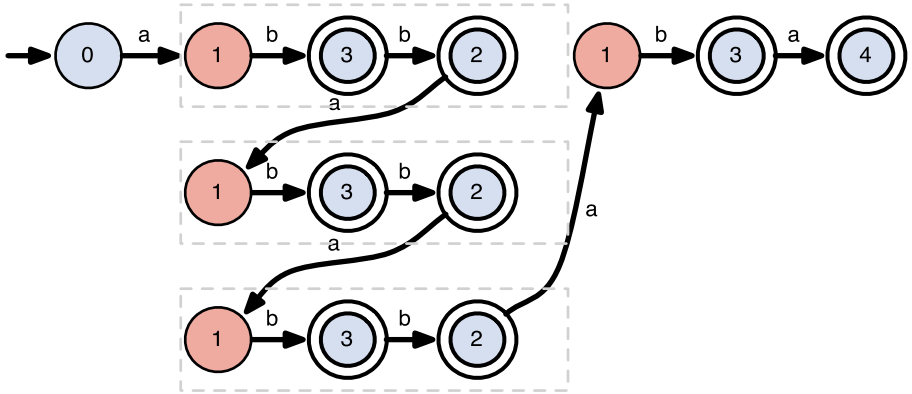
Berechenbarkeitstheorie

1. Vorlesung



Dr. Franziska Jahnke
Institut für Mathematische Logik und Grundlagenforschung
WWU Münster

Zentrale Themen

- Wie definiert man einen Computer formal?

  ⇝ Automatentheorie und Turingmaschinen

- Gibt es Probleme, die ein Computer nicht lösen kann? Falls ja, können wir eins finden?

  ⇝ Berechenbarkeitstheorie

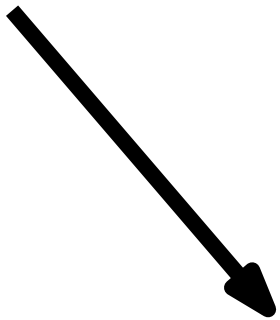
- Manche Probleme, die ein Computer lösen kann sind leicht, andere schwer. Wie können wir Probleme klassifizieren?

  ⇝ Komplexitätstheorie

Ziele der Vorlesung

- *Formales* Arbeiten lernen
- Verständnis für *Berechenbarkeit*
- Grundlagen der Theoretischen Informatik
  - Entwurf und Analyse von Algorithmen
  - Optimierung von Schaltkreisen
  - Programmverifikation
  - Compilerbau
  - Komplexitätstheorie
  - ....

Info I + II: Wie berechne ich etwas?

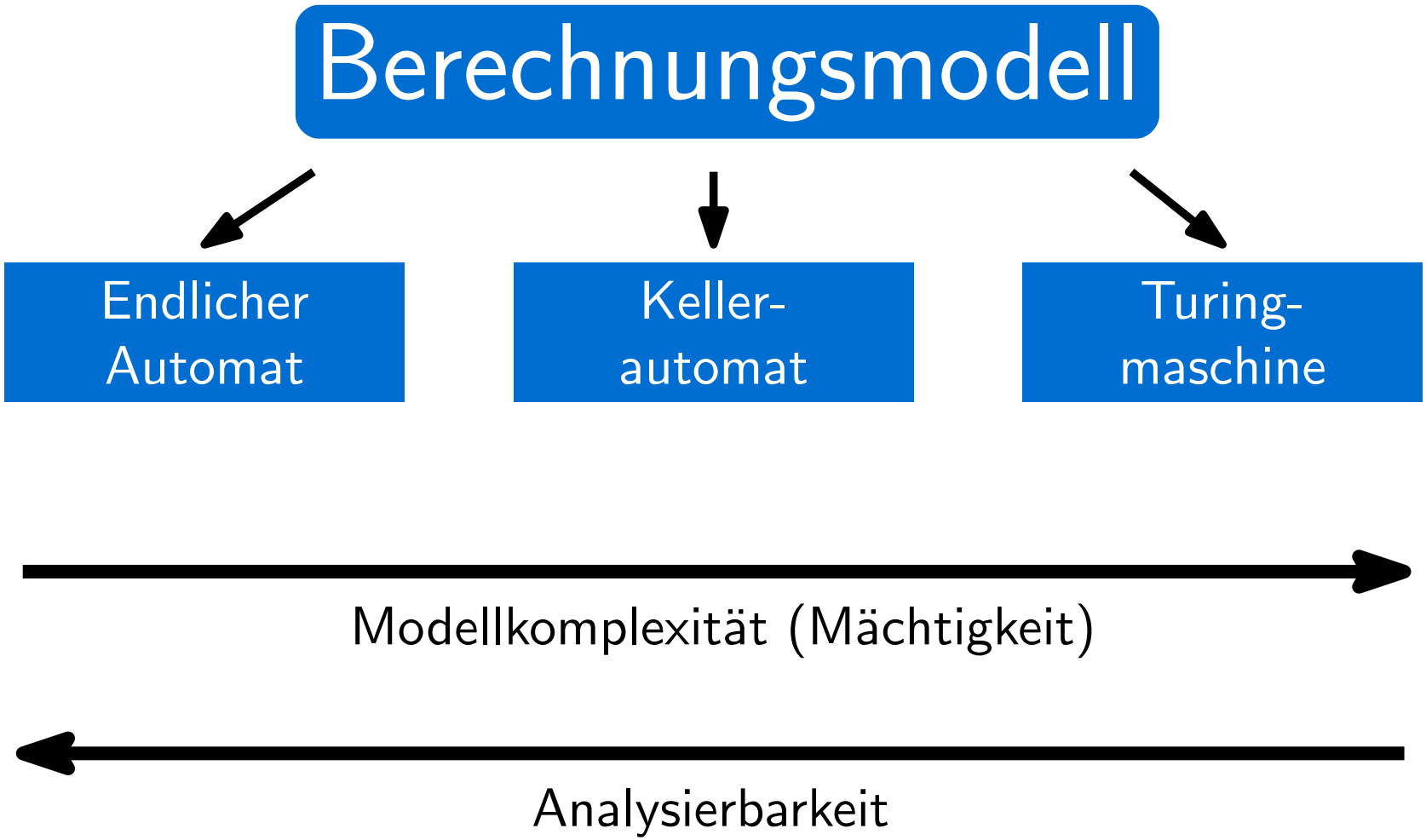


Berechenbarkeit

BT: Was kann ich berechnen?

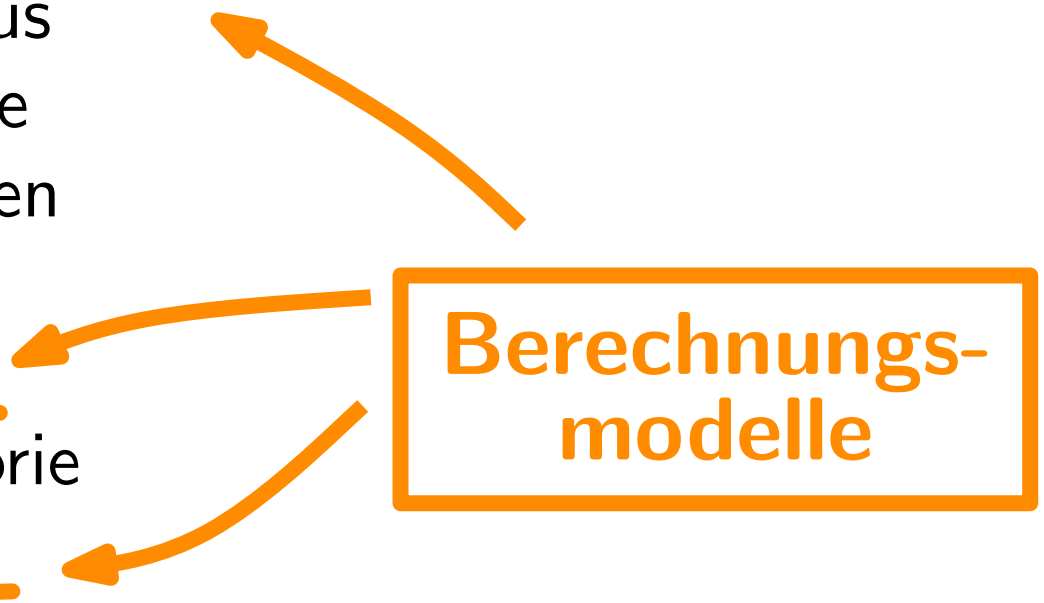
Kommt drauf an ...

- Welches Berechnungsmodell benutze ich?
- Wieviel Ressourcen stehen mir zur Verfügung? (Zeit/Speicher/Energie/...)


Berechnungsmodell
Endlicher Automat
Keller-automat
Turing-maschine
Modellkomplexität (Mächtigkeit)
Analysierbarkeit

Vorlesungsübersicht

1. Reguläre Sprachen
   - endliche Automaten
   - Nichtdeterminismus
   - reguläre Ausdrücke
2. Kontext-freie Sprachen
   - Grammatiken
   - Kellerautomaten
3. Berechenbarkeitstheorie
   - Turingmaschinen
   - Unentscheidbarkeit
4. Komplexitätstheorie
   - P und NP
   - NP-vollständige Problem



1. Kapitel

Reguläre Sprachen

Formale Sprachen

Formale Sprachen sind unser Hilfsmittel um Probleme und ihre Instanzen zu beschreiben!

Grundlegende Begriffe

- Ein **Alphabet** bezeichnet in dieser Vorlesung eine endliche Menge.
- Die Elemente eines Alphabetes nennen wir **Zeichen** oder **Buchstaben**.
- Ein **Wort** ist eine Folge von Zeichen.
- Eine **Sprache** ist eine Menge von Wörtern.

Beispiele: Alphabet: $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$
Sprache: $L = \{\mathtt{ab}, \mathtt{cbbb}, \mathtt{abcb}\}$
Sprache: $L = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \ldots\}$

- Länge eines Wortes = Anzahl der Zeichen
- Das leere Wort ist ein spezielles Wort der Länge 0, wir verwenden als Symbol ε (Epsilon)
- Die Menge aller **Wörter der Länge** k über einem Alphabet Σ bezeichnen wir mit Σ^k
- Die Menge **aller** Wörter über einem Alphabet Σ bezeichnen wir mit Σ^*, d.h. $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$.
- Die Menge **aller nicht-leeren** Wörter über einem Alphabet Σ bezeichnen wir mit Σ^+, d.h. $\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i = \Sigma^* \setminus \{\varepsilon\}$.

Beispiele:
$\Sigma = \{0, 1\}$
$\Sigma^2 = \{00, 10, 01, 11\}$
$\Sigma^0 = \{\varepsilon\} \neq \emptyset$
Ist L eine Sprache, dann gilt $L \subseteq \Sigma^*$

Kodierungen

Für die maschinelle Bearbeitung von Probleminstanzen müssen diese sinnvoll kodiert werden!

Alle Instanzen der Eingabe werden (sinnvoll) als Wörter über einem geeigneten Alphabet kodiert

Kodierungsfunktion: $\langle \cdot \rangle : \mathcal{I} \to \Sigma^*$

Beispiel: Multiplikation zweier natürlicher Zahlen

Instanzen: Paare ganzer Zahlen $\mathbf{N} \times \mathbf{N}$

Alphabet: $\Sigma = \{0, 1, \#\}$

Kodierung: $\langle (a, b) \rangle := \text{bin}(a)\#\text{bin}(b)$

Bsp: $\langle (3, 9) \rangle = 11\#1001$

Kodierungen

Für die maschinelle Bearbeitung von Probleminstanzen müssen diese sinnvoll kodiert werden!

Alle Instanzen der Eingabe werden (sinnvoll) als Wörter über einem geeigneten Alphabet kodiert

Kodierungsfunktion: $\langle \cdot \rangle : \mathcal{I} \to \Sigma^*$

Beispiel: Multiplikation zweier natürlicher Zahlen

Instanzen: Paare ganzer Zahlen $\mathbf{N} \times \mathbf{N}$

Alphabet: $\Sigma = \{1, \#\}$

Kodierung: $\langle (a, b) \rangle := 1^a \# 1^b$

Bsp: $\langle (3, 9) \rangle = 111\#111111111$

Probleme

Entscheidungsproblem

Eingabe: Instanz $I \in \mathcal{I}$

Ausgabe: ja oder nein

Beispiel: Eingabe: $I \in \mathcal{I} = \mathbf{N}$

$$\text{Ausgabe: } \begin{cases} \text{ja} & \text{falls } I \text{ Primzahl} \\ \text{nein} & \text{falls } I \text{ zusammengesetzte Zahl} \end{cases}$$

Entscheidungsprobleme werden durch Kodierungen ihrer ja-Instanzen als Sprache formal beschrieben.

$$\begin{aligned} L_{\text{prim}} &= \{\text{bin}(x) \mid k \text{ ist Primzahl}\} \subseteq \{0,1\}^* \\ &= \{10, 11, 101, 111, 1011, 1101, \ldots\} \end{aligned}$$

Operationen für Wörter

- **Konkatenation**: Wenn $u, v \in \Sigma^*$, dann bezeichnen wir mit $u \circ v$ (oft auch kurz uv) das Wort, welches aus den Zeichen von u gefolgt von den Zeichen aus v entsteht.

  Bsp.: $u = \mathtt{abb}$, $u = \mathtt{bca}$, dann ist $uv = \mathtt{abbbca}$.

- **Wiederholung**: Für ein Zeichen $\mathtt{a} \in \Sigma$ bezeichnet $\mathtt{a}^k = \underbrace{\mathtt{a}\,\mathtt{a}\,\cdots\mathtt{a}}_{k \text{ mal}}$

  Bsp.: $\mathtt{b}^3\mathtt{a}^2 = \mathtt{bbbaa}$.

- **Spiegelung**: Für ein Wort $u \in \Sigma^*$ bezeichnet $\overleftarrow{u}$ das Wort mit umgekehrt geordneten Zeichen.

  Bsp.: Wenn $u = \mathtt{abbc}$, dann ist $\overleftarrow{u} = \mathtt{cbba}$.

Operationen für Sprachen

L_1, L_2 Sprachen über Σ

$$L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ oder } w \in L_2\}$$ Vereinigung

$$L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ und } w \in L_2\}$$ Schnitt

$$L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$$ Konkatenation

$$L_1^k = \{w_1 w_2 \cdots w_k \mid w_i \in L_1 \quad i = 1, \ldots, k\}$$

$$L_1^* = \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} L_1^i$$ (Kleene) Stern

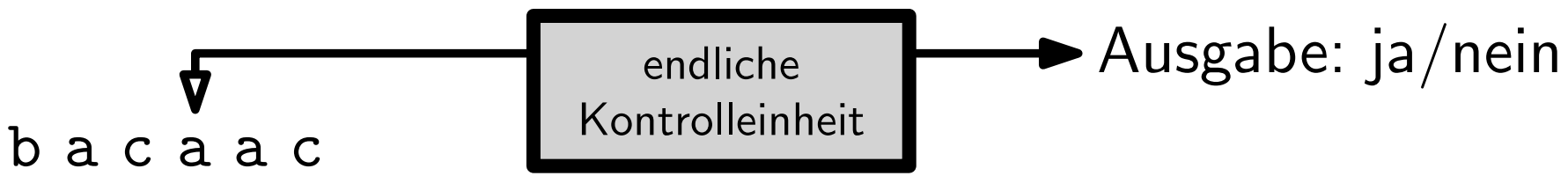
$$\bar{L}_1 = \{w \in \Sigma^* | w \notin L_1\}$$ Komplement

$$\overleftarrow{L}_1 = \{\overleftarrow{w} \mid w \in L_1\}$$ Spiegelung

Vereinigung, Konkatenation, und Kleene Stern heißen **reguläre Operationen**.

Der deterministische endliche Automat (DEA)

- Berechnungsmodell für Entscheidungsprobleme (bei uns in der VL)



- DEA verarbeitet Wort Zeichen für Zeichen (links→ rechts)
- DEA arbeitet intern mit endlich vielen Zuständen
- Übergang des Zustands in Abhängigkeit des aktuell gelesenen Zeichens
- Antwort hängt vom Zustand nach dem Lesen des letzten Zeichens ab

Wie spezifiziere ich einen DEA?

1.) Ich benötige die Bezeichnungen der Zustände.

Durch die **endliche** Menge Q angegeben.
In der Regel $Q = \{q_1, q_2, \ldots, q_n\}$.

2.) Ich muss die Übergänge der Zustände beschreiben.

Durch eine Funktion $\delta\colon Q \times \Sigma \to Q$.

z.B. $\delta(q_2, \mathtt{b}) = q_3$

Q	Σ	Q
q_1	a	q_1
q_1	b	q_3
q_2	a	q_1
q_2	b	q_3
$\vdots$	$\vdots$	$\vdots$

tabellarisch

Wie spezifiziere ich einen DEA?

1.) Ich benötige die Bezeichnungen der Zustände.

Durch die **endliche** Menge Q angegeben.
In der Regel $Q = \{q_1, q_2, \ldots, q_n\}$.

2.) Ich muss die Übergänge der Zustände beschreiben.

Durch eine Funktion $\delta\colon Q \times \Sigma \to Q$.

3.) Ich muss wissen welche Zustände am Ende für *akzeptierend* stehen.

Durch die Angabe aller akzeptierender Zustände als Menge $F \subseteq Q$.

4.) Ich muss den initialen Zustand kennzeichnen.

Durch die Angabe eines *Startzustandes* aus der Menge Q.

Formale Definition

Deterministischer Endlicher Automat

Ein DEA ist ein 5 Tupel $(Q,$ Zustandsmenge $)$bei gilt

- Q ist eine endliche Menge,
- Σ ist ein Alphabet,
- $\delta : Q \times \Sigma \to Q$ eine totale Funktion,
- $q_0 \in Q$,
- $F \subseteq Q$.

Übergangsfunktion

Startzustand

akzeptierende Zustände

Formale Definition

Deterministischer Endlicher Automat

Ein DEA ist ein 5 Tupel $(Q, \Sigma, \delta, q_0, F)$, wobei gilt

- Q ist eine endliche Menge,
- Σ ist ein Alphabet,
- $\delta : Q \times \Sigma \to Q$ eine totale Funktion,
- $q_0 \in Q$,
- $F \subseteq Q$.

Beispiel:

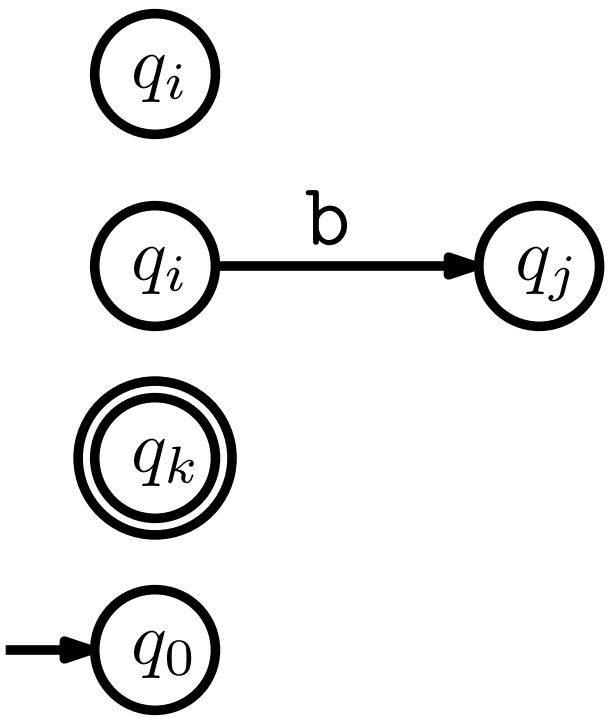
DEA $M = (\{q_1, q_2\}, \{\mathtt{a}, \mathtt{b}\}, \delta, q_1, \{q_2\})$

(Q, Σ, F)

δ:

Q	Σ	Q
q_1	a	q_1
q_1	b	q_2
q_2	a	q_2
q_2	b	q_1

Graphische Notation



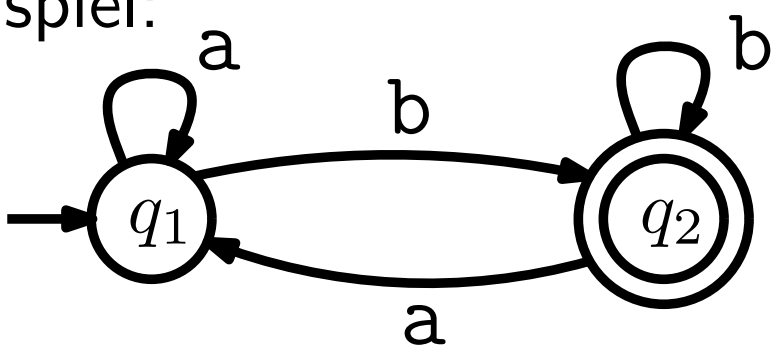
Zustand q_i

Zustandsübergang $\delta(q_i, \mathtt{b}) = q_j$

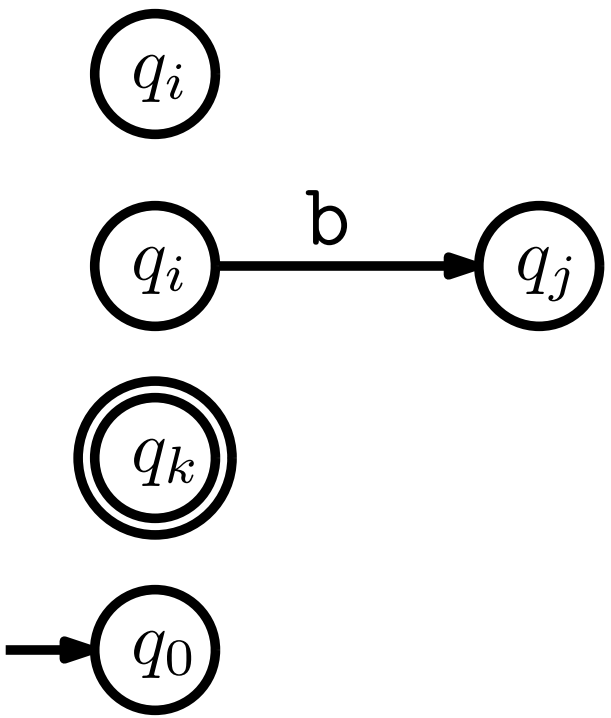
akzeptierender Zustand q_k

Startzustand q_0

Beispiel:



Graphische Notation



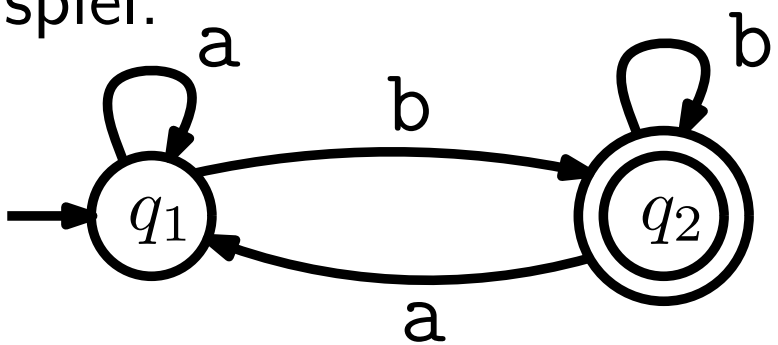
Zustand q_i

Zustandsübergang $\delta(q_i, \mathtt{b}) = q_j$

akzeptierender Zustand q_k

Startzusta

Beispiel:



$M = (\{q_1, q_2\}, \{\mathtt{a}, \mathtt{b}\}, \delta, q_1, \{q_2\})$

δ:

Q	Σ	Q
q_1	a	q_1
q_1	b	q_2
q_2	a	q_1
q_2	b	q_2

Arbeitsweise

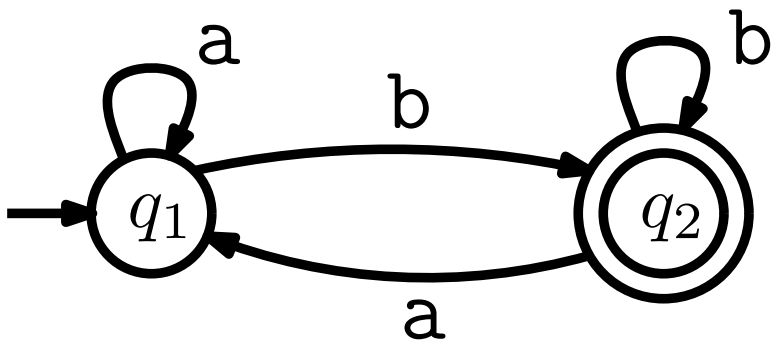
w-Lauf eines DEAs M für ein Wort $w \in \Sigma^*$:

→ Folge von Zustandsübergängen $(q_{i_1}, q_{i_2}, \ldots, q_{i_m})$ mit

1. $q_0 = q_{i_1}$,
2. $w = x_1 x_2 x_3 \cdots x_{m-1}$, mit $x_i \in \Sigma$,
3. $\delta(q_{i_k}, x_i) = q_{i_{k+1}}$ für alle $1 \leq k < m$.

w-Lauf, für $w =$ abba:

$(q_1, q_1, q_2, q_2, q_1)$



Arbeitsweise

w-Lauf eines DEAs M für ein Wort $w \in \Sigma^*$:

→ Folge von Zustandsübergängen $(q_{i_1}, q_{i_2}, \ldots, q_{i_m})$ mit

1. $q_0 = q_{i_1}$,
2. $w = x_1 x_2 x_3 \cdots x_{m-1}$, mit $x_i \in \Sigma$,
3. $\delta(q_{i_k}, x_i) = q_{i_{k+1}}$ für alle $1 \leq k < m$.

Akzeptanz eines Wortes (DEA)

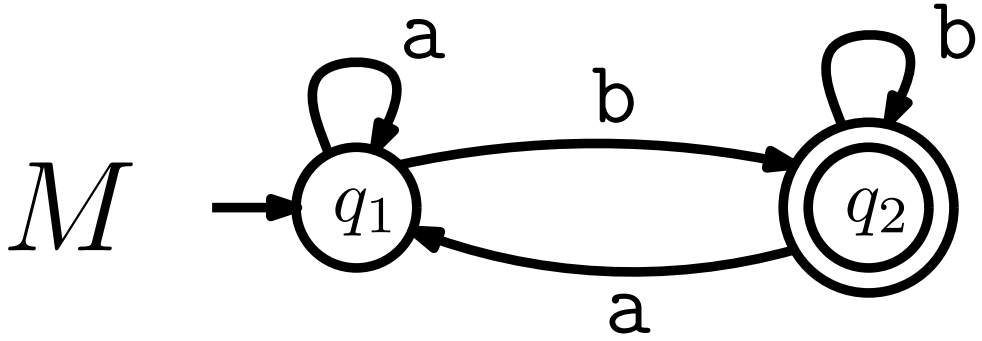
Ein DEA M **akzeptiert ein Wort** $w \in \Sigma^*$ genau dann, wenn der w-Lauf von M in einem akzeptierenden Zustand endet.

Erkennen einer Sprache (DEA)

Ein DEA M **erkennt (akzeptiert) die Sprache**

$$L(M) := \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}.$$

Beispiel 1



- Wort: abaa
  $\rightsquigarrow$ abaa-Lauf endet mit q_1
  da $q_1 \notin F$ folgt, abaa $\notin L(M)$
- Wort: bb
  $\rightsquigarrow$ bb-Lauf endet mit q_2
  da $q_2 \in F$ folgt, bb $\in L(M)$
- Wort: ε
  $\rightsquigarrow$ ε-Lauf endet mit q_1
  da $q_1 \notin F$ folgt, $\varepsilon \notin L(M)$

$$L(M) = \{w \in \Sigma^* \mid w \text{ endet auf b}\}$$

Iterierter Übergang

→ Formalismus für spätere Beweise

Definition

Für einen DEA $M = (Q, \Sigma, \delta, q_0, F)$ ist die **iterierte Übergangsfunktion** $\delta^k : Q \times \Sigma^k \to Q$ definiert durch:

1. $\delta^0(q, \varepsilon) = q$ für alle $q \in Q$,
2. $\delta^i(q, wx) = \delta(\delta^{i-1}(q, w), x)$ für alle $q \in Q$, $w \in \Sigma^{i-1}$, $x \in \Sigma$.

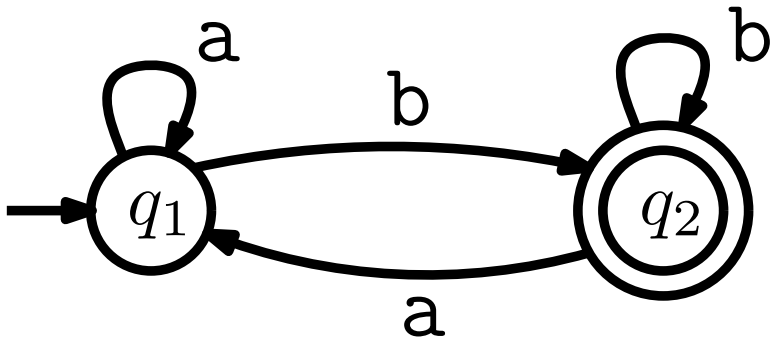
Schreibweise: $\delta^*(q, w)$ für $\delta^{|w|}(q, w)$

Alternative Definition der erkannten Sprache von $M = (Q, \Sigma, \delta, q_0, F)$:

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

Beispiel 1

M



- Wort: abaa

  $\delta^*(q_1, \mathtt{abaa}) = q_1$
  da $q_1 \notin F$ folgt, $\mathtt{abaa} \notin L(M)$

- Wort: bb

  $\delta^*(q_1, \mathtt{bb}) = q_2$
  da $q_2 \in F$ folgt, $\mathtt{bb} \in L(M)$

- Wort: ε

  $\delta^*(q_1, \varepsilon) = q_1$
  da $q_1 \notin F$ folgt, $\varepsilon \notin L(M)$

$$L(M) = \{w \in \Sigma^* \mid w \text{ endet auf b}\}$$