Theoretische Informatik

Teil 3
Endliche Automaten

Frühlingssemester 2019

L. Di Caro
D. Flumini
O. Stern

Lerninhalte

- Berechnungsmodell der endlichen Automaten (EA)
- Grenzen der EA: Nichtexistenzbeweise
- „Erweiterung“ der EA: Nichtdeterministische EA (NEA)
- Äquivalenz von regulären Ausdrücken und EA
- Abschlusseigenschaften regulärer Sprachen

Lernziele

- Die Studierenden wissen, wie ein Berechnungsmodell aufgebaut wird.
- Sie kennen verschiedene Darstellungsarten endlicher Automaten (EA) und können diese sinnvoll einsetzen.
- Sie sind mit den Definitionen deterministischer EA (DEA) und nichtdeterministischer EA (NEA) vertraut.
- Für gegebene Sprachen können Sie DEAs (NEAs) bauen oder die Sprache gegebener DEAs (NEAs) bestimmen.
- Sie können die Situation (Konfiguration) in einem DEA (NEA) vollständig beschreiben.
- Sie können eine Berechnung von DEAs (NEAs) formal darstellen und wissen, wie die Sprache endlicher Automaten definiert ist.
- Sie können die Zustandsklassen eines DEAs bestimmen.

Lernziele

- Sie können durch einen Widerspruchsbeweis untere Schranken für die Grösse von DEAs für eine bestimmte Sprache bestimmen.
- Sie können die Nichtregularität von gegebenen Sprachen beweisen.
- Sie sind mit dem Konzept des Nichtdeterminismus vertraut.
- Sie kennen die Vorteile von NEAs.
- Sie können die Teilmengenkonstruktion durchführen, um die Äquivalenz von DEAs und NEAs zu zeigen.
- Sie wissen, dass DEAs und reguläre Ausdrücke gleichmächtig sind.
- Sie kennen einige Abschlusseigenschaften regulärer Sprachen und können ähnliche Eigenschaften beweisen.
- Sie nutzen den ε-NEA für einen einfacheren Automatenentwurf.
- Sie wissen, dass ε-NEAs äquivalent zu NEAs sind.

Endliche Automaten

Motivation:

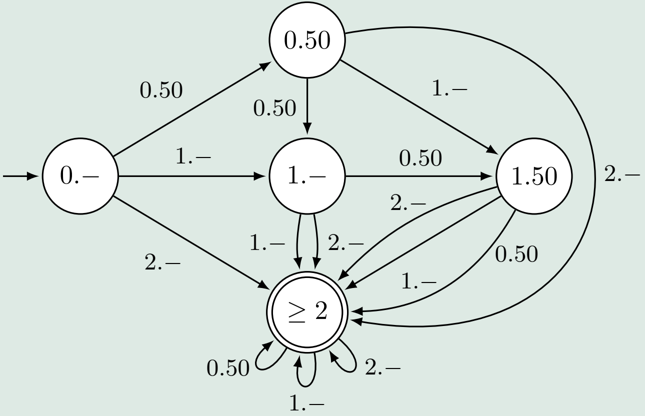
- Einfachstes Berechnungsmodell.
- Modellierung von Berechnungen ist anschaulich.
- Fundamentale Konzepte der Informatik werden zum ersten Mal angesprochen.
- Endliche Automaten (EA) sind allgegenwärtig.

Fakten:

- EA entsprechen speziellen Maschinen, die konkrete Entscheidungsprobleme lösen und dabei keine Variablen benutzen.
- EA arbeiten in Echtzeit.

Beispiel (Einstiegsaufgabe: Eintrittskarte Schwimmbad)

Kosten 2.- (mindestens), Automat akzeptiert 0.50, 1.- und 2.-



Das Modell der endlichen Automaten

Fragen:

- Welche elementaren Bausteine, aus denen man die Automaten zusammenstellen kann, stehen zur Verfügung?

  → Zustandsübergänge: $p \xrightarrow{a} q$
- Wie wird die Eingabe eingegeben?

  → Automat liest das Wort von links nach rechts
- Wieviel Speicher steht zur Verfügung? Wie geht man mit dem Speicher um?

  → kein Speicher (nur jener, in welchem der Automat gespeichert wird)

  → Variablen dürfen nicht benutzt werden

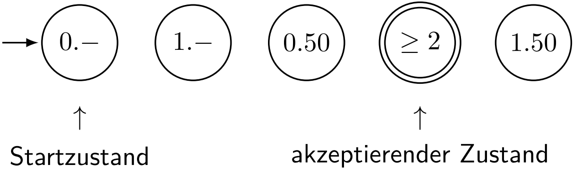
  → einzige Information ist der aktuelle Zustand
- Wie wird die Ausgabe bestimmt (und ausgegeben)?

  → akzeptierende Zustände

Das Modell der endlichen Automaten

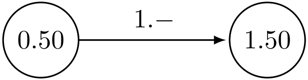
Ein **endlicher Automat** besteht aus

- Zuständen:



- Eingabealphabet: $0.50,\ 1.{-},\ 2.{-}$

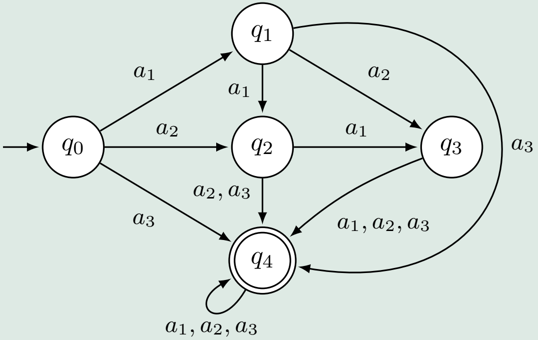
- Übergangsfunktion: z.B.



Darstellung endlicher Automaten

Beispiel (Einstiegsaufgabe: graphische Darstellung)

B:



Zustände:
$q_0 =$ Start
$q_1 = 0.50$
$q_2 = 1.–$
$q_3 = 1.50$
$q_4 = \geq 2$

Eingabealphabet:
$a_1 = 0.50$
$a_2 = 1.–$
$a_3 = 2.–$

Darstellung endlicher Automaten

Definition (Endlicher Automat)

Ein **(deterministischer) endlicher Automat** (EA) ist ein Quintupel

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

mit

- *endliche* Menge von **Zuständen** $Q = \{q_0, q_1, \ldots, q_n\}$ $(n \in \mathbb{N})$
- **Eingabealphabet** $\Sigma = \{a_1, a_2, \ldots, a_m\}$ $(m \in \mathbb{N})$
- **Übergangsfunktion** $\delta \colon Q \times \Sigma \to Q$
- **Startzustand** $q_0 \in Q$
- Menge der **akzeptierenden Zustände** $F \subseteq Q$

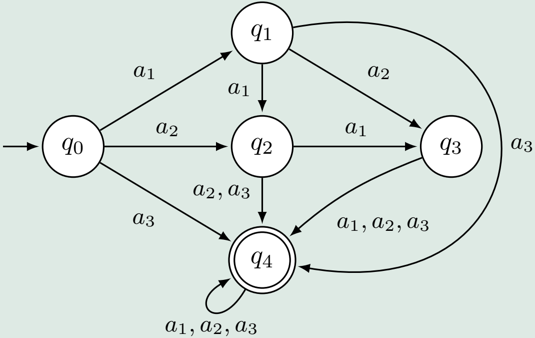
Anmerkung:Das **kartesische Produkt von** A **und** B ist definiert als

$$A \times B = \{(a, b) \mid a \in A \text{ und } b \in B\}.$$

Darstellung endlicher Automaten

Beispiel (Einstiegsaufgabe: Übergangsfunktion)

$\delta(q, a) = p$ bedeutet: EA wechselt zu p, falls in q Symbol a gelesen wird



$$\delta(q_0, a_1) = q_1$$
$$\delta(q_0, a_2) = q_2$$
$$\delta(q_0, a_3) = q_4$$
$$\delta(q_1, a_1) = q_2$$
$$\delta(q_1, a_2) = q_3$$
$$\delta(q_1, a_3) = q_4$$
$$\delta(q_2, a_1) = q_3$$

usw.

Darstellung endlicher Automaten

Beispiel (Einstiegsaufgabe: formale Darstellung)

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

- Zustandsmenge $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Eingabealphabet $\Sigma = \{a_1, a_2, a_3\}$
- Startzustand q_0
- akzeptierende Zustände $F = \{q_4\}$
- Übergangsfunktion δ:

Zustand	Eingabe		
	a_1	a_2	a_3
q_0	q_1	q_2	q_4
q_1	q_2	q_3	q_4
q_2	q_3	q_4	q_4
q_3	q_4	q_4	q_4
q_4	q_4	q_4	q_4

Aufbau von Berechnungsmodellen

Schema für den Aufbau von Berechnungsmodellen:

1. Definiere die *Struktur*, welche die exakte Beschreibung jedes Objekts der Modellklasse ermöglicht. ✓
2. Beschreibung der *Bedeutung (Semantik)* der Struktur:
   - *Konfiguration:* Vollständige Beschreibung einer Situation, in der sich das Modell befindet.
   - *Berechnungsschritt:* Übergang aus einer Konfiguration in eine andere durch die Ausführung eines Befehls des Modells.
3. *Berechnung:* Folge solcher elementaren Berechnungsschritte
4. Nun kann man jeder Eingabe das Arbeitsresultat als Ausgabe zuordnen.

To do: 2 bis 4 definieren

Situation in einem endlichen Automaten

Die vollständige Bechreibung der Situation, in der sich der Automat befindet, muss alle Informationen enthalten, die noch Einfluss auf spätere Berechnungen haben könnten.

Definition (Konfiguration)

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein EA. Eine **Konfiguration** von M ist ein Element aus $Q \times \Sigma^*$.

Wenn M sich in einer Konfiguration $(q, w) \in Q \times \Sigma^*$ befindet, bedeutet dies, dass M im Zustand q ist und noch das Suffix w des Eingabewortes lesen soll.

Zwei spezielle Konfigurationen

Definition (Startkonfiguration)

Eine Konfiguration $(q_0, w) \in \{q_0\} \times \Sigma^*$ nennen wir eine **Startkonfiguration von M auf w**.

Die Berechnung von M auf w muss in der Startkonfiguration beginnen und in einer Konfiguration enden, bei der alle Symbole von w schon gelesen worden sind:

Definition (Endkonfiguration)

Jede Konfiguration aus $Q \times \{\varepsilon\}$ nennt man eine **Endkonfiguration**.

Schritt von einer zur nächsten Situation

Beispiel (Einstiegsaufgabe: Berechnungsschritt)

Konfiguration: $(q_2, a_3a_2a_1a_2)$

- Zustand q_2
- $a_3a_2a_1a_2$ noch zu lesen

$$\Big\downarrow \text{ Anwendung von } \delta(q_2, a_3) = q_4$$

Konfiguration: $(q_4, a_2a_1a_2)$

- Zustand q_4
- $a_2a_1a_2$ noch zu lesen

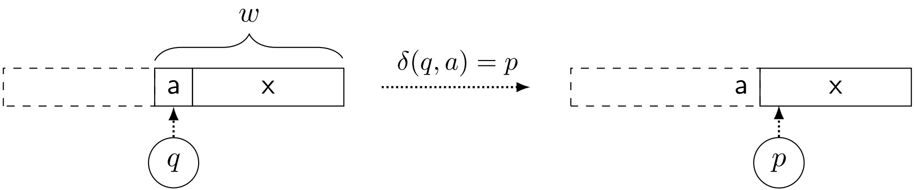
Schritt von einer zur nächsten Situation

Definition (Berechnungsschritt)

Ein **Berechnungsschritt** $\vdash_M$ von M ist die Anwendung der Übergangsfunktion auf die aktuelle Konfiguration und ist definiert durch

$$(q, w) \vdash_M (p, x)$$

genau dann, wenn $w = ax$, $a \in \Sigma$ und $\delta(q, a) = p$ gilt.

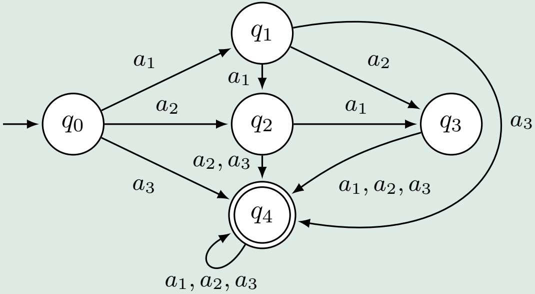


Konvention: Wir schreiben nur $\vdash$, wenn aus dem Kontext klar ist, um welchen Automaten M es sich handelt.

Folge von Schritten

Beispiel (Einstiegsaufgabe: Berechnung)

$B:$



$$(q_2, a_3a_2a_1a_2) \vdash_B (q_4, a_2a_1a_2) \vdash_B (q_4, a_1a_2) \vdash_B (q_4, a_2)$$

Folge von Schritten

Definition (Berechnung)

Eine endliche Folge von Berechnungsschritten nennt man eine **Berechnung**. Eine Berechnung

$$(q_a, w_1 w_2 \ldots w_n) \vdash_M (q_b, w_2 \ldots w_n) \vdash_M \ldots \vdash_M (q_e, w_j \ldots w_n)$$

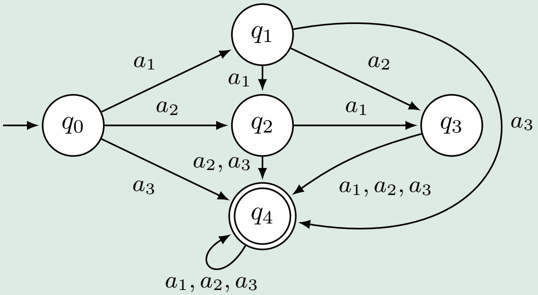
wird abgekürzt als $(q_a, w_1 w_2 \ldots w_n) \vdash_M^* (q_e, w_j \ldots w_n)$ geschrieben.

Definition (Berechnung von M auf w)

Die **Berechnung von M auf einer Eingabe** $w \in \Sigma^*$ ist eine Berechnung, die in der Startkonfiguration (q_0, w) startet und in einer Endkonfiguration (q_e, ε) für irgend ein $q_e \in Q$ endet.
Diese Berechnung ist **akzeptierend**, wenn $q_e \in F$ gilt, und **verwerfend**, falls $q_e \notin F$.

Beispiel (Einstiegsaufgabe: akzeptierende und verwerfende Berechnungen)

B :



- $(q_0, a_2a_1a_2) \vdash_B (q_2, a_1a_2) \vdash_B (q_3, a_2) \vdash_B (q_4, \varepsilon)$ akzeptierend
- $(q_0, a_1a_2) \vdash_B (q_1, a_2) \vdash_B (q_3, \varepsilon)$ verwerfend

Aufbau von Berechnungsmodellen

Schema für den Aufbau von Berechnungsmodellen:

1. Definiere die *Struktur*, welche die exakte Beschreibung jedes Objekts der Modellklasse ermöglicht. ✓
2. Beschreibung der *Bedeutung (Semantik)* der Struktur:
   - *Konfiguration:* Vollständige Beschreibung einer Situation, in der sich das Modell befindet. ✓
   - *Berechnungsschritt:* Übergang aus einer Konfiguration in eine andere durch die Ausführung eines Befehls des Modells. ✓
3. *Berechnung:* Folge solcher elementaren Berechnungsschritte ✓
4. Nun kann man jeder Eingabe das Arbeitsresultat als Ausgabe zuordnen.

Sprache eines endlichen Automaten

Definition (Sprache eines endlichen Automaten)

Die **Sprache** $L(M)$ eines endlichen Automaten M ist die Menge aller von M akzeptierten Wörter:

$$L(M) = \{w \in \Sigma^* \mid \text{Berechnung von } M \text{ auf } w \text{ endet in einem akzeptierenden Zustand}\}$$

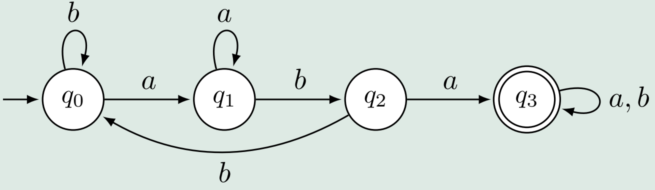
Satz

Die ***Klasse der regulären Sprachen*** *beinhaltet alle Sprachen, die von einem endlichen Automaten akzeptiert werden.*
Jede dieser Sprachen wird ***regulär*** *genannt.*

Sprache eines endlichen Automaten

Beispiel

$M_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\})$



Von M_1 akzeptierte Sprache:

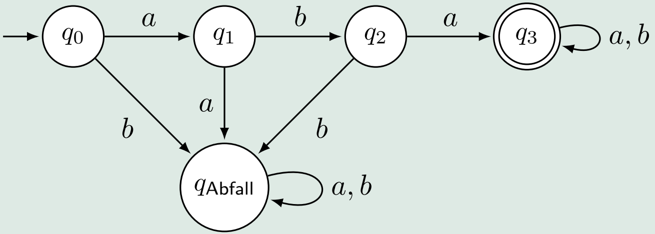
$$\begin{aligned} L(M_1) &= \{xabay \mid x, y \in \{a,b\}^*\} \\ &= \{w \in \{a,b\}^* \mid w \text{ enthält das Teilwort } aba\} \end{aligned}$$

Beispiel

Leichte Modifizierung der Sprache von vorher:

$$
\begin{aligned}
L(M_2) &= \{aba y \mid y \in \{a, b\}^*\} \\
&= \{w \in \{a, b\}^* \mid w \text{ hat als Präfix } aba\}
\end{aligned}
$$

M_2:

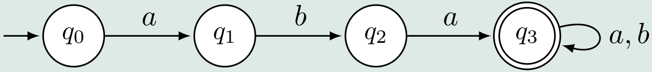


Sprache eines endlichen Automaten

Konvention: Den „Abfall"-Zustand darf man weglassen:

Beispiel

M_2:

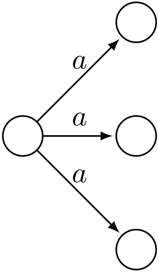


Wenn in einem deterministischen endlichen Automaten Übergänge fehlen, dann führen diese in einen Zustand, von dem kein Übergang mehr wegführt („Abfall"-Zustand).

Nichtdeterminismus

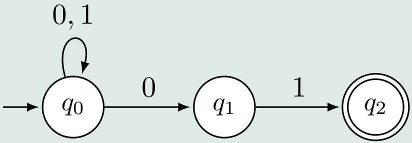
Bis jetzt: Bei *deterministischen* endlichen Automaten folgt jede Konfiguration eindeutig aus der vorhergehenden Konfiguration.

Neu: In *nichtdeterministischen* endlichen Automaten können auf eine Konfiguration entweder keine Konfiguration, genau eine Konfiguration oder mehrere unterschiedliche Konfigurationen folgen.



Beispiel

M_5:



$L(M_5) = \{x01 \mid x \in \{0,1\}^*\}$

Zustand	Eingabe 0	Eingabe 1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

Anmerkung: Der Automat kann also von einem Zustand in mehrere Zustände oder auch in keinen Zustand übergehen.

Nichtdeterministischer endlicher Automat

Definition (Nichtdeterministischer endlicher Automat)

Ein **nichtdeterministischer endlicher Automat** (NEA) ist ein Quintupel

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

wobei Q, Σ, q_0 und F wie beim deterministischen endlichen Automaten (ab jetzt: DEA) definiert sind und die Übergangsfunktion δ definiert ist als

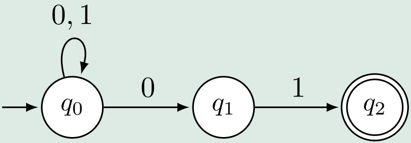
$$\delta\colon Q \times \Sigma \to \mathcal{P}(Q)\,.$$

Anmerkung: $\mathcal{P}(Q)$ bezeichnet die **Potenzmenge** von Q, also die Menge aller Teilmengen von Q.

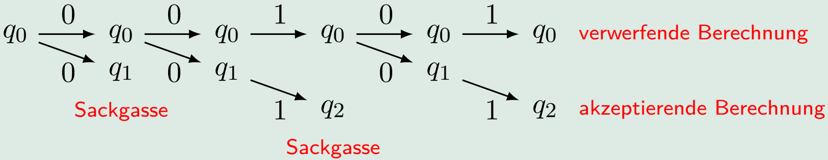
Nichtdeterministischer endlicher Automat

Beispiel

M_5:

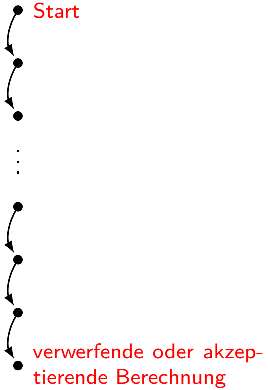


Mögliche Berechnungen auf dem Wort 00101**:**

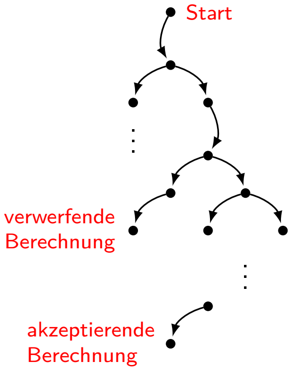


Anmerkung: Ein NEA akzeptiert ein Wort, wenn mindestens eine akzeptierende Berechnung existiert. Er rät also die richtige Berechnung.

Deterministische Berechnung:

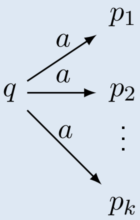


Nichtdeterministische Berechnung:



Definition (Berechnungsschritt in einem NEA)

Ein **Berechnungsschritt in einem NEA** ist die Anwendung der Übergangsfunktion auf die aktuelle Konfiguration. Dargestellt wird ein Berechnungsschritt durch k viele Zweige



wenn $\delta(q, a) = \{p_1, p_2, \ldots, p_k\}$ für ein $k \in \mathbb{N}$ gilt.

Nichtdeterministische Berechnung

Definition (Berechnung eines NEA auf w)

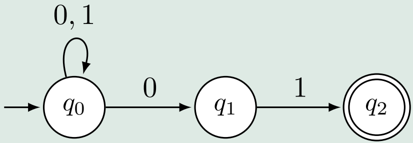
Die **Berechnung eines NEA** auf w startet in der Startkonfiguration. Dann wird in jedem Zweig des ersten Berechnungsschrittes der nächste Berechnungsschritt durchgeführt. Das geht so weiter, bis der NEA entweder in einer Sackgasse landet oder das ganze Wort gelesen worden ist.

Definition (akzeptierende Berechnung)

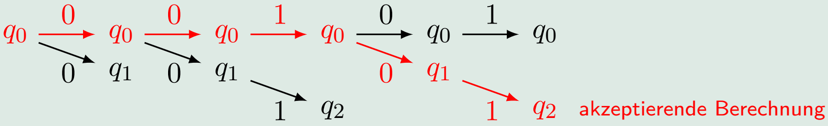
Die Berechnung des NEAs auf einem Wort w ist **akzeptierend**, wenn *mindestens eine* der Berechnungen auf w in einen akzeptierenden Zustand führt.

Beispiel

M_5:



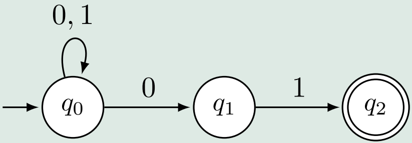
Mögliche Berechnungen auf dem Wort 00101**:**



$\Longrightarrow$ 00101 wird von M_5 akzeptiert, also gilt $00101 \in L(M_5)$.

Beispiel

M_5:



Mögliche Berechnungen auf dem Wort 0010:

$$
\begin{tikzcd}
q_0 \arrow[r, "0"] \arrow[dr, "0"'] & q_0 \arrow[r, "0"] \arrow[dr, "0"'] & q_0 \arrow[r, "1"] & q_0 \arrow[r, "0"] \arrow[dr, "0"'] & q_0 \\
& q_1 & q_1 \arrow[dr, "1"'] & & q_1 \\
& & & q_2 &
\end{tikzcd}
$$

$\Longrightarrow$ keine der Berechnungen ist akzeptierend, also ist $0010 \notin L(M_5)$.

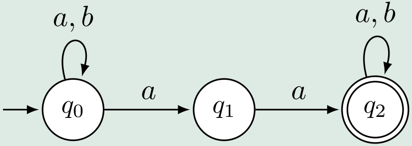
Sprache eines NEA

Definition

Die **Sprache** $L(M)$ eines NEA M ist die Menge aller Wörter, auf denen *mindestens eine* akzeptierende Berechnung existiert.

Beispiel

M_6:



$\implies L(M_6) = \{w \in \{a,b\}^* \mid \text{w enthält das Teilwort } aa\}$

Äquivalenz von DEA und NEA

Frage:

Sind NEAs mächtiger als DEAs?

Antwort: Nein!

Satz (Gleichmächtigkeit von DEA und NEA)

Es gibt einen DEA, der die Sprache L akzeptiert. $\Longleftrightarrow$ *Es gibt einen NEA, der die Sprache L akzeptiert.*

Beweis.

$\Longrightarrow$ Jeder DEA ist ein NEA.

$\Longleftarrow$ Teilmengenkonstruktion (siehe nächste Folie)

$\square$

Teilmengenkonstruktion

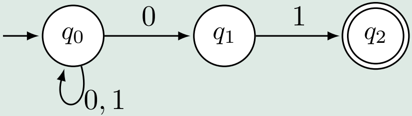
Sei $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ ein NEA. Der dazu äquivalente DEA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ wird konstruiert durch

- $Q_D = \mathcal{P}(Q_N)$
  (Menge *aller Teilmengen* von Q_N)
- $F_D = \{S \in Q_D \mid S \cap F_N \neq \emptyset\}$
  (alle Mengen aus Q_D, die mindestens einen akzeptierenden Zustand aus F_N enthalten)
- $\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$ für $S \in Q_D$ und $a \in \Sigma$
  (Menge aller Zustände von D, die von den Zuständen aus S durch Lesen von a erreichbar sind)

Teilmengenkonstruktion

Beispiel

NEA:

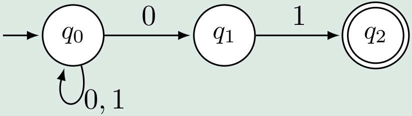


Zustand	Eingabe 0	Eingabe 1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\} = A$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\} = B$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\} = C$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

$\rightarrow$ ist der Startzustand und
$*$ sind die akzeptierenden Zustände

Beispiel

NEA:



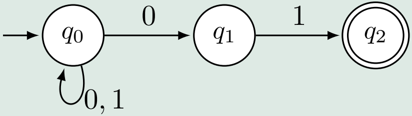
Zustand	Eingabe 0	Eingabe 1
$\rightarrow \{q_0\} = A$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\} = B$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\} = C$	$\{q_0, q_1\}$	$\{q_0\}$

Nicht erreicht werden können die Zustände:

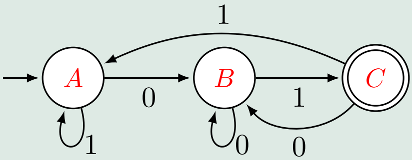
$\emptyset$, $\{q_1\}$, $\{q_2\}$, $\{q_0, q_2\}$, $\{q_0, q_1, q_2\}$

Beispiel

NEA:

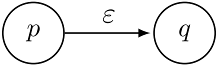


Resultierender **DEA**:



Zustand	Eingabe 0	Eingabe 1
$\rightarrow \{q_0\} = A$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\} = B$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\} = C$	$\{q_0, q_1\}$	$\{q_0\}$

Eine weitere Erweiterung des Automatenmodells zur Erleichterung der Automatenkonstruktion: **ε-Übergänge**



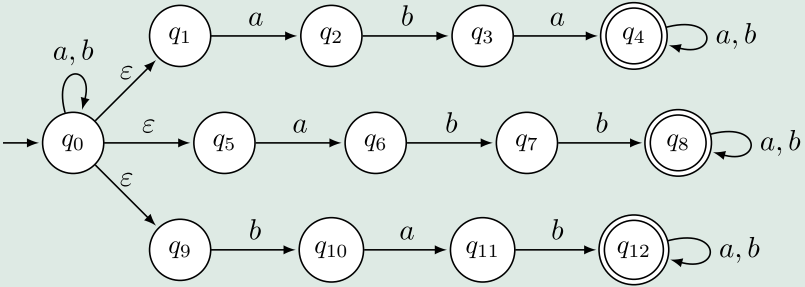
Der NEA kann so spontan den Zustand wechseln, ohne ein Eingabesymbol zu lesen.

NEAs mit ε-Übergängen

Beispiel

Suche nach einem von mehreren Mustern:

$$L = \{w \in \{a,b\}^* \mid w \text{ enthält eines der Teilwörter } aba, abb, bab\}$$



NEAs mit ε-Übergängen

Definition (Nichtdeterministischer endlicher Automat)

Ein **nichtdeterministischer endlicher Automat mit ε-Übergängen** (ε-NEA) wird beschrieben durch

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

wobei Q, Σ, q_0 und F wie beim deterministischen endlichen Automaten definiert sind und die Übergangsfunktion δ definiert ist als

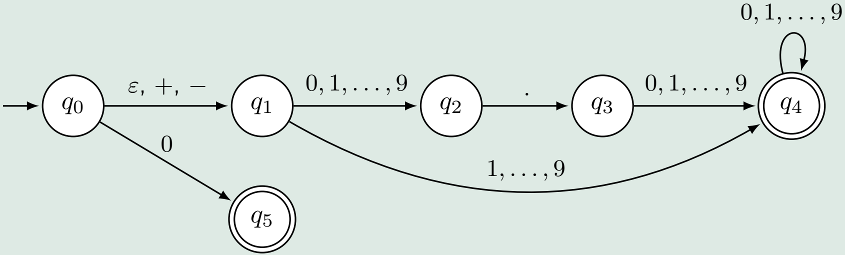
$$\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)\,.$$

Annahme: $\varepsilon \notin \Sigma$, damit es zu keinen Verwechslungen kommt.

Beispiel

ε-NEA, der Dezimalzahlen akzeptiert:

- optionales $+$ oder -
- Zeichenreihe von Ziffern mit optionalem Dezimalpunkt nach der ersten Ziffer
- Höchstens eine Ziffer vor dem Dezimalpunkt



Äquivalenz von DEA ε-NEA

Satz (Gleichmächtigkeit von DEA und ε-NEA)

Es gibt einen DEA, der die Sprache L akzeptiert. $\iff$ *Es gibt einen ε-NEA, der die Sprache L akzeptiert.*

Beweisidee.

$\Longrightarrow$ Jeder DEA ist ein ε-NEA.

$\Longleftarrow$ Teilmengenkonstruktion unter Berücksichtigung der ε-Übergängen (sogenannte **$\boldsymbol{\varepsilon}$-Hüllen** der Teilmengen bilden)

Vollständiger Beweis: siehe z. B. Hopcroft et al. S. 101 ff $\square$

Äquivalenz DEA und RA

Satz (Gleichmächtigkeit von RA und DEA)

Es gibt einen DEA, der die Sprache L akzeptiert. $\iff$ *Es gibt einen RA, der die Sprache L akzeptiert.*

Beweis.

$\Longrightarrow$ Dynamische Programmierung: Für jedes Paar von Zuständen p, q regulären Ausdrück finden, der alle Wörter beschreibt, die von p nach q führen.

$\Longleftarrow$ RA in einen speziellen NEA umwandeln, der auch spontane Übergänge (ohne ein Eingabesymbol zu lesen) zulässt. Diese sogenannten ε-NEAs können in NEAs und somit durch Teilmengenkonstruktion in DEAs umgewandelt werden.

$\square$

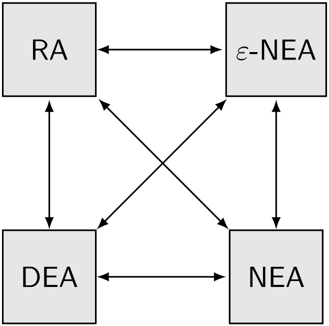
Übersicht über reguläre Sprachen

Reguläre Sprachen sind durch verschiedene äquivalente Mechanismen darstellbar:

Akzeptierender Mechanismus: DEA, NEA, ε-NEA

Beschreibender Mechanismus: RA

(**Generierender Mechanismus:** Reguläre (rechtslineare) kontextfreie Grammatiken)



Abschlusseigenschaften regulärer Sprachen

Satz

Seien L_1 und L_2 zwei reguläre Sprachen über Σ. Dann ist die ***Vereinigung***

$$L_1 \cup L_2 = \{w \mid w \in L_1 \textit{ oder } w \in L_2\}$$

auch regulär.

Beweis.

Weil L_1 und L_2 regulär sind, existieren reguläre Ausdrücke α_1 und α_2 für L_1 und L_2, das heisst $L(\alpha_1) = L_1$ und $L(\alpha_2) = L_2$.
Aus der Definition für reguläre Ausdrücke folgt, dass $\alpha_1|\alpha_2$ ein regulärer Ausdruck ist, also ist

$$L(\alpha_1|\alpha_2) = L_1 \cup L_2$$

regulär. □

Abschlusseigenschaften regulärer Sprachen

Satz

*Sei L eine reguläre Sprache über Σ. Dann ist auch das **Komplement***

$$\overline{L} = \Sigma^* - L = \{w \in \Sigma^* \mid w \notin L\}$$

regulär.

Beweis.

Sei A ein EA für L. Dann gibt es für jedes Wort $w \in \Sigma^*$ einen eindeutigen Zustand, in dem der EA endet, wenn er w liest.
Alle Wörter aus $L(A) = L$ führen in akzeptierende Zustände und alle Wörter aus $\Sigma^* - L = \overline{L}$ in nichtakzeptierende Zustände.
Vertauschen von akzeptierenden und nichtakzeptierenden Zuständen ergibt einen EA für $\overline{L}$. □

Abschlusseigenschaften regulärer Sprachen

Seien L, L_1 und L_2 reguläre Sprachen über Σ. Dann sind auch beispielsweise die folgenden Sprachen regulär (Beweise)[1]:

- **Schnitt:** $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ und } w \in L_2\}$
- **Mengendifferenz:** $L_1 - L_2 = \{w \mid w \in L_1 \text{ und } w \notin L_2\}$
- **Konkatenation:**
  $$L_1 \cdot L_2 = L_1 L_2 = \{w = w_1 w_2 \mid w_1 \in L_1 \text{ und } w_1 \in L_2\}$$
- **Kleenesche Hülle:**
  $$L^* = \{w = w_1 w_2 \ldots w_n \mid w_i \in L \text{ für alle } i \in \{1, 2, \ldots, n\}\}$$

[1] Siehe Hopcroft et al. Kapitel *Abschluss-Eigenschaften regulärer Sprachen*, S. 166 ff.

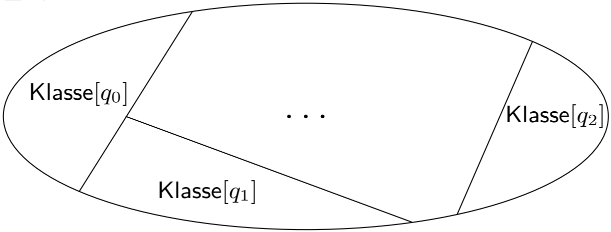
Zustandsklassen

$M = \{\{q_0, q_1, \ldots, q_n\}, \Sigma, \delta, q_0, F\}$ setzt eine Funktion um, die jedem Wort $w \in \Sigma^*$ einen Zustand $q \in Q$ nach dem Lesen von w zuordnet.
$\Longrightarrow$ Σ^* zerfällt in $|Q|$ Klassen:

Definition (Klasse eines Zustands)

$$\text{Klasse}[p] = \{w \in \Sigma^* \mid M \text{ endet nach dem Lesen von } w \text{ in } p\}$$

Σ^*:



Zustandsklassen

Eigenschaften der Klassen:

- Jedes Wort landet in einem Zustand:

$$\Sigma^* = \bigcup_{p \in Q} \text{Klasse}[p]$$

- Kein Wort landet nach dem Lesen in zwei Zuständen:

$$\text{Klasse}[p] \cap \text{Klasse}[q] = \emptyset$$

  für alle $p \neq q$, $p, q \in Q$

- Von M akzeptierte Sprache:

$$L(M) = \bigcup_{p \in F} \text{Klasse}[p]$$

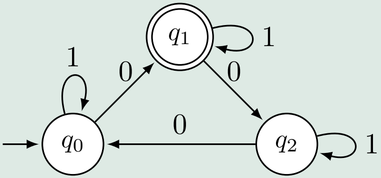
Definition (Anzahl Symbole in einem Wort)

$|w|_a$ ist die Anzahl der Symbole a im Wort w.

Beispiel

$L(M_3) = \{w \in \{0,1\}^* \mid |w|_0 \bmod 3 = 1\}$

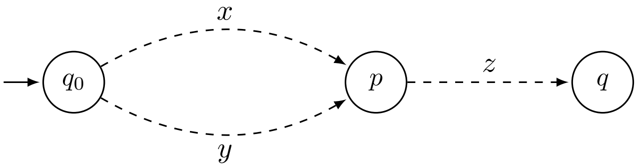
M_3:



Klasse$[q_0] = \{w \in \{0,1\}^* \mid$ (Anzahl Nullen in w) $\bmod 3 = 0\}$
Klasse$[q_1] = \{w \in \{0,1\}^* \mid$ (Anzahl Nullen in w) $\bmod 3 = 1\} = L(M_3)$
Klasse$[q_2] = \{w \in \{0,1\}^* \mid$ (Anzahl Nullen in w) $\bmod 3 = 2\}$

Grenzen endlicher Automaten

Wenn ein EA M nach dem Lesen zweier Präfixe x und y im gleichen Zustand landet, kann er nicht mehr zwischen x und y unterscheiden.



Satz

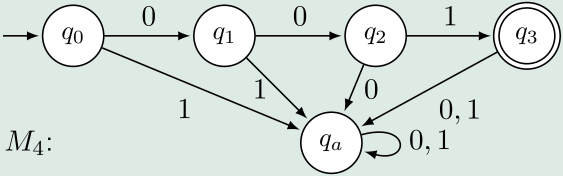
Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein EA und x und y zwei beliebige Wörter aus Σ^*, so dass $x, y \in$ Klasse$[p]$. Dann gilt

$$xz \in L(M) \iff yz \in L(M) \qquad (*)$$

für alle Wörter $z \in \Sigma^*$.

Grenzen endlicher Automaten

Beispiel



$L(M_4) = \{001\}$

Behauptung: Jeder EA, der die Sprache $\{001\}$ akzeptiert, muss zwischen den Präfixen $x = 0$ und $y = 00$ unterscheiden können.

Beweis durch Widerspruch: Annahme $x, y \in$ Klasse$[p]$.
Dann gilt für $z = 1$: $xz = 01 \notin \{001\}$, aber $yz = 001 \in \{001\}$.
Das steht im Widerspruch zu (*), also können x und y nicht zur gleichen Wortklasse gehören.

Grenzen endlicher Automaten

Untere Schranke für die Grösse endlicher Automaten:

Mit der vorherigen Technik können wir zeigen, dass ein Automat für eine bestimmte Sprache L mindestens n Zustände benötigt:

1. Überlegen, wie die n Zustandsklassen für den zu L gehörigen EA aussehen könnten.
2. n unterschiedliche Wörter finden, von denen keine zwei zur gleichen Zustandsklasse gehören.
3. Für je zwei dieser Wörter mit Hilfe von (*) und einem Widerspruchsbeweis zeigen, dass der EA zwischen diesen beiden Wörtern unterscheiden muss.
4. Daraus folgt direkt, dass jeder EA für L mindestens zwischen diesen n Zustandsklassen unterscheiden muss.

Beispiel

Zeige: Jeder EA für die Sprache
$L(M_3) = \{w \in \{0,1\}^* \mid |w|_0 \bmod 3 = 1\}$ hat mindestens 3 Zustände.

Beweis:

1. Intuition: Jeder EA für $L(M_3)$ muss die Anzahl der gelesenen Nullen modulo 3 zählen.
2. $x_1 = \varepsilon$, $x_2 = 0$, $x_3 = 00$
3. Widerspruch für alle Paare von Wörtern:
   Für x_1 und x_2: $z_{12} = \varepsilon \Rightarrow x_1 z_{12} = \varepsilon \notin L, \quad x_2 z_{12} = 0 \in L$
   Für x_1 und x_3: $z_{13} = 0 \Rightarrow x_1 z_{13} = 0 \in L, \quad x_3 z_{13} = 000 \notin L$
   Für x_2 und x_3: $z_{23} = \varepsilon \Rightarrow x_2 z_{23} = 0 \in L, \quad x_3 z_{23} = 00 \notin L$
4. Jeder EA für $L(M_3)$ muss zwischen mindestens drei Zuständen unterscheiden. □

Nichtregularität

Nichtexistenz von EA:

1. Müsste der Automat für L eine unendliche Anzahl von Eigenschaften speichern?
2. Unendlich viele Wörter finden, von denen keine zwei zur gleichen Zustandsklasse gehören.
3. Für je zwei dieser Wörter mit Hilfe von (*) und einem Widerspruchsbeweis zeigen, dass der EA zwischen diesen beiden Wörtern unterscheiden muss.
4. Ein Automat kann nur endlich viele Zustände haben, also ist L nicht regulär.

Nichtregularität

Beispiel

Zeige: Die Sprache $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ ist nicht regulär.

Beweis:

1. Der Automat müsste zuerst alle Nullen zählen $\rightarrow$ unendliche viele Eigenschaften.
2. $x_1 = 0, x_2 = 00, x_3 = 0^3, \ldots, x_i = 0^i, \ldots$
3. Keine zwei dieser Wörter gehören zur gleichen Zustandsklasse:
   Wähle $x_i = 0^i$, $x_j = 0^j$ für $i \neq j \in \mathbb{N}$. Dann sei $z_{ij} = 1^i$.
   $\Rightarrow x_i z_{ij} = 0^i 1^i \in L$, aber $x_j z_{ij} = 0^j 1^i \notin L$.
4. Der Automat hätte unendlich viele Zustände. Also kann L nicht regulär sein. $\square$