

Vorlesungsskript  
Einführung in die Theoretische  
Informatik

Wintersemester 2015/16

Prof. Dr. Johannes Köbler  
Humboldt-Universität zu Berlin  
Lehrstuhl Komplexität und Kryptografie

15. Oktober 2015

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Reguläre Sprachen</b>	<b>2</b>
2.1	Endliche Automaten . . . . .	2

# 1 Einleitung

Rechenmaschinen spielen in der Informatik eine zentrale Rolle. In dieser Vorlesung beschäftigen wir uns mit mathematischen Modellen für Maschinentypen von unterschiedlicher Berechnungskraft. Unter anderem lernen wir das Rechenmodell der Turingmaschine (TM) kennen, mit dem sich alle anderen Rechenmodelle simulieren lassen. Ein weiteres wichtiges Thema der Vorlesung ist die Frage, welche Probleme algorithmisch lösbar sind und wo die Grenzen der Berechenbarkeit verlaufen.

Schließlich untersuchen wir die Komplexität von algorithmischen Problemen, indem wir den benötigten Rechenaufwand möglichst gut nach oben und unten abschätzen. Eine besondere Rolle spielen hierbei die NP-vollständigen Probleme, deren Komplexität bis heute offen ist.

## Themen der Vorlesung

- Welche Rechenmodelle sind für bestimmte Aufgaben adäquat? (Automatentheorie)
- Welche Probleme sind lösbar? (Berechenbarkeitstheorie)
- Welcher Aufwand ist zur Lösung eines algorithmischen Problems nötig? (Komplexitätstheorie)

In den theoretisch orientierten Folgeveranstaltungen wird es dagegen um folgende Themen gehen.

## Thema der Vorlesung Algorithmen und Datenstrukturen

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

## Thema der Vorlesung Logik in der Informatik

- Mathematische Grundlagen der Informatik, Beweise führen, Modellierung (Aussagenlogik, Prädikatenlogik)

Der Begriff *Algorithmus* geht auf den persischen Gelehrten **Muhammed Al Chwarizmi** (8./9. Jhd.) zurück. Der älteste bekannte nicht-triviale Algorithmus ist der nach *Euklid* benannte Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen (300 v. Chr.). Von einem Algorithmus wird erwartet, dass er jede *Problemeingabe* nach endlich vielen Rechenschritten löst (etwa durch Produktion einer *Ausgabe*). Eine wichtige Rolle spielen Entscheidungsprobleme, bei denen jede Eingabe nur mit ja oder nein beantwortet wird. Problemeingaben können Zahlen, Formeln, Graphen etc. sein. Diese werden über einem *Eingabealphabet*  $\Sigma$  kodiert.

### Definition 1.

- Ein **Alphabet**  $\Sigma = \{a_1, \dots, a_m\}$  ist eine geordnete Menge von endlich vielen **Zeichen**.
- Eine Folge  $x = x_1 \dots x_n$  von  $n$  Zeichen heißt **Wort** (der **Länge**  $n$ ).
- Die Menge aller Wörter über  $\Sigma$  ist

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

wobei  $\Sigma^n = \{x_1 \dots x_n \mid n \geq 0 \text{ und } x_i \in \Sigma \text{ für } i = 1, \dots, n\}$  alle Wörter der Länge  $n$  enthält.

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

**Beispiel 2.** Sei  $\Sigma$  ein Alphabet. Dann sind  $\emptyset, \Sigma^*, \Sigma$  und  $\{\varepsilon\}$  Sprachen über  $\Sigma$ . Die Sprache  $\emptyset$  enthält keine Wörter und heißt **leere Sprache**. Die Sprache  $\Sigma^*$  enthält dagegen alle Wörter über  $\Sigma$ , während die Sprache  $\Sigma$  alle Wörter über  $\Sigma$  der Länge 1 enthält. Die Sprache

$\{\varepsilon\}$  enthält nur das leere Wort, ist also einelementig. Einelementige Sprachen werden auch als **Singleton-Sprachen** bezeichnet.

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

- $A \cap B = \{x \in \Sigma^* \mid x \in A, x \in B\}$  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
- $\bar{A} = \{x \in \Sigma^* \mid x \notin A\}$  das **Komplement** von  $A$ .

Neben den Mengenoperationen gibt es auch spezielle Sprachoperationen.

**Definition 3.**

- 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

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

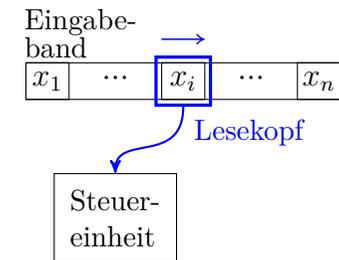
- Die **Sternhülle  $A^*$**  von  $A$  ist  $A^* = \bigcup_{n \geq 0} A^n$ .
- Die **Plushülle  $A^+$**  von  $A$  ist  $A^+ = \bigcup_{n \geq 1} A^n = AA^*$ .

## 2 Reguläre Sprachen

Wir betrachten zunächst Einschränkungen des TM-Modells, die vielfältige praktische Anwendungen haben, wie z.B. endliche Automaten (DFA, NFA), Kellerautomaten (PDA, DPDA) etc.

### 2.1 Endliche Automaten

Ein endlicher Automat führt bei einer Eingabe der Länge  $n$  nur  $n$  Rechenschritte aus. Um die gesamte Eingabe lesen zu können, muss der Automat also in jedem Schritt ein Zeichen der Eingabe verarbeiten.



**Definition 4.** 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

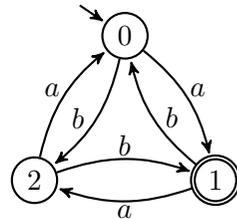
$$L(M) = \left\{ x_1 \dots x_n \in \Sigma^* \mid \begin{array}{l} \text{es gibt } q_1, \dots, q_{n-1} \in Z, q_n \in E \text{ mit} \\ \delta(q_i, x_{i+1}) = q_{i+1} \text{ f\u00fcr } i = 0, \dots, n-1 \end{array} \right\}.$$

$q_0, q_1, \dots, q_n$  heißt **Rechnung** von  $M(x_1 \dots x_n)$ , falls  $\delta(q_i, x_{i+1}) = q_{i+1}$  für  $i = 0, \dots, n-1$  gilt. Sie heißt **akzeptierend**, falls  $q_n \in E$  ist.

**Beispiel 5.** Betrachte den DFA  $M = (Z, \Sigma, \delta, 0, E)$  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:



Der Startzustand wird meist durch einen Pfeil und Endzustände werden durch einen doppelten Kreis gekennzeichnet.  $\triangleleft$

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 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 auch in der Form

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

schreiben.

**Behauptung 6.** Der DFA  $M$  aus Beispiel 5 akzeptiert die Sprache

$$L(M) = \{x \in \Sigma^* \mid \#_a(x) - \#_b(x) \equiv_3 1\},$$

wobei  $\#_a(x)$  die Anzahl der Vorkommen des Zeichens  $a$  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.

