

Vorlesungsskript
Einführung in die Theoretische
Informatik

Wintersemester 2022/23

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

20. Oktober 2022

Inhaltsverzeichnis

| | | |
|----------|------------------------------|----------|
| 1 | Einleitung | 1 |
| 2 | Reguläre Sprachen | 3 |
| 2.1 | Endliche Automaten | 3 |

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)

Die wichtigsten **Lernziele der Vorlesung** sind:

- Überblick über die wichtigsten Rechenmodelle (Automaten) wie z.B.
 - endliche Automaten
 - Kellerautomaten
 - Turingmaschinen
 - Registermaschinen
 - Schaltkreise
- Charakterisierung der Klassen aller mit diesen Rechenmodellen lösbaren 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 mathematische 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 Begriff **Algorithmus** geht auf den persischen Gelehrten **Muhammed Al Chwarizmi** (8./9. Jhd.) zurück. Der älteste bekannte

1 Einleitung

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 für jede zulässige **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** Σ kodiert.

Definition 1.

- Ein **Alphabet** ist eine linear geordnete Menge $\Sigma = \{a_1, \dots, a_m\}$ von $m \geq 1$ **Zeichen** $a_1 < \dots < a_m$.
- Eine Folge $x = x_1 \dots x_n$ von $n \geq 0$ Zeichen $x_i \in \Sigma$ heißt **Wort** der **Länge** n über Σ .
- Die Länge von x wird mit $|x|$ und die Menge aller Wörter der Länge n über Σ wird mit Σ^n bezeichnet.
- Die Menge aller Wörter über Σ ist

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$$

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

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

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

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

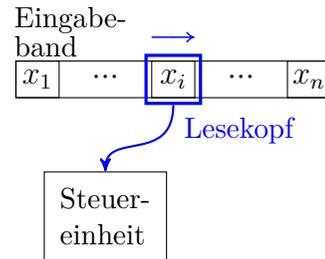
- Die **Sternhülle A^*** einer Sprache A ist $A^* = \bigcup_{n \geq 0} A^n$ und die **Plushülle A^+** von A ist $A^+ = \bigcup_{n \geq 1} A^n = AA^*$.

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**,
- Σ 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

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

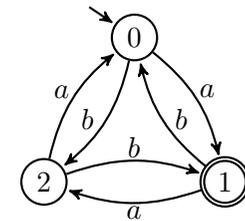
Eine Zustandsfolge q_0, q_1, \dots, q_n hei\u00dft **Rechnung** von $M(x_1 \dots x_n)$, falls $\delta(q_i, x_{i+1}) = q_{i+1}$ f\u00fcr $i = 0, \dots, n-1$ gilt. Sie hei\u00dft **akzeptierend**, falls $q_n \in E$ ist, und andernfalls **verwerfend**. Eine von einem DFA akzeptierte Sprache wird als **regul\u00e4r** bezeichnet. Die zugeh\u00f6rige Sprachklasse ist

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

Beispiel 5. Betrachte den DFA $M = (Z, \Sigma, \delta, 0, E)$ mit $Z = \{0, 1, 2\}$, $\Sigma = \{a, b\}$, $E = \{1\}$ und der \u00dcberf\u00f6hrungsfunktion

| | | | |
|----------|---|---|---|
| δ | 0 | 1 | 2 |
| a | 1 | 2 | 0 |
| b | 2 | 0 | 1 |

Graphische Darstellung:



Der Startzustand wird meist durch einen Pfeil und Endzust\u00e4nde werden durch einen doppelten Kreis gekennzeichnet.

Bei Eingabe $w_1 = aba$ f\u00fchrt M die akzeptierende Rechnung $0, 1, 0, 1$ durch, d.h. $w_1 \in L(M)$. Dagegen verwirft M das Wort $w_2 = abba$ (verwerfende Rechnung: $0, 1, 0, 2, 0$). \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 k\u00f6nnen wir die Funktion

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

induktiv wie folgt definieren. F\u00fcr $q \in Z$, $x \in \Sigma^*$ 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 l\u00e4sst sich nun elegant durch

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

beschreiben.

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.

Beweis. Da M nur den Endzustand 1 hat, ist $L(M) = \{x \in \Sigma^* \mid \hat{\delta}(0, x) = 1\}$, d.h. wir müssen folgende Äquivalenz zeigen:

$$\hat{\delta}(0, x) = 1 \Leftrightarrow \#_a(x) - \#_b(x) \equiv_3 1.$$

Hierzu reicht es, die Kongruenz

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

zu beweisen, wofür wir Induktion über die Länge n von x benutzen.

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

Induktionsschritt ($n \rightsquigarrow n + 1$): Sei $x = x_1 \dots x_{n+1}$ gegeben und sei $i = \hat{\delta}(0, x_1 \dots x_n)$. Nach IV gilt dann

$$i \equiv_3 \#_a(x_1 \dots x_n) - \#_b(x_1 \dots x_n).$$

Wegen $\delta(i, a) \equiv_3 i + 1$ und $\delta(i, b) \equiv_3 i - 1$ folgt daher

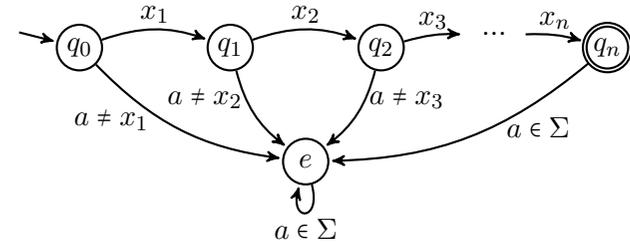
$$\begin{aligned} \delta(i, x_{n+1}) &\equiv_3 i + \#_a(x_{n+1}) - \#_b(x_{n+1}) \\ &\equiv_3 \#_a(x_1 \dots x_n) - \#_b(x_1 \dots x_n) + \#_a(x_{n+1}) - \#_b(x_{n+1}) \\ &= \#_a(x) - \#_b(x). \end{aligned}$$

und somit

$$\hat{\delta}(0, x) = \delta(\hat{\delta}(0, x_1 \dots x_n), x_{n+1}) = \delta(i, x_{n+1}) \equiv_3 \#_a(x) - \#_b(x). \quad \blacksquare$$

Beobachtung 7. Alle Singletonsprachen sind regulär.

Beweis. Für jedes Wort $x = x_1 \dots x_n$ existiert ein DFA M_x mit $L(M_x) = \{x\}$:



Formal ist M_x also das Tupel $(Z, \Sigma, \delta, q_0, E)$ mit $Z = \{q_0, \dots, q_n, e\}$, $E = \{q_n\}$ und der Überföhrungsfunktion

$$\delta(q, a_j) = \begin{cases} q_{i+1}, & q = q_i \text{ für ein } i \text{ mit } 0 \leq i \leq n - 1 \text{ und } a_j = x_{i+1} \\ e, & \text{sonst.} \end{cases}$$

■

Als nächstes betrachten wir Abschlusseigenschaften der Sprachklasse REG.

Definition 8. 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.

Beispiel 9. Der Schnittoperator \cap bildet zwei Sprachen L_1 und L_2 auf die Sprache $L_1 \cap L_2$ ab. ◁

Definition 10. 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 11. Der Abschluss der Singletonsprachen unter \cap besteht aus allen Singletonsprachen und der leeren Sprache.

Der Abschluss der Singletonsprachen unter \cup besteht aus allen nicht-leeren endlichen Sprachen.

Der Abschluss der Singletonsprachen unter \cap , \cup und Komplement besteht aus allen endlichen und co-endlichen Sprachen.* \triangleleft

Definition 12. Für eine Sprachklasse \mathcal{C} bezeichne $co\text{-}\mathcal{C}$ die Klasse $\{\bar{L} \mid L \in \mathcal{C}\}$ aller Komplemente von Sprachen in \mathcal{C} .

Es ist klar, dass \mathcal{C} genau dann unter Komplementbildung abgeschlossen ist, wenn $co\text{-}\mathcal{C} = \mathcal{C}$ ist.

Beobachtung 13. Mit $L_1, L_2 \in \text{REG}$ sind auch die Sprachen $\bar{L}_1 = \Sigma^* \setminus L_1$, $L_1 \cap L_2$ und $L_1 \cup L_2$ regulär.

Beweis. Sind $M_i = (Z_i, \Sigma, \delta_i, q_0, E_i)$, $i = 1, 2$, DFAs mit $L(M_i) = L_i$, so akzeptiert der DFA

$$\bar{M}_1 = (Z_1, \Sigma, \delta_1, q_0, Z_1 \setminus E_1)$$

das Komplement \bar{L}_1 von L_1 . Der Schnitt $L_1 \cap L_2$ von L_1 und L_2 wird dagegen von dem DFA

$$M = (Z_1 \times Z_2, \Sigma, \delta, (q_0, q_0), E_1 \times E_2)$$

mit

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

akzeptiert (M wird auch **Kreuzproduktautomat** genannt). Wegen $L_1 \cup L_2 = \overline{\bar{L}_1 \cap \bar{L}_2}$ ist dann aber auch die Vereinigung von L_1 und L_2 regulär. (Wie sieht der zugehörige DFA aus?) ■

Aus Beobachtung 13 folgt, dass alle endlichen und alle co-endlichen Sprachen regulär sind. Da die in Beispiel 5 betrachtete Sprache weder endlich noch co-endlich ist, haben wir damit allerdings noch nicht alle regulären Sprachen erfasst.

Es stellt sich die Frage, ob REG neben den mengentheoretischen Operationen Schnitt, Vereinigung und Komplement unter weiteren Operationen wie etwa Produkt oder Sternhülle abgeschlossen ist. Im übernächsten Abschnitt werden wir sehen, dass die Klasse REG als der Abschluss der endlichen Sprachen unter Vereinigung, Produkt und Sternhülle charakterisierbar (und somit auch unter diesen Operationen abgeschlossen) ist.

Beim Versuch, einen endlichen Automaten für das Produkt $L(M_1)L(M_2)$ zweier regulärer Sprachen zu konstruieren, stößt man auf die Schwierigkeit, den richtigen Zeitpunkt für den Übergang von (der Simulation von) M_1 zu M_2 zu finden. Unter Verwendung eines nichtdeterministischen endlichen Automaten lässt sich dieses Problem jedoch leicht lösen, da dieser den richtigen Zeitpunkt „erraten“ kann. Im nächsten Abschnitt werden wir nachweisen, dass auch nichtdeterministische endliche Automaten nur reguläre Sprachen erkennen können.

Eine Sprache $L \subseteq \Sigma^$ ist co-endlich, wenn ihr Komplement \bar{L} endlich ist.