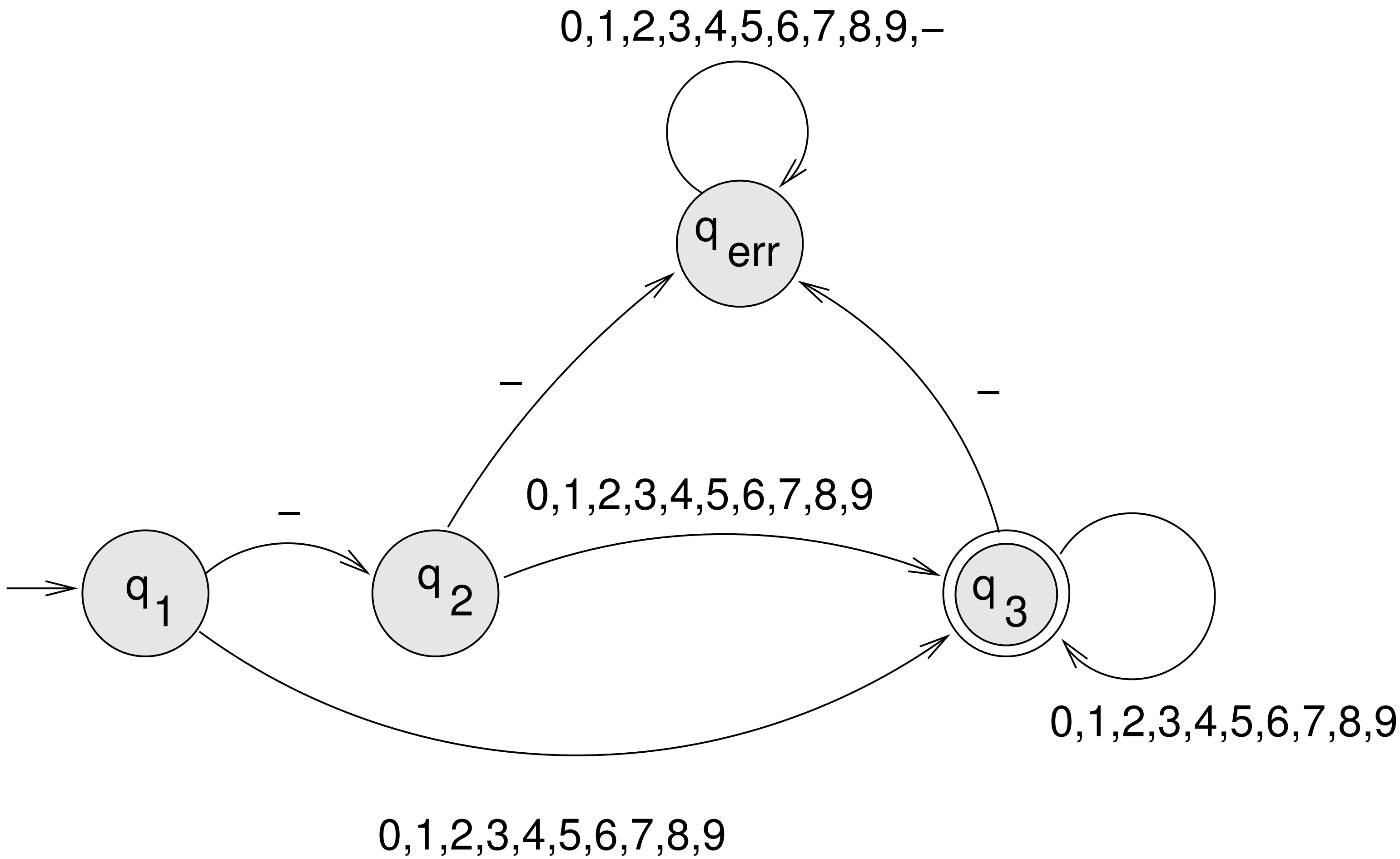
Zu jedem Endlichen Automaten $M = (Q, \Sigma, \delta, q_0, F)$ gibt es einen äquivalenten Endlichen Automaten $M' = (Q', \Sigma, \delta', q_0, F)$, bei dem $\delta' : Q' \times \Sigma \longrightarrow Q'$ eine totale Funktion ist.

Man führt einen neuen, nicht akzeptierenden **Fehlerzustand** q_{err} ein: $Q' = Q \cup \{q_{err}\}$.

Hat M für (q, a) keinen definierten Folgezustand, so geht M' in den Fehlerzustand über, in dem er für alle Eingaben verbleibt:

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{falls } q \in Q \text{ und } \delta(q,a) \text{ ist definiert} \\ q_{err} & \text{sonst} \end{cases}$$

Beispiel mit vervollständigter Übergangsfunktion



Formale Definition regulärer Sprachen

Seien $M = (Q, \Sigma, \delta, q_0, F)$ ein EA, $w = a_1 a_2 \dots a_n$ ein Wort über Σ.

- M *akzeptiert* w, wenn es Zustände $r_0, r_1, \dots, r_n$ aus Q gibt, so dass
  1. $r_0 = q_0$,
  2. $\delta(r_i, a_{i+1}) = r_{i+1}$ für $i = 0, \dots, n-1$ und
  3. $r_n \in F$

- M akzeptiert eine Sprache L, wenn $L = \{w \mid M \text{ akzeptiert } w\}$

- Eine Sprache heißt **regulääre Sprache**, wenn ein EA sie akzeptiert.

Nichtdeterminismus

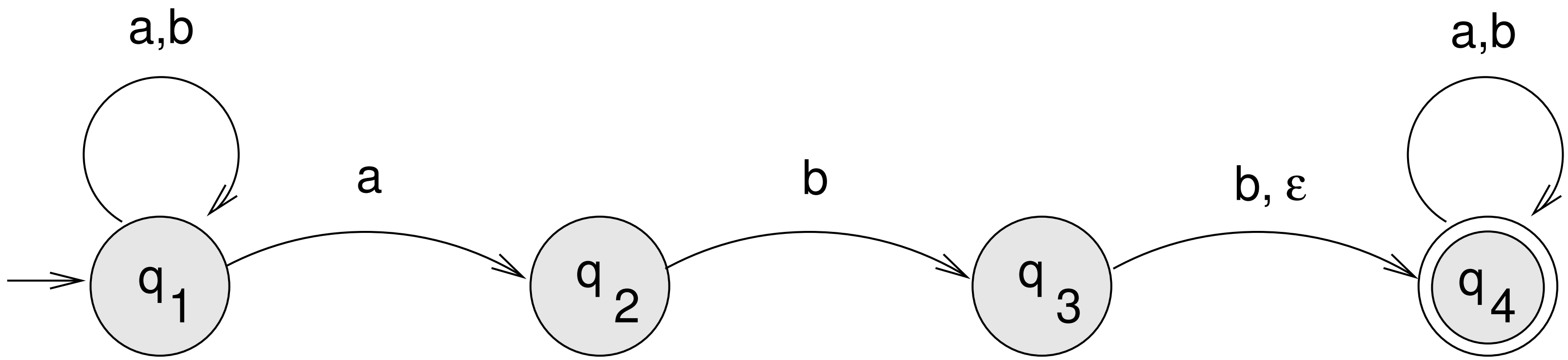
- Bei einem deterministischen Automaten ist in jedem Zustand der Folgezustand zu einem Eingabesymbol eindeutig bestimmt, die Zustandübergänge werden durch eine *Funktion* $\delta : Q \times \Sigma \longrightarrow Q$ beschrieben.

- Nichtdeterminismus ist eine Verallgemeinerung dieses Konzepts, bei der ein Automat mehrere Auswahlmöglichkeiten haben kann:

  a) Bei einem nichtdeterministischen Automaten kann es zu einem Zustand und einem Eingabesymbol mehrere mögliche Folgezustände geben.
  b) Bei einem nichtdeterministischen Automaten kann es Zustandsübergange geben, die unabhängig von der Eingabe sind. Bei diesen sogenannten ε-Übergängen wechselt der Automat den Zustand, ohne das nächste Eingabesymbol zu lesen.

- Für eine bestimmte Eingabe w kann es daher bei einem nichtdeterministischen Automaten viele unterschiedliche Berechnungen (Zustandsfolgen) geben. Manche davon mögen in einen Endzustand führen, andere nicht.

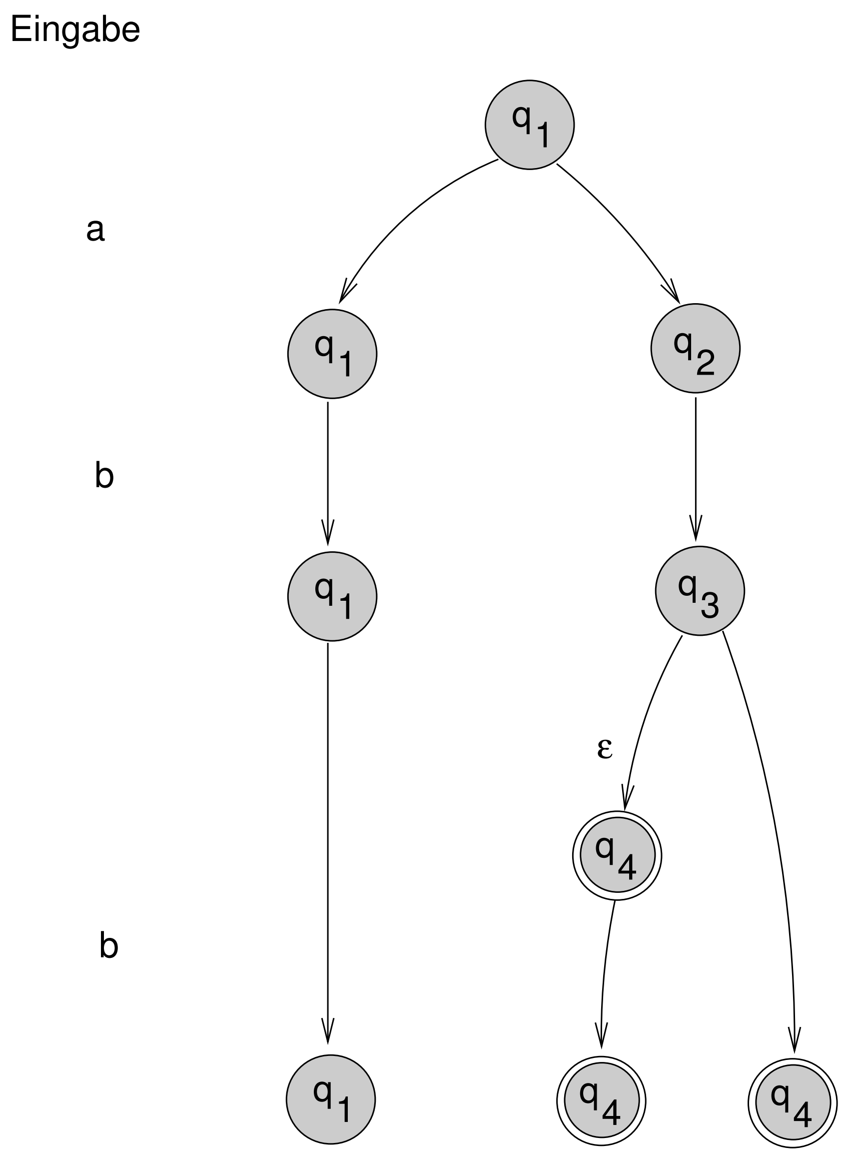
Beispiel für einen Nichtdeterministischen Endlichen Automaten



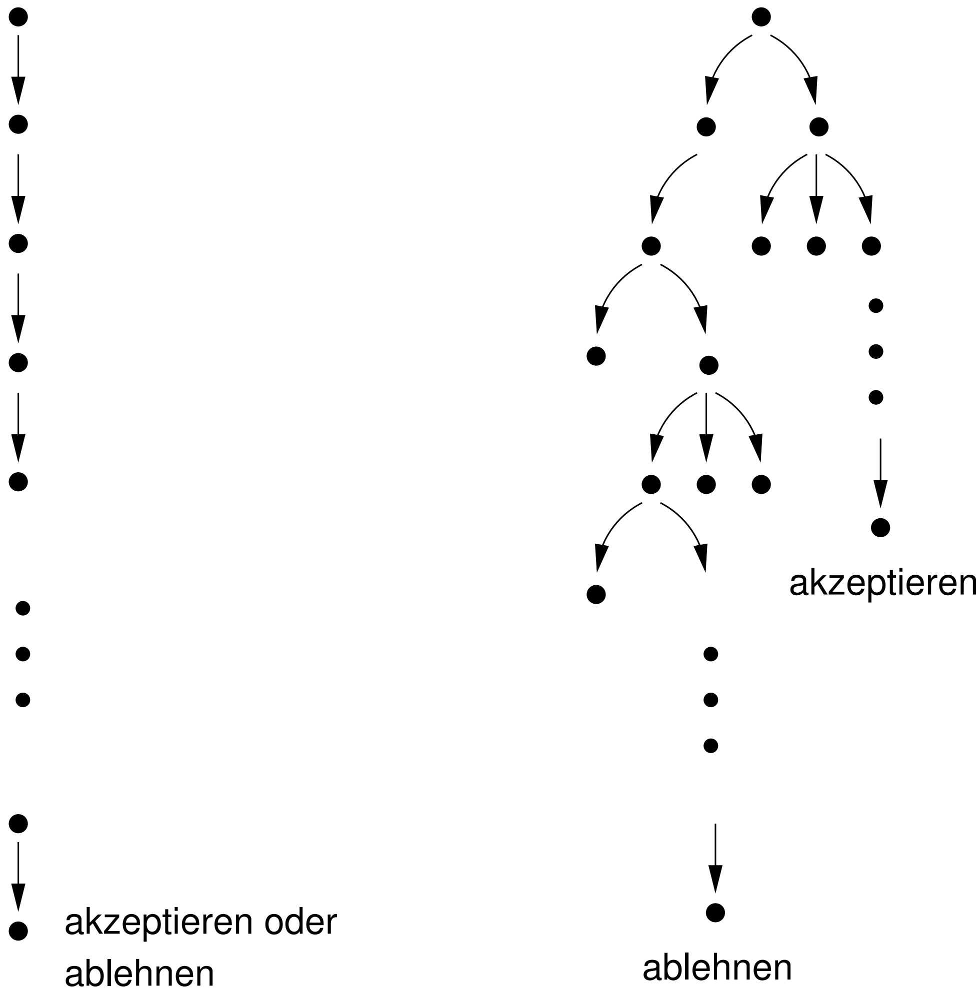
Berechnungen für Eingabewort abb:

$$
\begin{array}{ll}
(1) & q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \\
(2) & q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3 \xrightarrow{\varepsilon} q_4 \xrightarrow{b} q_4 \\
(3) & q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3 \xrightarrow{b} q_4
\end{array}
$$

Entscheidungsbaum



Deterministische / nichtdeterministische Berechnungen



Formale Definition eines Nichtdeterministischen Endlichen Automaten

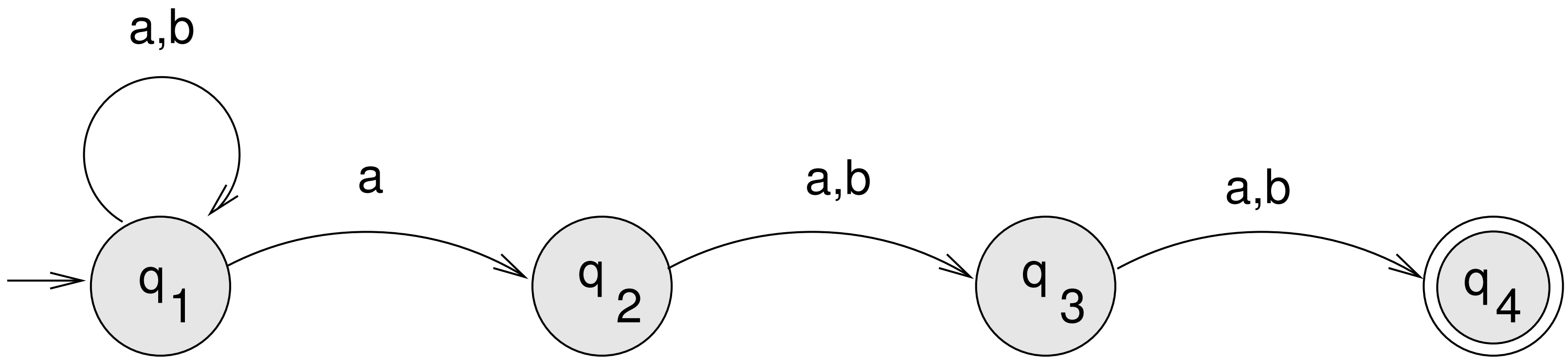
Definition 2:

Ein Nichtdeterministischer Endlicher Automat (NEA) ist ein 5-Tupel $(Q, \Sigma, \delta, q_0, F)$ mit folgenden Bestandteilen

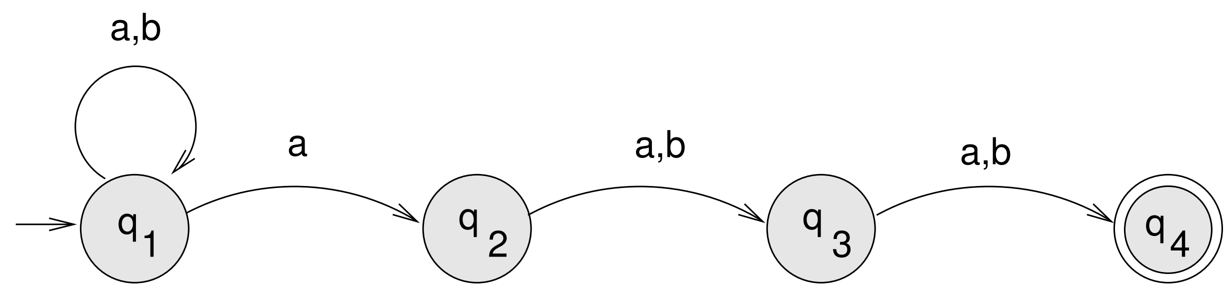
1. Q ist die endliche Menge der Zustände
2. Σ ist ein Alphabet
3. $\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ ist die Übergangsfunktion
4. $q_0 \in Q$ ist der Startzustand
5. $F \subseteq Q$ ist die Menge der Endzustände

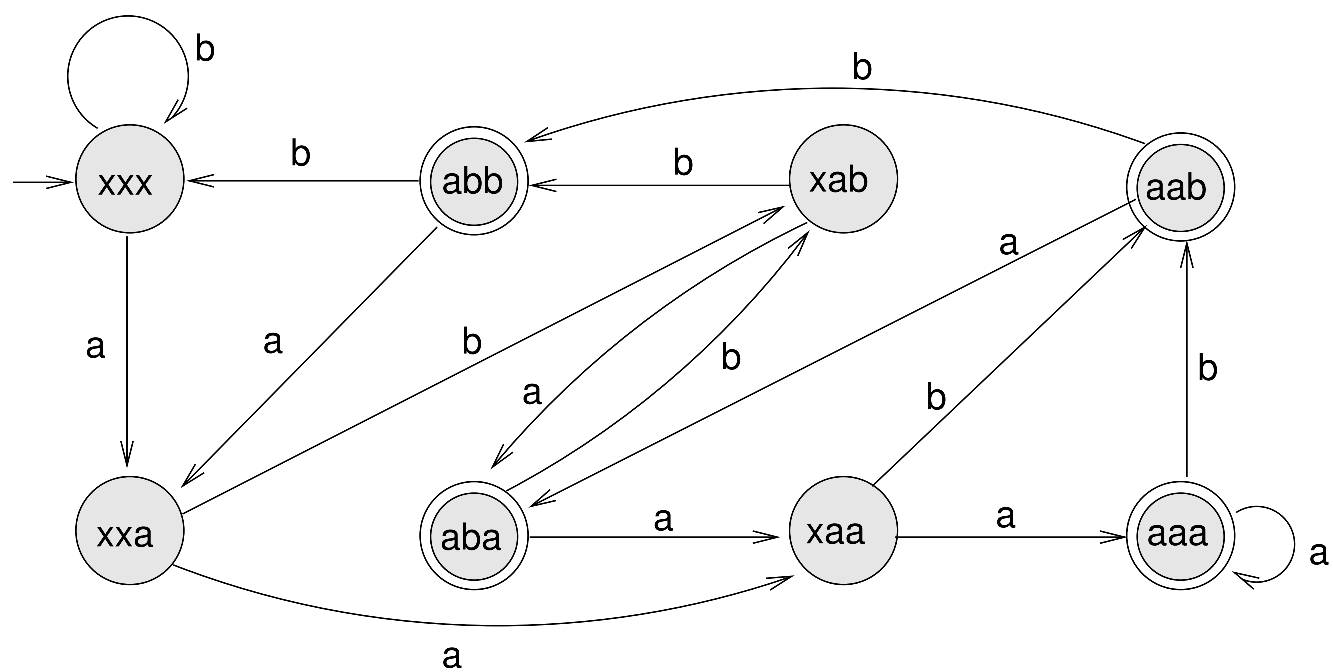
Da der Nichtdeterminismus das DEA-Konzept verallgemeinert, ist jeder DEA auch ein NEA.

NEA-Beispiel 2

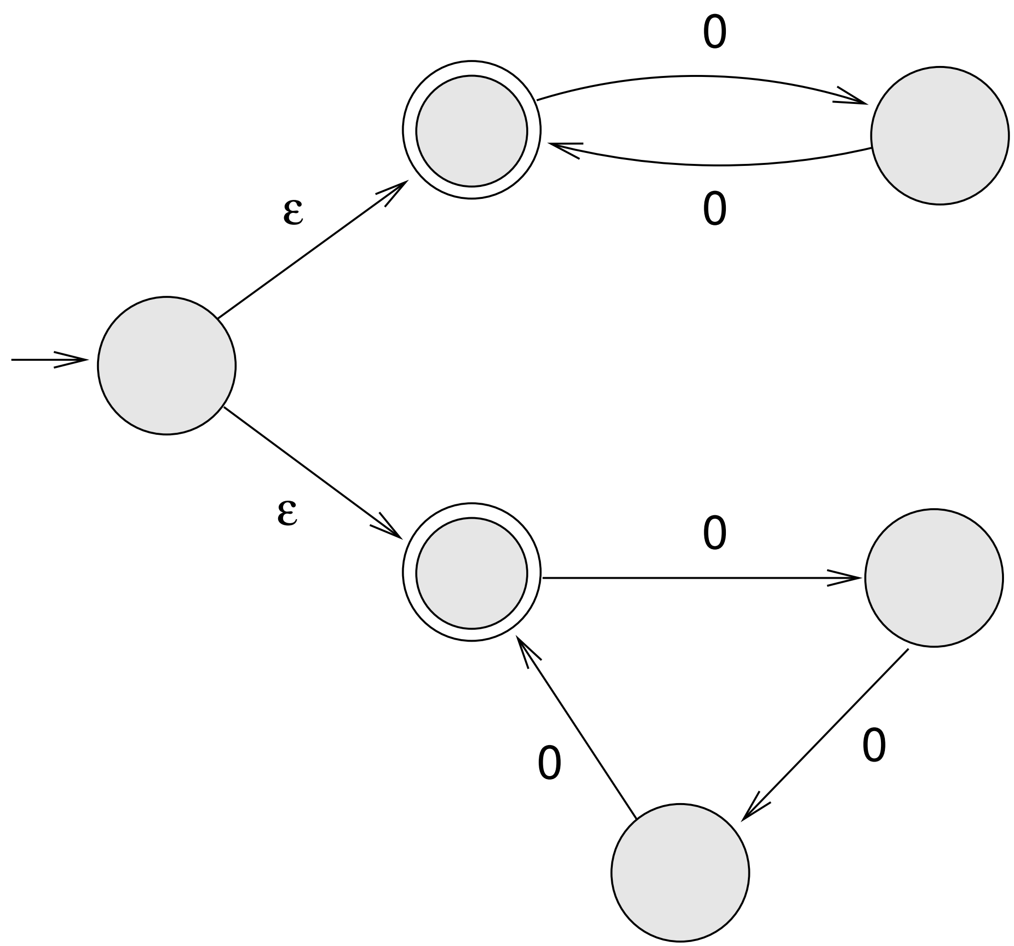


Gegenüberstellung: NEA und äquivalenter DEA





NEA-Beispiel 3



NEA-Beispiel 4