# Einführung in die Theoretische Informatik

Johannes Köbler

Institut für Informatik
Humboldt-Universität zu Berlin

WS 2022/23

## Aktuelle Infos unter



- **https://hu.berlin/ethi**

bzw.

- **https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws22/einftheo**

# Übungen

## Anmeldung

- bei Agnes (möglichst bald, Status egal)
- und bei Moodle (u.a. wg. Punktevergabe)
- Mails von Agnes und von Moodle werden standardmäßig an den HU-Account gesendet (bitte regelmäßig checken)

## Ausgabe der Aufgabenblätter und Abgabe der Lösungen

- bei Moodle und auf der VL-Webseite
- neue Blätter erscheinen in der Regel dienstags
- MC-Tests sind bis Montag darauf 23:59 Uhr fällig
- schriftliche Aufgaben sind bis zum Dienstag zwei Wochen nach Ausgabe um 23:59 Uhr in Moodle hochzuladen
- mündliche Aufgaben **können** hochgeladen werden bis einen Tag vor dem eigenen Übungstermin, Details siehe Moodle

# Übungen



## Bearbeitung der Aufgaben

- in Gruppen von zwei bis drei Teilnehmern
- Teilnehmer müssen nicht in der gleichen Übungsgruppe sein
- bitte jede Aufgabe in einer separaten PDF-Datei bearbeiten, da diese getrennt abzugeben sind
- bitte in **jeder** Datei Folgendes angeben:
  - die Namen und HU-Accountnamen der Gruppenteilnehmer
  - den Namen der Abgabegruppe aus Moodle (z.B. AG123)
  - Dateinamen wählen, wie auf dem Übungsblatt angegeben

# Schein, Klausur, Skript



## Scheinkriterien

- Lösen von $\geq 50\%$ der schriftlichen Aufgaben
- Bestehen von 12 von 14 MC-Tests (Blatt 2 bis 15), d.h. jeweils $\geq 50\%$ der Punkte pro Test in Moodle.

## **Klausur**

- **Anfang März**
- **Nachklausur: Anfang April**

## Skript und Folien

- werden wöchentlich ins Netz gestellt (auf der Webseite)

Gibt es zum organisatorischen Ablauf noch Fragen?

# Inhalt der Vorlesung



**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

# Lernziele



- 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

# Maschinenmodelle



- 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

# Alphabet, Wort, Sprache



## Definition

- Ein Alphabet ist eine endliche linear geordnete Menge

  $$\Sigma = \{a_1, \ldots, a_m\}$$

  von $m \geq 1$ Zeichen $a_1 < \cdots < a_m$
- Eine Folge $x = x_1 \ldots x_n$ von $n \geq 0$ Zeichen $x_i \in \Sigma$ heißt Wort der Länge $n$ über $\Sigma$
- Die Länge von $x$ wird mit $|x|$ und die Menge aller Wörter der Länge $n$ über $\Sigma$ wird mit $\Sigma^n$ bezeichnet
- Die Menge aller Wörter über $\Sigma$ ist

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

- 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 $\Sigma$

# Beispiele für Sprachen



## Beispiel

- Sprachen über $\Sigma$ sind beispielsweise $\varnothing, \Sigma^*, \Sigma$ und $\{\varepsilon\}$
- $\varnothing$ 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

# Operationen auf Sprachen



- Da Sprachen Mengen sind, können wir sie bzgl. Inklusion vergleichen
- Zum Beispiel gilt $\varnothing \subseteq \{\varepsilon\} \subseteq \Sigma^*$
- 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

Die Konkatenation von zwei Wörtern $x = x_1 \dots x_n$ und $y = y_1 \dots y_m$ ist das Wort $x \circ y = x_1 \dots x_n y_1 \dots y_m$, das wir auch einfach mit $xy$ bezeichnen

## 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 $\square : M \times M \to M$ und einem neutralen Element $e$ heißt Monoid
- $(\Sigma^*, \circ, \varepsilon)$ ist also ein Monoid

# Spezielle Sprachoperationen



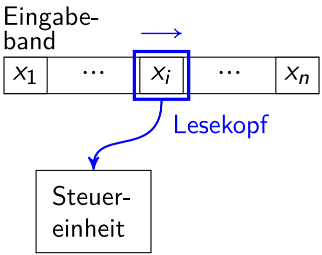
Neben den Mengenoperationen Schnitt, Vereinigung und Komplement gibt es auch spezielle Sprachoperationen

## Definition

- Das Produkt (Verkettung, Konkatenation) von zwei 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 einer Sprache $A$ ist $A^* = \bigcup_{n \geq 0} A^n$
- Die Plushülle einer Sprache $A$ ist $A^+ = \bigcup_{n \geq 1} A^n = AA^*$

# Algorithmische Erkennung von Sprachen

- 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

# Formale Definition eines endlichen Automaten



## Definition

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

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

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

# Die Klasse der regulären Sprachen



### 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

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

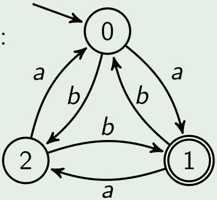
# DFAs beherrschen Modulare Arithmetik

## Beispiel

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

| $\delta$ | 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: $0, 1, 0, 2, 0$)

# DFAs beherrschen Modulare Arithmetik



### Behauptung

Die von $M_3$ erkannte Sprache ist

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

- $\#_a(x)$ 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

# 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 ersten $i$ Zeichen der Eingabe $x$?

## Antwort

- $i = 0$: den Startzustand $q_0$
- $i = 1$: den Zustand $\delta(q_0, x_1)$
- $i = 2$: den Zustand $\delta(\delta(q_0, x_1), x_2)$

  $\vdots$
- $i = n$: den Zustand $\delta(\dots \delta(\delta(q_0, x_1), x_2), \dots x_n)$

# Das Erreichbarkeitsproblem für DFAs

## Definition

- Für $x \in \Sigma^*$ 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

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

  induktiv über die Länge $n$ von $x$ wie folgt definieren:
  Für $q \in Z$, $x \in \Sigma^n$ und $a \in \Sigma$ sei

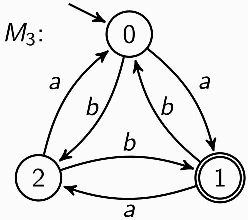
  $$\begin{aligned} n = 0: &\quad \hat{\delta}(q, \varepsilon) = q, \\ n \rightsquigarrow n+1: &\quad \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a) \end{aligned}$$

- 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 die Äquivalenz

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

**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 folgender Behauptung

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

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

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

# DFAs beherrschen Modulare Arithmetik

**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 für alle $q \in Z = \{0,1,2\}$:

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

- Somit folgt

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

□

# Singletonsprachen sind regulär



## Vereinbarung

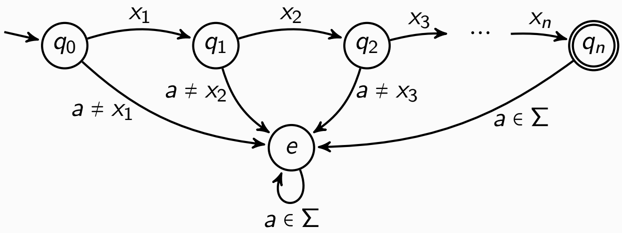
Für das Folgende sei $\Sigma = \{a_1, \ldots, a_m\}$ ein fest gewähltes Alphabet

## Beobachtung 1

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

## Beweis

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




□

# REG ist unter Komplement abgeschlossen

## Beobachtung 2

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

## Beweis

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

## Definition

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

## Korollar

*co*-REG = REG

# REG ist unter Schnitt abgeschlossen



**Beobachtung 3**

Sind $L_1, L_2 \in \mathrm{REG}$, 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 □

# REG ist unter Vereinigung abgeschlossen



## Beobachtung 4

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

## Beweis

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

## Frage

Wie sieht der zugehörige DFA $M'$ aus?

## Antwort

- Seien $M_i = (Z_i, \Sigma, \delta_i, q_i, E_i)$, $i = 1, 2$, DFAs mit $L(M_i) = L_i$
- Dann gilt $L(M') = L_1 \cup L_2$ für den DFA

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

mit $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$ für alle $(p, q) \in Z_1 \times Z_2$ und $a \in \Sigma$

# 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 $\triangleleft$

# REG ist unter Mengenoperationen abgeschlossen



**Korollar**

Die Klasse REG der regulären Sprachen ist unter folgenden Operationen abgeschlossen:

- Komplement
- Schnitt
- Vereinigung

# Wie umfangreich ist REG?

## Folgerung

- Aus den Beobachtungen folgt, dass alle endlichen und alle co-endlichen Sprachen regulär sind
- Da die reguläre Sprache

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

  weder endlich noch co-endlich ist, haben wir damit allerdings noch nicht alle regulären Sprachen erfasst

# Wie umfangreich ist REG?



## Nächstes Ziel

Zeige, dass REG unter Produktbildung und Sternhülle abgeschlossen ist

## Problem

Bei der Konstruktion eines DFA $M$ für das Produkt $L(M_1)L(M_2)$ bereitet es Schwierigkeiten, den richtigen Zeitpunkt für das Ende der Simulation des DFA $M_1$ und den Start der Simulation des DFA $M_2$ zu finden

## Lösungsidee

Ein nichtdeterministischer endlicher Automat (NFA) kann den richtigen Zeitpunkt „raten“

## Verbleibendes Problem

Zeige, dass auch NFAs nur reguläre Sprachen erkennen

# Nichtdeterministische endliche Automaten

## Definition

- Ein nichtdeterministischer endlicher Automat $N = (Z, \Sigma, \Delta, Q_0, E)$ (kurz: NFA; *Nondeterministic Finite Automaton*) ist genau so aufgebaut wie ein DFA, nur dass er
  - eine Menge $Q_0 \subseteq Z$ von Startzuständen hat und
  - die Überführungsfunktion die Form $\Delta : Z \times \Sigma \to \mathcal{P}(Z)$ hat

  Hierbei bezeichnet $\mathcal{P}(Z)$ die Potenzmenge (also die Menge aller Teilmengen) von $Z$; diese wird oft auch mit $2^Z$ bezeichnet
- Die von einem NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ akzeptierte (oder erkannte) Sprache ist

$$L(N) = \left\{ x_1 \dots x_n \in \Sigma^* \;\middle|\; \begin{array}{c} \text{es gibt } q_0 \in Q_0, q_1, \dots, q_{n-1} \in Z, q_n \in E \\ \text{mit } q_{i+1} \in \Delta(q_i, x_{i+1}) \text{ für } i = 0, \dots, n-1 \end{array} \right\}$$

- Eine Zustandsfolge $q_0, \dots, q_n$ heißt Rechnung von $N(x_1 \dots x_n)$, falls $q_0 \in Q_0$ und $q_{i+1} \in \Delta(q_i, x_{i+1})$ für $i = 0, \dots, n-1$ gilt

# Eigenschaften von NFAs



- Ein NFA $N$ kann bei einer Eingabe $x$ also nicht nur eine, sondern mehrere verschiedene Rechnungen parallel ausführen
- Ein Wort $x$ gehört genau dann zu $L(N)$, wenn $N(x)$ mindestens eine akzeptierende Rechnung hat
- Im Gegensatz zu einem DFA, der jede Eingabe zu Ende liest, kann ein NFA $N$ „stecken bleiben"
- Dieser Fall tritt ein, wenn $N$ in einen Zustand $q$ gelangt, in dem er das nächste Eingabezeichen $x_i$ wegen

  $$\Delta(q, x_i) = \varnothing$$

  nicht lesen kann

# Eigenschaften von NFAs

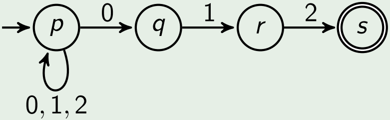


## Beispiel

- Betrachte den NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ mit $Z = \{p, q, r, s\}$, $\Sigma = \{0, 1, 2\}$, $Q_0 = \{p\}$, $E = \{s\}$ und der Überführungsfunktion

| $\Delta$ | $p$ | $q$ | $r$ | $s$ |
|---|---|---|---|---|
| 0 | $\{p, q\}$ | $\varnothing$ | $\varnothing$ | $\varnothing$ |
| 1 | $\{p\}$ | $\{r\}$ | $\varnothing$ | $\varnothing$ |
| 2 | $\{p\}$ | $\varnothing$ | $\{s\}$ | $\varnothing$ |

Graphische Darstellung:




- Ist $w_1 = 012 \in L(N)$? Ja: (akzeptierende Rechnung: $p, q, r, s$)
  Es gibt aber auch verwerfende Rechnungen bei Eingabe $w_1$: $p, p, p, p$
- Ist $w_2 = 021 \in L(N)$? Nein, da es keine akzeptierende Rechnung gibt
- Es gilt $L(N) = \Sigma^* 012 = \{x012 \mid x \in \Sigma^*\}$ ◁

# Ein NFA für das Produkt von regulären Sprachen

## Beobachtung 5

Seien $N_i = (Z_i, \Sigma, \Delta_i, Q_i, E_i)$ NFAs mit $L(N_i) = L_i$ für $i = 1, 2$. Dann wird auch das Produkt $L_1 L_2$ von einem NFA erkannt

## Beweis

- Wir können $Z_1 \cap Z_2 = \varnothing$ annehmen
- Dann gilt $L(N) = L_1 L_2$ für den NFA $N = (Z_1 \cup Z_2, \Sigma, \Delta, Q_1, E)$ mit

$$\Delta(p, a) = \begin{cases} \Delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p, a) \cup \bigcup_{q \in Q_2} \Delta_2(q, a), & p \in E_1, \\ \Delta_2(p, a), & p \in Z_2 \end{cases}$$

und

$$E = \begin{cases} E_2, & Q_2 \cap E_2 = \varnothing, \\ E_1 \cup E_2, & \text{sonst} \end{cases}$$

# Ein NFA für das Produkt von regulären Sprachen

- Dann gilt $L(N) = L_1 L_2$ für den NFA $N = (Z_1 \cup Z_2, \Sigma, \Delta, Q_1, E)$ mit
$$\Delta(p,a) = \begin{cases} \Delta_1(p,a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p,a) \cup \bigcup_{q \in Q_2} \Delta_2(q,a), & p \in E_1, \\ \Delta_2(p,a), & p \in Z_2 \end{cases}$$
und $E = E_2$, falls $Q_2 \cap E_2 = \varnothing$, bzw. $E = E_1 \cup E_2$ sonst

Beweis von $L_1 L_2 \subseteq L(N)$: Seien $x = x_1 \dots x_k \in L_1, y = y_1 \dots y_l \in L_2$ und seien $q_0, \dots, q_k$ und $p_0, \dots, p_l$ akz. Rechnungen von $N_1(x)$ und $N_2(y)$. Dann ist $q_0, \dots, q_k, p_1, \dots, p_l$ eine akz. Rechnung von $N(xy)$, da $q_0 \in Q_1$ und

- $q_i \in \Delta(q_{i-1}, x_i)$ für $i = 1, \dots, k$ ist (wg. $q_i \in \Delta_1(q_{i-1}, x_i)$) und
- im Fall $l = 0$ zudem $q_k \in E$ ist (wg. $q_k \in E_1$ und $p_l \in Q_2 \cap E_2$), sowie
- im Fall $l \geq 1$ zudem $p_1 \in \Delta(q_k, y_1)$ (wg. $q_k \in E_1$, $p_0 \in Q_2$, $p_1 \in \Delta_2(p_0, y_1)$) und $p_j \in \Delta(p_{j-1}, y_j)$ für $j = 2, \dots, l$ (wegen $p_j \in \Delta_2(p_{j-1}, y_j)$) und $p_l \in E_2 \subseteq E$ ist

# Ein NFA für das Produkt von regulären Sprachen

- Dann gilt $L(N) = L_1L_2$ für den NFA $N = (Z_1 \cup Z_2, \Sigma, \Delta, Q_1, E)$ mit

$$\Delta(p,a) = \begin{cases} \Delta_1(p,a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p,a) \cup \bigcup_{q \in Q_2} \Delta_2(q,a), & p \in E_1, \\ \Delta_2(p,a), & p \in Z_2 \end{cases}$$

  und $E = E_2$, falls $Q_2 \cap E_2 = \varnothing$, bzw. $E = E_1 \cup E_2$ sonst

Beweis von $L(N) \subseteq L_1L_2$: Sei $x = x_1 \ldots x_n \in L(N)$ und sei $q_0, \ldots, q_n$ eine akz. Rechnung von $N(x)$. Dann gilt $q_0 \in Q_1$, $q_n \in E$, $q_0, \ldots, q_i \in Z_1$ und $q_{i+1}, \ldots, q_n \in Z_2$ für ein $i \leq n$.

- Im Fall $i < n$ impliziert der Übergang $q_{i+1} \in \Delta(q_i, x_{i+1})$, dass $q_i \in E_1$ und $q_{i+1} \in \Delta_2(q, x_{i+1})$ für ein $q \in Q_2$ ist. Da zudem $q_n \in E \cap Z_2 = E_2$ ist, ist $q_0, \ldots, q_i$ eine akz. Rechnung von $N_1(x_1 \ldots x_i)$ und $q, q_{i+1}, \ldots, q_n$ eine akz. Rechnung von $N_2(x_{i+1} \ldots x_n)$
- Im Fall $i = n$ ist $q_n \in E \cap Z_1$, was $q_n \in E_1$ und $Q_2 \cap E_2 \neq \varnothing$ impliziert. Also ist $q_0, \ldots, q_n$ eine akz. Rechnung von $N_1(x_1 \ldots x_n)$ und $\varepsilon \in L_2$ $\square$

# Ein NFA für die Sternhülle einer regulären Sprache



## Beobachtung 6

Ist $N = (Z, \Sigma, \Delta, Q_0, E)$ ein NFA, so wird auch die Sprache $L(N)^*$ von einem NFA erkannt

## Beweis

Die Sprache $L(N)^*$ wird von dem NFA

$$N^* = (Z \cup \{q_{neu}\}, \Sigma, \Delta^*, Q_0 \cup \{q_{neu}\}, E \cup \{q_{neu}\})$$

mit

$$\Delta^*(p,a) = \begin{cases} \Delta(p,a), & p \in Z \setminus E, \\ \Delta(p,a) \cup \bigcup_{q \in Q_0} \Delta(q,a), & p \in E, \\ \varnothing, & p = q_{neu} \end{cases}$$

erkannt

□

# Überblick



## Ziel

Zeige, dass REG unter Produktbildung und Sternhülle abgeschlossen ist

## Problem

Bei der Konstruktion eines DFA für das Produkt $L_1 L_2$ bereitet es Schwierigkeiten, den richtigen Zeitpunkt für den Übergang von (der Simulation von) $M_1$ zu $M_2$ zu finden

## Lösungsidee (bereits umgesetzt)

Ein nichtdeterministischer Automat (NFA) kann den richtigen Zeitpunkt für den Übergang „raten“

## Noch zu zeigen

NFAs erkennen genau die regulären Sprachen

# NFAs erkennen genau die regulären Sprachen



**Satz (Rabin und Scott)**

$\mathrm{REG} = \{L(N) \mid N \text{ ist ein NFA}\}$

**Beweis von $\mathrm{REG} \subseteq \{L(N) \mid N \text{ ist ein NFA}\}$**

Diese Inklusion ist klar, da jeder DFA $M = (Z, \Sigma, \delta, q_0, E)$ in einen äquivalenten NFA

$$N = (Z, \Sigma, \Delta, Q_0, E)$$

transformiert werden kann, indem wir $\Delta(q, a) = \{\delta(q, a)\}$ und $Q_0 = \{q_0\}$ setzen. $\square$

Für die umgekehrte Inklusion ist das Erreichbarkeitsproblem für NFAs von zentraler Bedeutung

# Das Erreichbarkeitsproblem für NFAs



## Frage

Sei $N = (Z, \Sigma, \Delta, Q_0, E)$ ein NFA und sei $x = x_1 \ldots x_n$ eine Eingabe. Welche Zustände sind in $i$ Schritten erreichbar?

## Antwort

- in 0 Schritten: alle Zustände in $Q_0$
- nach einem Schritt: alle Zustände in

$$Q_1 = \bigcup_{q \in Q_0} \Delta(q, x_1)$$

- nach $i$ Schritten: alle Zustände in

$$Q_i = \bigcup_{q \in Q_{i-1}} \Delta(q, x_i)$$

# Simulation von NFAs durch DFAs



## Idee

- Wir können einen NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ durch einen DFA $M = (Z', \Sigma, \delta, q_0', E')$ simulieren, der in seinem Zustand die Information speichert, in welchen Zuständen sich $N$ momentan befinden könnte
- Die Zustände von $M$ sind also Teilmengen $Q$ von $Z$ (d.h. $Z' = \mathcal{P}(Z)$) mit $Q_0$ als Startzustand (d.h. $q_0' = Q_0$) und der Endzustandsmenge

$$E' = \{Q \subseteq Z \mid Q \cap E \neq \emptyset\}$$

- Die Überführungsfunktion $\delta : \mathcal{P}(Z) \times \Sigma \to \mathcal{P}(Z)$ von $M$ berechnet dann für einen Zustand $Q \subseteq Z$ und ein Zeichen $a \in \Sigma$ die Menge

$$\delta(Q, a) = \bigcup_{q \in Q} \Delta(q, a)$$

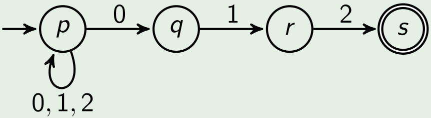
  aller Zustände, die $N$ ausgehend von einem beliebigen Zustand $q \in Q$ bei Lesen des Zeichens $a$ erreichen kann
- $M$ wird auch als der zu $N$ gehörige Potenzmengenautomat bezeichnet

# Simulation von NFAs durch DFAs



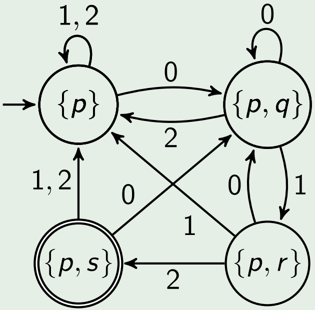
## Beispiel

- Betrachte den NFA $N$




- Ausgehend von $Q_0 = \{p\}$ liefert $\delta$ dann die folgenden Werte:

| $\delta$ | 0 | 1 | 2 |
|---|---|---|---|
| $\{p\}$ | $\{p, q\}$ | $\{p\}$ | $\{p\}$ |
| $\{p, q\}$ | $\{p, q\}$ | $\{p, r\}$ | $\{p\}$ |
| $\{p, r\}$ | $\{p, q\}$ | $\{p\}$ | $\{p, s\}$ |
| $\{p, s\}$ | $\{p, q\}$ | $\{p\}$ | $\{p\}$ |



# Simulation von NFAs durch DFAs

## Bemerkung

- Im obigen Beispiel werden für die Konstruktion des Potenzmengen-automaten nur 4 der insgesamt

$$|\mathcal{P}(Z)| = 2^{|Z|} = 2^4 = 16$$

  Zustände benötigt, da die übrigen 12 Zustände nicht erreichbar sind (hierbei bezeichnet $|A|$ die Mächtigkeit einer Menge $A$)
- Es gibt jedoch Beispiele, bei denen alle $2^{|Z|}$ Zustände benötigt werden (siehe Übungen)

## NFAs erkennen genau die regulären Sprachen



**Beweis von $\{L(N) \mid N \text{ ist ein NFA}\} \subseteq \text{REG}$**

- Sei $N = (Z, \Sigma, \Delta, Q_0, E)$ ein NFA und sei $M = (\mathcal{P}(Z), \Sigma, \delta, Q_0, E')$ der zugehörige Potenzmengenautomat mit $\delta(Q, a) = \bigcup_{q \in Q} \Delta(q, a)$ und $E' = \{Q \subseteq Z \mid Q \cap E \neq \emptyset\}$
- Dann folgt die Korrektheit von $M$ mittels folgender Behauptung, die wir auf der nächsten Folie beweisen

**Behauptung**

$\hat{\delta}(Q_0, x)$ enthält genau die von $N$ nach Lesen von $x$ erreichbaren Zustände

- Für alle Wörter $x \in \Sigma^*$ gilt

$$
\begin{aligned}
x \in L(N) \quad &\Leftrightarrow \quad N \text{ kann nach Lesen von } x \text{ einen Endzustand erreichen} \\
&\overset{\text{Beh.}}{\Leftrightarrow} \quad \hat{\delta}(Q_0, x) \cap E \neq \emptyset \\
&\Leftrightarrow \quad \hat{\delta}(Q_0, x) \in E' \\
&\Leftrightarrow \quad x \in L(M)
\end{aligned}
$$

$\square$