

# Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik  
Humboldt-Universität zu Berlin

WS 2018/19

- <https://hu.berlin/ethi18>

bzw.

- <https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws18/einftheo>

## Anmeldung

- bei Agnes (möglichst bald)
- und bei Moodle (wg. Punktevergabe und Zuordnung zu Übungsgruppen)
- Mails von Agnes und von Moodle werden standardmäßig an den CMS-Account gesendet (bitte regelmäßig checken)

## Abgabe der Aufgabenblätter

- in der VL sowie bei Moodle und auf der VL-Webseite

## Rückgabe

- in den Übungsgruppen

## Bearbeitung der Aufgaben

- in Gruppen von **zwei bis drei** Teilnehmern
- Teilnehmer müssen **nicht** in der gleichen Übungsgruppe sein
- bitte jede Aufgabe auf einem **separaten** Blatt bearbeiten, da diese getrennt abzugeben sind
- bitte auf **jedem Blatt** Folgendes angeben:
  - die Namen und **CMS-Benutzernamen** der Gruppenteilnehmer
  - den Namen der **Abgabegruppe** aus Moodle (z.B. AG123)
  - den Übungstermin, an dem Sie die korrigierten Blätter zurückerhalten möchten

## Scheinkriterien

- Lösen von  $\geq 50\%$  der schriftlichen Aufgaben
- Lösen von  $\geq 50\%$  der Multiple-Choice Aufgaben in Moodle

## Klausur

- **Termin: 28.02.2019, 12 Uhr**
- **Nachklausur: 02.04.2019, 9 Uhr**

## Skript

- wird wöchentlich ins Netz gestellt

Gibt es zum organisatorischen Ablauf noch Fragen?

## Themen dieser VL:

- Welche Rechenmodelle eignen sich zur Lösung welcher algorithmischen Problemstellungen? **Automatentheorie**
- Welche algorithmischen Probleme sind überhaupt lösbar? **Berechenbarkeitstheorie**
- Welcher Aufwand ist zur Lösung eines geg. algorithmischen Problems nötig? **Komplexitätstheorie**

## Themen der VL Algorithmen und Datenstrukturen:

- Wie lassen sich praktisch relevante Problemstellungen möglichst effizient lösen? **Algorithmik**

## Themen der VL Logik in der Informatik:

- Mathem. Grundlagen der Informatik, Beweise führen, Modellierung **Aussagenlogik, Prädikatenlogik**

- Überblick über die wichtigsten Rechenmodelle (Automaten) wie z.B.
  - endliche Automaten
  - Kellerautomaten
  - Turingmaschinen
  - Registermaschinen
  - Schaltkreise
- Charakterisierung der Klassen aller mit diesen Rechenmodellen lösbarer Probleme durch
  - unterschiedliche Typen von formalen Grammatiken
  - Abschlusseigenschaften unter geeigneten Sprachoperationen
  - Reduzierbarkeit auf typische Probleme (Vollständigkeit)
- Erkennen von Grenzen der Berechenbarkeit
- Klassifikation wichtiger algorithmischer Probleme nach ihrer Komplexität

- Rechenmaschinen spielen in der Informatik eine zentrale Rolle
- Es gibt viele unterschiedliche math. Modelle für Rechenmaschinen
- Diese können sich in ihrer Berechnungskraft unterscheiden
- Die Turingmaschine (TM) ist ein universales Berechnungsmodell, da sie alle anderen bekannten Rechenmodelle simulieren kann
- Wir betrachten zunächst Einschränkungen des TM-Modells, die vielfältige praktische Anwendungen haben, wie z.B.
  - endliche Automaten (DFA, NFA)
  - Kellerautomaten (PDA, DPDA) etc.

# Der Algorithmenbegriff

- Der Begriff **Algorithmus** geht auf den persischen Gelehrten **Muhammed Al Chwarizmi** (8./9. Jhd.) zurück
- Ältester bekannter nicht-trivialer Algorithmus: **Euklidischer Algorithmus** zur Berechnung des ggT (300 v. Chr.)
- Von einem Algorithmus wird erwartet, dass er bei jeder zulässigen **Problemeingabe** nach endlich vielen Rechenschritten eine korrekte **Ausgabe** liefert
- Eine wichtige Rolle spielen Entscheidungsprobleme, bei denen jede Eingabe nur mit ja oder nein beantwortet wird
- Die (maximale) Anzahl der Rechenschritte bei allen möglichen Eingaben ist nicht beschränkt, d.h. mit wachsender Eingabelänge kann auch die Rechenzeit beliebig anwachsen
- Die Beschreibung eines Algorithmus muss jedoch endlich sein
- Problemeingaben können Zahlen, Formeln, Graphen etc. sein
- Diese werden über einem Eingabealphabet  $\Sigma$  kodiert

## Definition

- Ein **Alphabet** ist eine geordnete endliche Menge

$$\Sigma = \{a_1, \dots, a_m\}, \quad m \geq 1$$

von **Zeichen**  $a_i$

- Eine Folge  $x = x_1 \dots x_n \in \Sigma^n$  von Zeichen heißt **Wort**
- Die **Länge** von  $x = x_1 \dots x_n \in \Sigma^n$  ist  $n$  und wird mit  $|x|$  bezeichnet
- Die Menge aller Wörter über  $\Sigma$  ist

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$$

- Das (einzige) Wort der Länge  $n = 0$  ist das **leere Wort**, welches wir mit  $\varepsilon$  bezeichnen, d.h.  $\Sigma^0 = \{\varepsilon\}$
- Jede Teilmenge  $L \subseteq \Sigma^*$  heißt **Sprache** über dem Alphabet  $\Sigma$

## Beispiel

- Sprachen über  $\Sigma$  sind beispielsweise  $\emptyset$ ,  $\Sigma^*$ ,  $\Sigma$  und  $\{\varepsilon\}$
- $\emptyset$  enthält keine Wörter und heißt **leere Sprache**
- $\Sigma^*$  enthält dagegen alle Wörter über  $\Sigma$
- $\Sigma$  enthält alle Wörter über  $\Sigma$  der Länge 1
- $\{\varepsilon\}$  enthält nur das leere Wort, ist also einelementig
- Sprachen, die genau ein Wort enthalten, werden auch als **Singletonsprachen** bezeichnet
- in der Informatik spielen Programmiersprachen eine wichtige Rolle

- Da Sprachen Mengen sind, können wir sie bzgl. Inklusion vergleichen
- Zum Beispiel gilt  $\emptyset \subseteq \{\varepsilon\} \subseteq \Sigma^*$
- Wir können Sprachen auch vereinigen, schneiden und komplementieren
- Seien  $A$  und  $B$  Sprachen über  $\Sigma$ . Dann ist
  - $A \cap B = \{x \in \Sigma^* \mid x \in A \wedge x \in B\}$  der **Schnitt** von  $A$  und  $B$ ,
  - $A \cup B = \{x \in \Sigma^* \mid x \in A \vee x \in B\}$  die **Vereinigung** von  $A$  und  $B$ , und
  - $\overline{A} = \{x \in \Sigma^* \mid x \notin A\}$  das **Komplement** von  $A$

# Konkatenation von Wörtern

## Definition

Seien  $x = x_1 \dots x_n$  und  $y = y_1 \dots y_m$  Wörter. Dann wird das Wort

$$x \circ y = x_1 \dots x_n y_1 \dots y_m$$

als **Konkatenation** von  $x$  und  $y$  bezeichnet. Für  $x \circ y$  schreiben wir auch einfach  $xy$ .

## Beispiel

- Für  $x = aba$  und  $y = abab$  erhalten wir  $xy = abaabab$  und  $yx = abababa$
- Die Konkatenation ist also nicht kommutativ
- Allerdings ist  $\circ$  assoziativ, d.h. es gilt  $x(yz) = (xy)z$   
Daher können wir hierfür auch einfach  $xyz$  schreiben
- Es gibt auch ein neutrales Element, da  $x\varepsilon = \varepsilon x = x$  ist
- Eine algebraische Struktur  $(M, \square, e)$  mit einer assoziativen Operation  $\square : M \times M \rightarrow M$  und einem neutralen Element  $e$  heißt **Monoid**
- $(\Sigma^*, \circ, \varepsilon)$  ist also ein Monoid

Neben den Mengenoperationen Schnitt, Vereinigung und Komplement gibt es auch spezielle Sprachoperationen

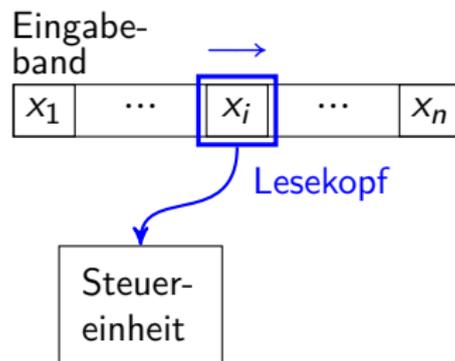
## Definition

- Das **Produkt** (**Verkettung**, **Konkatenation**) der Sprachen  $A$  und  $B$  ist
$$AB = \{xy \mid x \in A, y \in B\}$$
- Ist  $A = \{x\}$  eine Singletonsprache, so schreiben wir für  $\{x\}B$  auch einfach  $xB$
- Die  **$n$ -fache Potenz**  $A^n$  einer Sprache  $A$  ist induktiv definiert durch

$$A^n = \begin{cases} \{\varepsilon\}, & n = 0, \\ A^{n-1}A, & n > 0 \end{cases}$$

- Die **Sternhülle** von  $A$  ist  $A^* = \bigcup_{n \geq 0} A^n$
- Die **Plushülle** von  $A$  ist  $A^+ = \bigcup_{n \geq 1} A^n = AA^*$

- Ein einfaches Rechenmodell zum Erkennen von Sprachen ist der endliche Automat:



- Ein endlicher Automat
  - nimmt zu jedem Zeitpunkt genau einen von endlich vielen Zuständen an
  - macht bei Eingaben der Länge  $n$  genau  $n$  Rechenschritte und
  - liest in jedem Schritt genau ein Eingabezeichen

## Definition

- Ein **endlicher Automat** (kurz: **DFA**; *Deterministic Finite Automaton*) wird durch ein 5-Tupel  $M = (Z, \Sigma, \delta, q_0, E)$  beschrieben, wobei
  - $Z \neq \emptyset$  eine **endliche** Menge von **Zuständen**
  - $\Sigma$  das **Eingabealphabet**,
  - $\delta: Z \times \Sigma \rightarrow Z$  die **Überföhrungsfunktion**
  - $q_0 \in Z$  der **Startzustand** und
  - $E \subseteq Z$  die Menge der **Endzustände** ist
- Die von  $M$  **akzeptierte** oder **erkannte Sprache** ist

$$L(M) = \left\{ x_1 \dots x_n \in \Sigma^* \mid \begin{array}{l} \text{es gibt } q_1, \dots, q_{n-1} \in Z, q_n \in E \text{ mit} \\ \delta(q_i, x_{i+1}) = q_{i+1} \text{ f\u00fcr } i = 0, \dots, n-1 \end{array} \right\}$$

- Eine Zustandsfolge  $q_0, q_1, \dots, q_n$  hei\u00dft **Rechnung** von  $M(x_1 \dots x_n)$ , falls  $\delta(q_i, x_{i+1}) = q_{i+1}$  f\u00fcr  $i = 0, \dots, n-1$  gilt
- Sie hei\u00dft **akzeptierend**, falls  $q_n \in E$  ist, und andernfalls **verwerfend**

## Frage

Welche Sprachen lassen sich durch endliche Automaten erkennen und welche nicht?

## Definition

Eine von einem DFA akzeptierte Sprache wird als **regulär** bezeichnet. Die zugehörige Sprachklasse ist

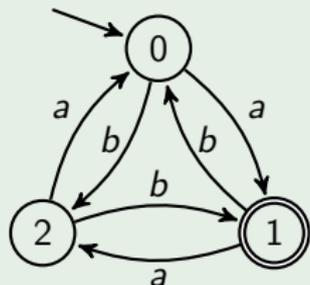
$$\text{REG} = \{L(M) \mid M \text{ ist ein DFA}\}$$

## Beispiel

Sei  $M_3 = (Z, \Sigma, \delta, 0, E)$  ein DFA mit  $Z = \{0, 1, 2\}$ ,  $\Sigma = \{a, b\}$ ,  $E = \{1\}$  und der Überföhrungsfunktion

$\delta$	0	1	2
a	1	2	0
b	2	0	1

Graphische Darstellung:



Endzustände werden durch einen doppelten Kreis und der Startzustand wird durch einen Pfeil gekennzeichnet

Frage: Welche Wörter akzeptiert  $M_3$ ?

- Ist  $w_1 = aba \in L(M_3)$ ? Ja (akzeptierende Rechnung: 0, 1, 0, 1)
- Ist  $w_2 = abba \in L(M_3)$ ? Nein (verwerfende Rechnung: 0, 1, 0, 2, 0)

### Behauptung

Die von  $M_3$  erkannte Sprache ist

$$L(M_3) = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}, \text{ wobei}$$

- $\#_a(x)$  die Anzahl der Vorkommen von  $a$  in  $x$  bezeichnet und
- $i \equiv_m j$  (in Worten:  $i$  ist kongruent zu  $j$  modulo  $m$ ) bedeutet, dass  $i - j$  durch  $m$  teilbar ist

### Beweis der Behauptung durch Induktion über die Länge von $x$

Wir betrachten zunächst das Erreichbarkeitsproblem für DFAs

## Frage

Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA und sei  $x = x_1 \dots x_n \in \Sigma^*$ . Welchen Zustand erreicht  $M$  nach Lesen der Eingabe  $x$ ?

## Antwort

- nach 0 Schritten: den Startzustand  $q_0$
- nach 1 Schritt: den Zustand  $\delta(q_0, x_1)$
- nach 2 Schritten: den Zustand  $\delta(\delta(q_0, x_1), x_2)$
- $\vdots$
- nach  $n$  Schritten: den Zustand  $\delta(\dots \delta(\delta(q_0, x_1), x_2), \dots x_n)$

# Das Erreichbarkeitsproblem für DFAs

## Definition

- Bezeichne  $\hat{\delta}(q, x)$  denjenigen Zustand, in dem sich  $M$  nach Lesen von  $x$  befindet, wenn  $M$  im Zustand  $q$  gestartet wird
- Dann können wir die Funktion

$$\hat{\delta} : Z \times \Sigma^* \rightarrow Z$$

induktiv über die Länge von  $x$  wie folgt definieren:

Für  $q \in Z$ ,  $x \in \Sigma^*$  und  $a \in \Sigma$  sei

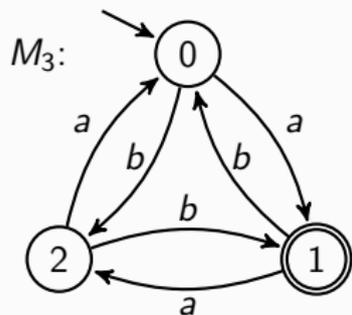
$$\begin{aligned}\hat{\delta}(q, \varepsilon) &= q, \\ \hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a)\end{aligned}$$

- Die von  $M$  erkannte Sprache lässt sich nun elegant durch

$$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in E\}$$

beschreiben

## DFAs beherrschen Modulare Arithmetik



## Behauptung

Für alle  $x \in \{a, b\}^*$  gilt:

$$x \in L(M_3) \Leftrightarrow \#_a(x) - \#_b(x) \equiv_3 1$$

## Beweis

- 1 ist der einzige Endzustand von  $M$
- Daher ist  $L(M_3) = \{x \in \{a, b\}^* \mid \hat{\delta}(0, x) = 1\}$
- Obige Behauptung ist also äquivalent zu

$$\forall x \in \{a, b\}^*: \hat{\delta}(0, x) = 1 \Leftrightarrow \#_a(x) - \#_b(x) \equiv_3 1$$

- Folglich reicht es, für alle  $x \in \{a, b\}^*$  folgende Kongruenz zu zeigen:

$$\hat{\delta}(0, x) \equiv_3 \#_a(x) - \#_b(x)$$

**Induktionsbehauptung:** Für alle  $x \in \{a, b\}^n$  gilt  $\hat{\delta}(0, x) \equiv_3 \#_a(x) - \#_b(x)$

**Induktionsanfang ( $n = 0$ ):** klar, da  $\hat{\delta}(0, \varepsilon) = \#_a(\varepsilon) - \#_b(\varepsilon) = 0$  ist

**Induktionsschritt ( $n \rightsquigarrow n + 1$ ):** Sei  $x = x_1 \dots x_{n+1} \in \{a, b\}^{n+1}$  gegeben

- Nach Induktionsvoraussetzung (IV) gilt für  $x' = x_1 \dots x_n$ :

$$\hat{\delta}(0, x') \equiv_3 \#_a(x') - \#_b(x')$$

- Zudem gilt

$$\begin{aligned} \delta(i, x_{n+1}) &\equiv_3 \begin{cases} i + 1, & x_{n+1} = a \\ i - 1, & x_{n+1} = b \end{cases} \\ &= i + \#_a(x_{n+1}) - \#_b(x_{n+1}) \end{aligned} \quad (*)$$

- Somit folgt

$$\begin{aligned} \hat{\delta}(0, x) &= \delta(\hat{\delta}(0, x'), x_{n+1}) \\ &\equiv_3 \hat{\delta}(0, x') + \#_a(x_{n+1}) - \#_b(x_{n+1}) \quad (*) \\ &\equiv_3 \#_a(x') - \#_b(x') + \#_a(x_{n+1}) - \#_b(x_{n+1}) \quad (IV) \\ &\equiv_3 \#_a(x) - \#_b(x) \end{aligned}$$

# Singletonsprachen sind regulär

## Vereinbarung

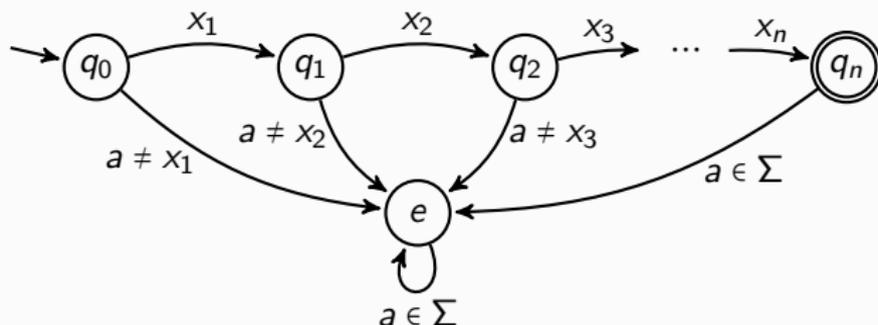
Für das Folgende sei  $\Sigma = \{a_1, \dots, a_m\}$  ein fest gewähltes Alphabet.

## Beobachtung 1

Alle Sprachen, die nur ein Wort  $x = x_1 \dots x_n \in \Sigma^*$  enthalten, sind regulär.

## Beweis

Folgender DFA  $M$  erkennt die Sprache  $L(M) = \{x\}$ :



# REG ist unter Komplement abgeschlossen

## Beobachtung 2

Ist  $L \in \text{REG}$ , so ist auch die Sprache  $\bar{L} = \Sigma^* \setminus L$  regulär.

## Beweis

- Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA mit  $L(M) = L$ .
- Dann wird das Komplement  $\bar{L}$  von  $L$  von dem DFA  $\bar{M} = (Z, \Sigma, \delta, q_0, Z \setminus E)$  akzeptiert. □

## Definition

Für eine Sprachklasse  $\mathcal{C}$  bezeichne  $\text{co-}\mathcal{C}$  die Klasse  $\{\bar{L} \mid L \in \mathcal{C}\}$  aller Komplemente von Sprachen in  $\mathcal{C}$ .

## Korollar

$\text{co-REG} = \text{REG}$ .

## Beobachtung 3

Sind  $L_1, L_2 \in \text{REG}$ , so ist auch die Sprache  $L_1 \cap L_2$  regulär.

## Beweis

- Seien  $M_i = (Z_i, \Sigma, \delta_i, q_i, E_i)$ ,  $i = 1, 2$ , DFAs mit  $L(M_i) = L_i$ .
- Dann wird der Schnitt  $L_1 \cap L_2$  von dem DFA

$$M = (Z_1 \times Z_2, \Sigma, \delta, (q_1, q_2), E_1 \times E_2)$$

mit

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

erkannt.

- $M$  wird auch als **Kreuzproduktautomat** bezeichnet. □

## Beobachtung 4

Die Vereinigung  $L_1 \cup L_2$  von regulären Sprachen  $L_1$  und  $L_2$  ist regulär.

## Beweis

Es gilt  $L_1 \cup L_2 = \overline{(\overline{L_1} \cap \overline{L_2})}$ . □

## Frage

Wie sieht der zugehörige DFA aus?

## Antwort

$$M' = (Z_1 \times Z_2, \Sigma, \delta, (q_1, q_2), (E_1 \times Z_2) \cup (Z_1 \times E_2)).$$

# Abschlusseigenschaften von Sprachklassen

## Definition

- Ein ( $k$ -stelliger) Sprachoperator ist eine Abbildung  $op$ , die  $k$  Sprachen  $L_1, \dots, L_k$  auf eine Sprache  $op(L_1, \dots, L_k)$  abbildet.
- Eine Sprachklasse  $\mathcal{K}$  heißt unter  $op$  abgeschlossen, wenn gilt:  
$$L_1, \dots, L_k \in \mathcal{K} \Rightarrow op(L_1, \dots, L_k) \in \mathcal{K}.$$
- Der Abschluss von  $\mathcal{K}$  unter  $op$  ist die (bzgl. Inklusion) kleinste Sprachklasse  $\mathcal{K}'$ , die  $\mathcal{K}$  enthält und unter  $op$  abgeschlossen ist.

## Beispiel

- Der 2-stellige Schnittoperator  $\cap$  bildet  $L_1$  und  $L_2$  auf  $L_1 \cap L_2$  ab.
- Der Abschluss der Singletonsprachen unter  $\cap$  besteht aus allen Singletonsprachen und der leeren Sprache.
- Der Abschluss der Singletonsprachen unter  $\cup$  besteht aus allen nichtleeren endlichen Sprachen.
- Der Abschluss der Singletonsprachen unter  $\cap$ ,  $\cup$  und Komplement besteht aus allen endlichen und co-endlichen Sprachen.