Theoretische Informatik: Automaten und formale Sprachen

Prof. Dr. Sibylle Schwarz
HTWK Leipzig, Fakultät IM
Gustav-Freytag-Str. 42a, 04277 Leipzig
Zimmer Z 411 (Zuse-Bau)
https://informatik.htwk-leipzig.de/schwarz
sibylle.schwarz@htwk-leipzig.de

Sommersemester 2025

Einordnung der Theoretischen Informatik

Informatik Lehre von der Darstellung und Verarbeitung von Information durch Algorithmen

Teilgebiete der Informatik:

theoretisch
- Sprachen zur Formulierung von Information und Algorithmen,
- Möglichkeiten und Grenzen der maschinellen Berechenbarkeit,
- Grundlagen für technische und praktische (und angewandte) Informatik

technisch
- maschinelle Darstellung von Information
- Mittel zur Ausführung von Algorithmen

(Rechnerarchitektur, Hardware-Entwurf, Netzwerk, ...)

praktisch Entwurf und Implementierung von Algorithmen (Betriebssysteme, Compilerbau, SE, ...)

angewandt Anwendung von Algorithmen (Text- und Bildverarbeitung, Datenbanken, KI, Medizin-, Bio-, Wirtschafts-, Medieninformatik, ...)

Teilgebiete der theoretischen Informatik

Formale Sprachen

- Repräsentation von Informationen und Aufgaben in maschinenlesbarer Form (Mensch-Maschine-Kommunikation)
- Ausdrucksstärke und Flexibilität von Programmiersprachen
- Übersetzung von Programmiersprachen (z.B. in ausführbaren Code)

Maschinenmodelle (Automaten)

- Möglichkeiten und Grenzen verschiedener Modelle zur (maschinellen) Ausführung von Algorithmen

Berechenbarkeitstheorie (Master)

- Welche Aufgaben sind überhaupt algorithmisch (mit Hilfe verschiedener Maschinenmodelle) lösbar? Auch negative Antworten sind sehr hilfreich (sparen Aufwand für ungeeignete Lösungsansätze)

Komplexitätstheorie (Master)

- Welche Aufgaben sind mit beschränkten Ressourcen (z.B. Zeit, Speicherplatz) lösbar?
- Für welche Aufgaben können schnelle Algorithmen existieren?

Prinzipien der theoretischen Informatik

ältester Zweig der Informatik (lange vor dem ersten Computer)

Mathematische Prinzipien:

- Abstraktion
  - ermöglicht verallgemeinerte Aussagen und breit einsetzbare Verfahren,
  - Ergebnisse und Verfahren oft nicht sofort praktisch anwendbar, müssen auf spezielle Situationen angepasst werden.
- Beweisbarkeit
  - erfordert präzise Modellierung der Aufgabe
  - Nachweis der Korrektheit von Hard- und Software (Tests können dies nicht !)

Wissen aus der theoretischen Informatik

- veraltet über viele Jahre kaum
- Grundlage für Verständnis von (schnelllebigem) Spezialwissen, z.B. konkrete Programmiersprachen, Domain-spezifische Sprachen, Transformationen verschiedener Darstellungen

Aus der Modulbeschreibung

C993 Theoretische Informatik: Automaten und formale Sprachen

Arbeitsaufwand: Präsenzzeit 56 h (= 2 SWS V + 2 SWS S)
Vor- und Nachbereitungszeit 94 h (≈ 6 h je Woche)

Voraussetzungen: anwendungsbereite Kenntnisse auf den Gebieten Modellierung, Logik, Algorithmen und Datenstrukturen, Aufwandsabschätzungen

Lernziele: Die Studierenden sind in der Lage, wichtige Klassen formaler Sprachen als Grundlage von Programmier- und Beschreibungssprachen einzuordnen und kennen die wesentlichen Eigenschaften der Sprachklassen.
Sie kennen die entsprechenden abstrakten Maschinenmodelle und Algorithmen und können sie zur Darstellung und Lösung praktischer Aufgabenstellungen einsetzen.
Die Studierenden wissen, dass nicht jedes formal darstellbare Problem algorithmisch lösbar ist.

Inhalt der Lehrveranstaltung

- Formale Sprachen
  - Wiederholung: Alphabet, Wort, Sprache, Operationen darauf
  - reguläre Ausdrücke
  - Wortersetzung
  - Grammatiken
  - Chomsky-Hierarchie
- Maschinenmodelle
  - Endliche Automaten
  - Kellerautomaten
  - Turing-Maschinen
- Berechenbarkeit (Ausblick auf Master-Modul)
  - berechenbare Funktionen
  - Berechnungsmodelle
  - These von Church
  - algorithmische Entscheidbarkeit / Unentscheidbarkeit
- Komplexität (Ausblick auf Master-Modul)

jeweils mit vielen Beispielen

Lehrveranstaltungen

Folien, Übungsserien, aktuelle Informationen unter
`https://informatik.htwk-leipzig.de/schwarz/lehre/ss25/tib`

Vorlesung	donnerstags / freitags	(2 h / Woche)
Selbststudium	(Hausaufgaben):	(6 h / Woche)
schriftliche	Übungsserien (ca. zu jeder Vorlesung) Besprechung in der folgenden Übung	
Autotool	jeweils nach der Vorlesung bis Sonntag der nächsten Woche	
Seminar	Besprechung der Übungsserien (incl. Selbsstest) (Vorrechnen der Musterlösungen)	

Prüfung

Prüfungsvorleistungen:

- $\geq$ 3 Kurzvorträge zu schriftlichen Übungsaufgaben (Übungen) und
- $\geq$ 50% aller Punkte für Autotool-Pflichtaufgaben

Prüfung: Klausur 90 min
Aufgabentypen ähnlich Übungsaufgaben
(Hilfsmittel: beidseitig handbeschriebenes A4-Blatt)

Formale Sprachen

Syntax natürlicher Sprachen:

- Rechtschreibung: korrekte Wörter
- Grammatik: Aufbau korrekter Sätze

Definition von Programmiersprachen:

Syntax	Form der Sprachelemente
Semantik	Bedeutung der Sprachelemente und -strukturen
Pragmatik	Regeln zur zweckmäßigen Anwendung

Syntax von Programmiersprachen:

- Schlüsselwörter, Bezeichner, Darstellung von Zahlen, ...
- Programmstrukturen:
  Form der Ausdrücke, Anweisungen, Deklarationen, ...

Formale Sprachen: Beispiele

Programmiersprachen (Java):

```
while (b != 0) { if (a > b) a = a - b; else b = b - a; }
```

Regeln für korrekte Syntax (EBNF):

```
Statement ::=  ... | IfStmt | WhileStmt | ... ;
WhileStmt ::= "while" "(" Expr ")" Statement;
IfStmt    ::= "if" "(" Expr ")" Statement ( "else" Statement )?;
Expr      ::= ...
```

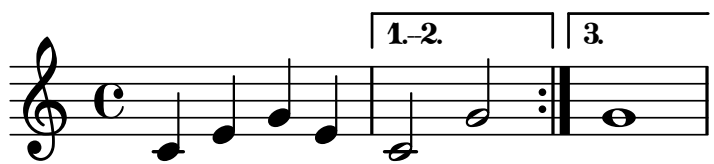
Domain-spezifische Sprachen , z.B. Autotool-Lösungen zu AL-Modell

```
listToFM[(x,True),(y,False),(z,False)]
```

Regeln für korrekte Syntax (EBNF):

```
belegung      ::= "listToFM" "[" var-wert-ps "]"
var-wert-ps   ::= "" | var-wert-paar | var-wert-paar "," var-wert-ps
var-wert-paar ::= "(" var-name, wert ")"
wert          ::= "True" | "False"
var-name      ::= ...
```

Graphische Sprachen , z.B.



Maschinenmodell: endlicher Automat

Beschreibung des dynamischen Verhaltens von Systemen

Modellierung von Abläufen (Zustandsübergangssysteme)

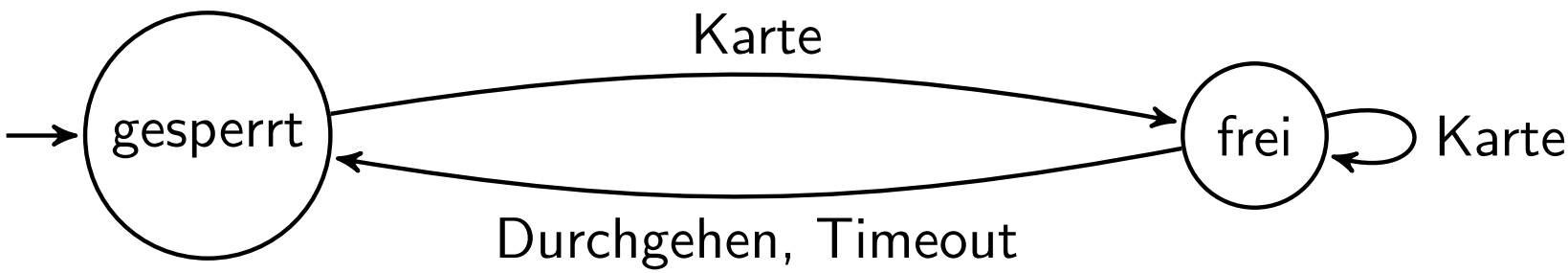
Beispiele:

- Bedienoperationen an Geräten oder Software
- Schaltfolgen von Ampelanlagen
- Steuerung von Produktionsanlagen
- Ablauf von (Geschäfts-)Prozessen

Beispiel: (Pool-)Einlass mit Karte

Automat definiert durch

- Zustände: gesperrt, frei
- Startzustand: gesperrt
- Aktionen (Eingabesymbole):
  Karte (anlegen), Durchgehen, Timeout
- Zustandsübergänge: (gesperrt, Karte) $\rightarrow$ frei
  (frei, Karte) $\rightarrow$ frei
  (frei, Durchgehen) $\rightarrow$ gesperrt
  (frei, Timeout) $\rightarrow$ gesperrt

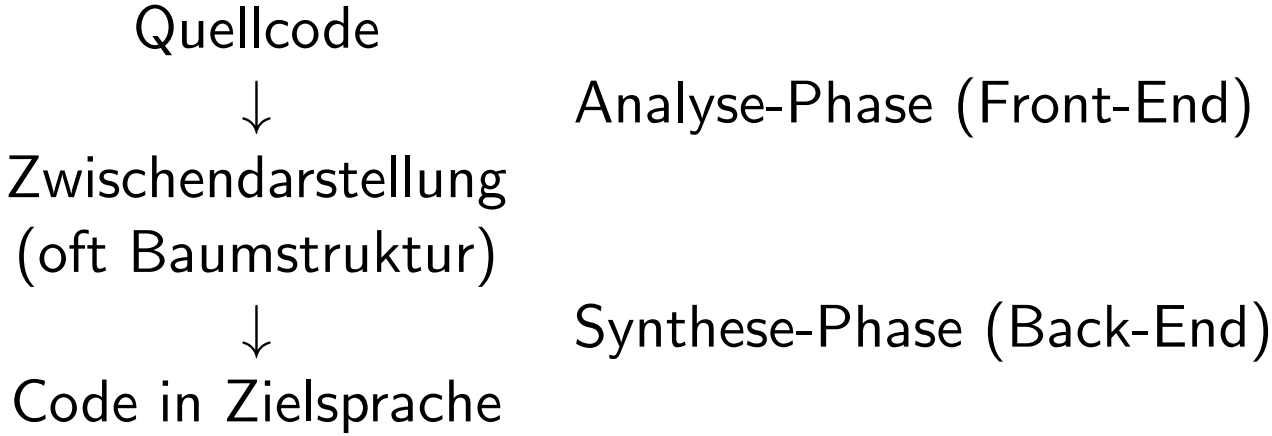


definiert mögliche (erlaubte) Folgen von Aktionen
Diese Folgen lassen sich durch reguläre Ausdrücke darstellen:
(Karte Karte*(Durchgehen + Timeout))*

Anwendung bei der Übersetzung von Programmen

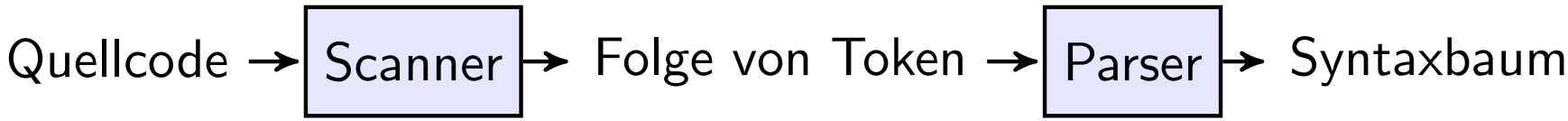
Übersetzung von Quell- in Zielsprache
(z.B. C, Java in Maschinen- oder Byte-Code)

meist in zwei Phasen über eine (gemeinsame) Abstraktion:



(im Master-Modul Compilerbau)

Analyse-Phase



lexikalische Analyse (Scanner)
: lineare Analyse des Quelltextes,
Aufteilung in Einheiten (Token)
z.B. Schlüsselwörter, Bezeichner, Zahlen
reguläre Sprachen, endliche Automaten

Syntaxanalyse (Parser)
: hierarchische Struktur des Quelltextes
z.B. Ausdrücke, Verzweigungen, Schleifen
kontextfreie Sprachen, Kellerautomaten

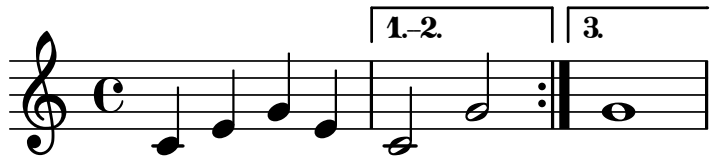
semantische Analyse Annotationen im Syntaxbaum,
: z.B. Typprüfungen

Einsatz ähnlicher Transformations- und Analyse-Methoden

- Compiler für Programmiersprachen (z. B. Java → Bytecode)
- Interpreter für Programmiersprachen (z. B. Java-Bytecode)
- Übersetzung von Daten zwischen verschiedenen Formaten
  z. B. LilyPond (`http://www.lilypond.org`) übersetzt

```
\repeat volta 3 { c' e' g' e' | }
\alternative { { c'2  g' | } { g'1 | } }
```

  u. A. in



- Verarbeitung von Domain-spezifischen Sprachen
- Textformatierung
- Dokumentbeschreibungssprachen
- kontextabhängige Hilfe in Entwicklungsumgebungen
- statische Analyse zur Fehlersuche in Programmen
- graphische Editoren (z.B. für UML-Diagramme) mit automatischer Programmgenerierung

Berechenbarkeit / Entscheidbarkeit

Halteproblem:
Kann ein (Test-)programm U existieren, welches für jedes beliebige (Dienst-)Programm P (Eingabe als Quelltext) entscheidet, ob P nach endlich vielen Schritten anhält?

Antwort (Herleitung später): Nein

Folgerungen:

- Alle Versuche, ein solches Programm zu erzeugen, müssen fehlschlagen.
- Sinnvoll ist jedoch die Suche nach Verfahren, die für eine möglichst große Teilmenge aller (Dienst-)Programme P entscheiden, ob P nach endlich vielen Schritten anhält.
- Entwickler von P (Dienstleister) muss nachweisen, dass sein Programm P nach endlich vielen Schritten anhält.

(im INM-Modul Theoretische Informatik)

Komplexität

Beispiel Primzahltest

Aufgabe: Ist eine gegebene Zahl n eine Primzahl?

Instanz der Aufgabe: Ist 12347 eine Primzahl?

lösbar durch den Algorithmus:

1. Für alle $i \in \{2, \ldots, n\}$:
   Test: Ist n durch i teilbar?
   - ja: Ende mit Ausgabe n ist nicht prim.
   - nein: weiter (mit Test für $i + 1$)
2. Ausgabe: n ist prim.

Dieser Test ist für große Zahlen aufwendig. Geht es besser?

- Was bedeutet aufwendig und besser?
- Wie aufwendig ist eine Berechnung?
- Wie aufwendig ist die Lösung einer Aufgabe?

(im INM-Modul Theoretische Informatik)

Literatur

- Uwe Schöning:
  Theoretische Informatik - kurzgefasst (Spektrum 2001)
- John E. Hopcroft, Jeffrey D. Ullman:
  Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie (Addison-Wesley 1990)

Online Verfügbar (über Bilbliothek):

- Dirk W. Hoffmann: Theoretische Informatik (Hanser 2018)
- Ulrich Hedtstück: Einführung in die Theoretische Informatik Formale Sprachen und Automatentheorie (Oldenbourg 2012)
- Lukas König, Friederike Pfeiffer-Bohnen, Hartmut Schmeck Theoretische Informatik – ganz praktisch (De Gruyter 2016)
- Heinz-Peter Gumm, Manfred Sommer
  Informatik – Band 3: Formale Sprachen, Compilerbau, Berechenbarkeit und Komplexität (De Gruyter 2019)
- Gottfried Vossen, Kurt-Ulrich Witt:
  Grundkurs Theoretische Informatik (Vieweg 2016)
- Renate Winter: Theoretische Informatik (Oldenbourg 2002)

WH: Alphabet, Wort, Sprache

Für jede Menge A heißt

$$A^n = \underbrace{A \times \cdots \times A}_{n} = \{w_1 \cdots w_n \mid \forall i : w_i \in A\}$$

Menge aller Wörter der Länge n über A
(n-Tupel, Vektoren, Folgen, Listen, Zeichenketten)

$A^* = \bigcup_{\{n \in \mathbb{N}\}} A^n$ Menge aller Wörter über A
(endliche Folgen, Listen, Zeichenketten)

$A^0 = \{\varepsilon\}$ mit leerem Wort ε

Alphabet (endliche) Menge A von Symbolen

Wort endliche Folge von Symbolen $w = w_1 \cdots w_n$ mit
$\forall i \in \{1, \ldots, n\} : w_i \in A$

Länge eines Wortes $|w|$ = Anzahl der Symbole in w

Anzahl der Vorkommen eines Symboles in einem Wort
$|w|_a$ = Anzahl der a in w (für $a \in A$)

Sprache Menge von Wörtern $L \subseteq A^*$

Beispiele

- Alphabet $A = \{0, 1\}$
  Wörter $\in A^* = \{0, 1\}^*$: Menge aller Binärwörter
  Sprachen $\subseteq A^*$, z.B.
  - $\{w \in \{0, 1\}^* \mid w_1 \neq 0\}$ Menge aller Binärzahlen ohne führende Nullen
  - $\{w \in \{0, 1\}^* \mid w_1 \neq 0 \wedge w_{|w|-1} = w_{|w|} = 0\}$ Menge aller Binärdarstellungen durch 4 teilbarer Zahlen ohne führende Nullen
- Alphabet $A = \{a, b\}$
  Wörter $\in A^* = \{a, b\}^*$: Menge aller Wörter, die höchstens die Buchstaben a und b enthalten
  Sprachen $\subseteq A^*$, z.B.
  - $\emptyset$
  - $\{a, b\}$
  - $\{a\}^* = \{\varepsilon, a, aa, aaa, \ldots\}$
  - $\{w \in \{a, b\}^* \mid w_1 = a \wedge w_{|w|} = a\} = \{a, aa, aaa, aba, aaaa, abaa, aaba, abba, \ldots\}$

WH: Darstellung von Wörtern

extensional durch Angabe der Symbole in ihrer Reihenfolge
Beispiele: $u = 321$,
$v = abababababa$,
$w = w_1 \cdots w_4$ mit $w_1 = w_2 = w_3 = a$, $w_4 = b$

intensional durch Angabe einer Eigenschaft, die für jeden Index i das i-te Symbol eindeutig bestimmt.

Beispiele:

$u \in \{0, \ldots, 4\}^3$ mit $\forall i \in \{1, \ldots, 3\} : u_i = 4 - i$,

$$v \in \{a, b\}^{11} \text{ mit } \forall i \in \{1, \ldots, 11\} : v_i = \begin{cases} a & \text{falls } i \in 2\mathbb{N} + 1 \\ b & \text{sonst} \end{cases}$$

$$w \in \{a, b\}^4 \text{ mit } w_4 = b \wedge \forall i \in \{1, \ldots, 3\} : w_i = a$$

WH: Darstellung von Sprachen

extensional durch Angabe der Elemente
(nur Beschreibung endlicher Sprachen möglich)
Beispiele: $\{\varepsilon, a, aa, aaa\}$, $\{b, ba, baa, baaa\}$,
$\{a, b, aa, bb, aaa, bbb\}$

intensional durch Angabe einer Eigenschaft, die genau alle Wörter der Sprache haben.
(auch Beschreibung unendlicher Sprachen möglich)
Beispiele: $\{w \in \{a\}^* \mid |w| \leq 3\}$,
$\{w \in \{a, b\}^* \mid w_1 = b \wedge \forall i \in \{2, \ldots, |w|\} : w_i = a\}$,
$\{w \in \{a, b\}^* \mid \forall i \in \{2, \ldots, |w|\} : w_i = w_1\}$

später in dieser LV noch mehr Formalismen zur endlichen Beschreibung von eingeschränkten Sprachklassen
(reguläre Ausdrücke, Grammatiken, Automaten, ...)

WH: Operationen auf Wörtern

Operationen auf Wörtern $u, v \in A^*$:

Verkettung $\circ : A^* \times A^* \to A^*$, wobei

$$\forall u \in A^* \; \forall v \in A^* \; \forall i \in \{1, \ldots, |u| + |v|\} :$$

$$(u \circ v)_i = \begin{cases} u_i & \text{falls } i \leq |u| \\ v_{i-|u|} & \text{sonst} \end{cases}$$

assoziativ, nicht kommutativ, ε ist neutral

Damit ist $(A^*, \circ, \varepsilon)$ ein Monoid.

Spiegelung $^R : A^* \to A^*$, wobei

$$\forall u \in A^* \; \forall i \in \{1, \ldots, |u|\} : u_i^R = u_{|u|+1-i}$$

$u \in A^*$ heißt Palindrom gdw. $u^R = u$

Fakt

- *Für jedes Wort $u \in A^*$ gilt $\left(u^R\right)^R = u$.*
- *Für je zwei beliebige Wörter $u, v \in A^*$ gilt $(u \circ v)^R = v^R \circ u^R$.*

ÜA: Beweis

WH: Relationen zwischen Wörtern

Präfix $\sqsubseteq \subseteq A^* \times A^*$:

$$\forall u \in A^* \; \forall v \in A^* : ((u \sqsubseteq v) \quad \leftrightarrow \quad (\exists w \in A^* \; (u \circ w = v)))$$

z.B. *tom* $\sqsubseteq$ *tomate* (mit $w = ate$)

Suffix (Postfix):

$$\forall u \in A^* \; \forall v \in A^* : (u \text{ Suffix von } v \quad \leftrightarrow \quad (\exists w \in A^* \; (w \circ u = v)))$$

z.B. *enten* ist Suffix von *studenten* (mit $w = stud$)

Infix (Faktor, zusammenhängendes Teilwort):

$$\forall u \in A^* \; \forall v \in A^* : (u \text{ Infix von } v \; \leftrightarrow \; (\exists w \in A^* \; \exists w' \in A^* : (w \circ u \circ w' = v)))$$

z.B. *oma* ist Infix von *tomate* (mit $w = t$ und $w' = te$)

Präfix-, Suffix- und Infixrelation sind Halbordnungen
(aber keine totalen Ordnungen).

Weitere Ordnungen auf Wörtern

Jede totale Ordnung $<$ auf dem Alphabet A definiert totale Ordnungen auf A^*:

lexikographische Ordnung auf A^* ($\leq_{\text{lex}} \subseteq A^* \times A^*$):

$$\forall u, v \in A^* : u \leq_{\text{lex}} v \text{ gdw.}$$

1. $u \sqsubseteq v$ oder
2. $\exists w \in A^* \; \exists x, y \in A : x < y \wedge wx \sqsubseteq u \wedge wy \sqsubseteq v$

quasi-lexikographische Ordnung auf A^* ($\leq_{\text{qlex}} \subseteq A^* \times A^*$):

$$\forall u, v \in A^* : u \leq_{\text{qlex}} v \text{ gdw.}$$

1. $|u| < |v|$ oder
2. $|u| = |v| \wedge u \leq_{\text{lex}} v$

Beispiele: für $A = \{a, b\}$ mit $a < b$

- $ab \sqsubseteq aba$, $ab \leq_{\text{lex}} aba$, $ab \leq_{\text{qlex}} aba$
- $abab \not\sqsubseteq abba$, aber $abab \leq_{\text{lex}} abba$ und $abab \leq_{\text{qlex}} abba$,
- $aaa \leq_{\text{lex}} ab$, aber $aaa \not\leq_{\text{qlex}} ab$
- $ab \not\leq_{\text{lex}} aaba$, aber $ab \leq_{\text{qlex}} aaba$

WH: Sprachen als Mengen

Sprachen $L \subseteq A^*$ sind Mengen von Wörtern
Eigenschaften: leer, endlich, abzählbar, überabzählbar
Mengenrelationen auf Sprachen:

$$L \subseteq L' \quad \text{gdw.} \quad \forall w \in A^* : w \in L \rightarrow w \in L' \text{ gilt}$$
$$L = L' \quad \text{gdw.} \quad \forall w \in A^* : w \in L \leftrightarrow w \in L' \text{ gilt}$$

Mengenoperationen auf Sprachen:

$$\begin{aligned} L \cup L' &= \{w \mid w \in L \vee w \in L'\} \\ L \cap L' &= \{w \mid w \in L \wedge w \in L'\} \\ L \setminus L' &= \{w \mid w \in L \wedge w \notin L'\} \\ L \Delta L' &= (L \setminus L') \cup (L' \setminus L) \end{aligned}$$

Komplement einer Sprache $L \subseteq A^*$: $\overline{L} = A^* \setminus L$
Beispiel:

$$L = \bigcup_{n \in 3\mathbb{N}} A^n \qquad \overline{L} = \bigcup_{n \in \{3i+1 \mid i \in \mathbb{N}\}} A^n \cup \bigcup_{n \in \{3i+2 \mid i \in \mathbb{N}\}} A^n$$

WH: Operationen auf Sprachen

Spiegelung $L^R = \{w^R \mid w \in L\}$

Verkettung $\circ$ von Sprachen:

$$L_1 \circ L_2 = \{u \circ v \mid u \in L_1 \wedge v \in L_2\}$$

iterierte Verkettung von Sprachen $L \subseteq A^*$

$$L^0 = \{\varepsilon\} \qquad \forall n \in \mathbb{N}: \quad L^{n+1} = L^n \circ L = \underbrace{L \circ \cdots \circ L}_{n+1-\text{mal}}$$

$$L^* = \bigcup_{n \in \mathbb{N}} L^n \qquad L^+ = \bigcup_{n \in \mathbb{N} \setminus \{0\}} L^n$$

Spezialfall $L = \{u\} \in A^*$:

$$u^n = \underbrace{u \cdots u}_{n-\text{mal}} \in A^*, \quad u^* = \{u\}^* = \{u^n \mid n \in \mathbb{N}\} \subseteq A^*$$

$$u^+ = u^* \setminus \{\varepsilon\} = \{u\}^+ = \{u^n \mid n \in \mathbb{N} \setminus \{0\}\} \subseteq A^*$$

WH: Reguläre Ausdrücke – Syntax

Die Menge $\mathrm{RegExp}(A)$ aller regulären Ausdrücke über einem Alphabet A ist (induktiv) definiert durch:

- IA $\emptyset \in \mathrm{RegExp}(A)$,
  $\varepsilon \in \mathrm{RegExp}(A)$ und
  für jedes Symbol $a \in A$ gilt $a \in \mathrm{RegExp}(A)$
- IS für alle $E \in \mathrm{RegExp}(A)$ und $F \in \mathrm{RegExp}(A)$ gilt
  $(E + F), EF, (E)^* \in \mathrm{RegExp}(A)$.

Beispiele: $\varepsilon + a$, $\varepsilon + \emptyset$, $(a + \emptyset)^*$, $\varepsilon + ((ab)^* a)^*$

dieselbe Definition kürzer: $\mathrm{RegExp}(A) = \mathrm{Term}(\Sigma_F, \emptyset)$
für die Signatur
$\Sigma_F = \{(\emptyset, 0), (\varepsilon, 0), (^*, 1), (+, 2), (\cdot, 2)\} \cup \{(a, 0) \mid a \in A\}$
(Baumdarstellung)

WH: Reguläre Ausdrücke – Semantik

Jeder reguläre Ausdruck $E \in \mathrm{RegExp}(A)$ repräsentiert eine Sprache $L(E) \subseteq A^*$.

$$
\begin{array}{lrcl}
 & L(\emptyset) & = & \emptyset \\
 & L(\varepsilon) & = & \{\varepsilon\} \\
\forall a \in A: & L(a) & = & \{a\} \\
\forall E, F \in \mathrm{RegExp}(A): & L(E + F) & = & L(E) \cup L(F) \\
\forall E, F \in \mathrm{RegExp}(A): & L(EF) & = & L(E) \circ L(F) \\
\forall E, F \in \mathrm{RegExp}(A): & L(E^*) & = & (L(E))^*
\end{array}
$$

Eine Sprache $L \subseteq A^*$ heißt genau dann regulär, wenn ein regulärer Ausdruck $E \in \mathrm{RegExp}(A)$ mit $L = L(E)$ existiert.

Beispiel: Die Menge L aller Dezimaldarstellungen natürlicher Zahlen ist regulär wegen $L = L\,(0 + (1 + 2 + \cdots + 9)(0 + 1 + \cdots + 9)^*)$

WH: Beispiele

Für $A = \{a, b\}$ gilt

$$
\begin{aligned}
L(ab^*) &= \{a, ab, abb, abbb, abbbb, \ldots\} = \{ab^i \mid i \in \mathbb{N}\} \\
L((ab)^*) &= \{\varepsilon, ab, abab, ababab, \ldots\} = \{(ab)^i \mid i \in \mathbb{N}\} \\
L((a+b)^*) &= \{a, b\}^* \\
L(a^*b^*) &= \{u \circ v \mid u \in a^* \wedge v \in b^*\} \\
L((a^*b^*)^*) &= \{a, b\}^* \\
L((a+b)^*aba) &= \{u \circ aba \mid u \in A^*\}^*
\end{aligned}
$$

Reguläre Ausdrücke ermöglichen eine endliche Darstellung unendlicher Sprachen.

Äquivalenz regulärer Ausdrücke

Zwei reguläre Ausdrücke $E, F \in \mathrm{RegExp}(A)$ heißen genau dann äquivalent, wenn $L(E) = L(F)$ gilt.

Beispiele (ÜA):

- $(a + b)^*$, $(a^* + b^*)^*$ und $a^*(ba^*)^*$ sind paarweise äquivalent
- ab^* und $(ab)^*$ sind nicht äquivalent
- $(11 + 0 + 110 + 011)^*$ und $(11 + 0)^*$ sind ...

Fakt

Die Äquivalenz regulärer Ausdrücke ist eine Äquivalenzrelation.

ÜA: Beweis

Interessante Fragen für formale Sprachen

- Ist ein gegebenes Wort w in der Sprache L enthalten?
  (häufig Wortproblem genannt)
- Enthält die Sprache L nur endlich viele Wörter?
- Gilt $L_1 \subseteq L_2$ für zwei gegebene Sprachen L_1 und L_2?
- Gilt $L_1 = L_2$ für zwei gegebene Sprachen L_1 und L_2?

Fragen zur Regularität:

- Lässt sich die Sprache L durch einen regulären Ausdruck definieren?
  (Gilt $\exists E \in \mathrm{RegExp}(A) : L = L(E)$?)
- Woran kann man erkennen, ob sich eine Sprache durch einen regulären Ausdruck definieren lässt?
- Gilt $L(E) = \emptyset$ für einen gegebenen regulären Ausdruck E?
- Gilt $L(E) = A^*$ für einen gegebenen regulären Ausdruck E?
- Ist ein gegebenes Wort w in der durch den regulären Ausdruck E definierten Sprache $L(E)$ enthalten?

Alle Antworten sind für endliche Sprachen einfach,

aber für unendliche Sprachen oft schwierig.

Wortproblem praktisch

Eingabe : Sprache $L \subseteq A^*$, Wort $w \in A^*$
Frage: Gilt $w \in L$?
Ausgabe: ja oder nein

Beispiele:

- Syntaktische Tests:
  - Ist die gegebene Zeichenkette die Dezimaldarstellung einer ganzen Zahl?
    (Sprache: Menge aller gültigen Dezimaldarstellungen)
  - Ist die gegebene Zeichenkette eine korrekt geformte Email-Adresse (der HTWK)?
  - Ist der gegebene Quelltext ein syntaktisch korrektes Java-Programm?
  - Ist die gegebene Zeichenkette die Binärdarstellung einer geraden Zahl? (durch drei teilbaren Zahl, usw.)
- Folgen von Aktionen:
  - An- und Ausziehen (in umgekehrter Reihenfolge)
  - Ist eine Folge von Aktionen möglich / zulässig ?
  - Führt eine Folge von Eingaben zu einem Fehler?

Wortersetzungssysteme

Alphabet A

Wortersetzungsregel $(l, r) \in A^* \times A^*$ (geschrieben $l \to r$)

Wortersetzungssystem endliche Menge von Wortersetzungsregeln

Beispiele:

- Regel $ba \to ab$,
- Wortersetzungssystem $S = \{a \to ab, ba \to c, abc \to \varepsilon\}$

Anwendung von Wortersetzungsregeln

Eine Regel $l \to r$ ist auf ein Wort $w \in A^*$ anwendbar,
falls l ein Infix von w ist ($w = v \circ l \circ v'$).

Beispiel: Regel $oma \to u$ ist

- auf *isomatte* anwendbar, $v = is, v' = tte$,
- auf *tomate* anwendbar, $v = t, v' = te$,
- auf *matte*, *sommer* und *normal* nicht anwendbar.

Eine Anwendung der Regel $l \to r$ auf ein Wort
$w = v \circ l \circ v'$ ergibt das Wort $v \circ r \circ v'$.
(Ersetzung des Infix l durch r)

Beispiel: $ab \to a$ angewendet auf $baababa = v \circ l \circ v'$

- mit $v = ba$ und $v' = aba$ ergibt $baaaba$
- mit $v = baab$ und $v' = a$ ergibt $baabaa$

Anwendung einer Regel auf ein Wort an einer Position im Wort

Ableitungsschritt

Ableitungsschritt $(w, (l \to r), p, w')$ im Wortersetzungssystem S mit

- Ausgangswort w,
- auf w anwendbare Regel $l \to r$ aus S,
- Position $p \in \{1, \ldots, |w|\}$ im Wort w, an der der Infix l beginnt
- w' ist das nach Anwendung der Regel $l \to r$ an Position p auf w entstandene Wort.

Beispiel:
$S = \{ab \to ba, a \to b\}$, $w = aba$
mögliche Ableitungsschritte in S:
$(aba, (ab \to ba), 1, baa)$
$(aba, (a \to b), 3, abb)$
$(aba, (a \to b), 1, bba)$

Ein-Schritt-Ableitungsrelation

Jedes Wortersetzungssystem $S \subseteq A^* \times A^*$ definiert eine Relation $\rightarrow_S \subseteq A^* \times A^*$, wobei genau dann $w \rightarrow_S w'$ gilt, wenn ein Ableitungsschritt $(w, (l \rightarrow r), p, w')$ mit $(l \rightarrow r) \in S$ existiert.

Beispiel: Für $S = \{ab \rightarrow ba, a \rightarrow b\}$ gilt

- $aba \rightarrow_S baa$ wegen $(aba, (ab \rightarrow ba), 1, baa)$
- $aba \rightarrow_S bba$ wegen $(aba, (a \rightarrow b), 1, bba)$
- $aba \rightarrow_S abb$ wegen $(aba, (a \rightarrow b), 3, abb)$
- $aba \not\rightarrow_S bbb$

Ableitungen

Eine Folge von Ableitungsschritten

$$(u, (l_1 \to r_1), p_1, u_2), (u_2, (l_2 \to r_2), p_2, u_3), \cdots, (u_{n-1}, (l_{n-1} \to r_{n-1}, p_{n-1}, v)$$

im Wortersetzungssystem S heißt Ableitung von u nach v in S.

Beispiel: $S = \{ab \to ba, a \to b\}$, $u = aba$
Folge von Ableitungsschritten
$(aba, (ab \to ba), 1, baa)$, $(baa, (a \to b), 3, bab)$, $(bab, (a \to b), 2, bbb)$

$$\underline{ab}a \overset{ab \to ba}{\longrightarrow} ba\underline{a} \overset{a \to b}{\longrightarrow} b\underline{a}b \overset{a \to b}{\longrightarrow} bbb$$

Länge der Ableitung = Anzahl der Ableitungsschritte

In jedem System S existiert für jedes $u \in A^*$ die leere Ableitung (der Länge 0) von u nach u.

Wiederholung: Hüllen binärer Relationen

$R \cup I_M$ heißt reflexive Hülle von $R \subseteq M^2$
(mit Identität $I_M = \{(x, x) \mid x \in M\}$)

$R \cup R^{-1}$ heißt symmetrische Hülle von $R \subseteq M^2$
(mit inverser Relation $R^{-1} = \{(y, x) \mid (x, y) \in R\}$)

Wiederholung: Verkettung $\circ$ der Relationen $R \subseteq M^2$ und $S \subseteq M^2$

$$R \circ S = \{(x, z) \in M^2 \mid \exists y \in M : (x, y) \in R \wedge (y, z) \in S\}$$

Iterierte Verkettung von $R \subseteq M^2$ mit sich selbst:

$$\begin{aligned} R^0 &= I_M \\ R^{n+1} &= R^n \circ R \\ R^+ &= \bigcup_{n \in \mathbb{N} \setminus \{0\}} R^n \subseteq M^2 \qquad \text{transitive Hülle} \\ R^* &= \bigcup_{n \in \mathbb{N}} R^n \subseteq M^2 \qquad \text{reflexiv-transitive Hülle} \end{aligned}$$

Ersetzungsrelation

Jedes Wortersetzungssystem $S \subseteq (A^* \times A^*)$ definiert die Ersetzungsrelation $\rightarrow_S^* \subseteq (A^* \times A^*)$, wobei genau dann $u \rightarrow_S^* v$ gilt, wenn eine Ableitung von u nach v existiert.

Beispiel: $S = \{a \rightarrow aa\}$,

- für jedes $n \geq 1$ gilt $ba \rightarrow_S^* b\underbrace{a \cdots a}_{n}$

  wegen $ba \rightarrow_S baa \rightarrow_S baaa \rightarrow_S \cdots \rightarrow_S b\underbrace{a \cdots a}_{n}$
- $b \rightarrow_S^* b$, aber für kein Wort $w \neq b$ gilt $b \rightarrow_S^* w$

($\rightarrow_S^*$ ist die reflexive transitive Hülle von $\rightarrow_S$)

(Beschränktes) Münzenspiel als Wortersetzungssystem

Symbole Höhe der Stapel $A = \{0, \ldots, 4\}$

Konfiguration Wort $w \in A^5$

Startkonfiguration Wort $W = 00400$

Spielzug Anwendung einer Regel aus dem Wortersetzungssystem

$$S = \left\{ i(j+2)k \to (i+1)j(k+1) \mid \begin{array}{l} i,j,k \in \{0,\ldots,4\} \\ \wedge\ i+(j+2)+k \leq 4 \end{array} \right\}$$

$$= \left\{ \begin{array}{l} 040 \to 121, 220 \to 301, 022 \to 103, 121 \to 202 \\ 030 \to 111, 130 \to 211, 031 \to 112, \\ 020 \to 101, 120 \to 201, 021 \to 102 \end{array} \right\}$$

Das Paar $(S, 00400)$ definiert die Menge (Sprache) aller aus 00400 durch $\to_S^*$ erreichbaren Konfigurationen:

$$L(S, 00400) = \{00400, 01210, 02020, 10120, 02101, 10201, 11011\}$$

Auf das Wort 11011 ist keine Regel aus S anwendbar.
Deshalb heißt 11011 Normalform bzgl. S

Wortersetzungssysteme – Beispiele

$S_1 = \{||| \to |\}$ mit $u = |||||||$ und $v = ||||$
Was wird hier „berechnet“?

$S_2 = \{11 \to 1, 00 \to 1, 01 \to 0, 10 \to 0\}$ und $u = 1101001$
Wirkung verschiedener Ableitungreihenfolgen?

$S_3 = \{c \to aca, c \to bcb, c \to a, c \to b, c \to \varepsilon\}$ und $u = c$
Menge aller in S_3 aus c ableitbaren Wörter, die kein c enthalten?

Sprachen aus Wortersetzungssystemen

Jedes Paar (Wortersetzungssystem S, Anfangswort w) über einem Alphabet A definiert die Sprache

$$L(S, w) = \{v \in A^* \mid w \to_S^* v\}$$

(Menge aller Wörter v, die von w durch eine Ableitung in S erreicht werden)
Beispiel: $S = \{c \to aca, c \to bcb\}$, $w = c$
$L(S, w) = \{v \circ c \circ v^R \mid v \in \{a, b\}^*\}$ (Menge aller Palindrome über $\{a, b, c\}$, die genau an der mittleren Position ein c enthalten)
Jedes Paar (Wortersetzungssystem S, Menge M von Wörtern) über einem Alphabet A definiert die Sprache

$$L(S, M) = \bigcup_{w \in M} L(S, w)$$

(alle Wörter v, die von irgendeinem $w \in M$ durch eine Ableitung in S erreicht werden)
Beispiel: Für $S = \{a \to aa\}$ und $M = \{b, ba\}$ ist $L(S, M) = L(ba^*)$

Ausdrucksstärke von Wortersetzungssystemen

Wortersetzungssysteme

- ermöglichen eine endliche Darstellung unendlicher Sprachen.
  (als Erzeugungsvorschrift für alle Wörter der Sprache)
  Beispiele: $L(\{\varepsilon \to aaa\}, \varepsilon) = \{a^{3n} \mid n \in \mathbb{N}\} = L(aaa)^*$
  $L(\{2 \to 020, 2 \to 121\}, 2) = \{w2w^R \mid w \in \{0,1\}^*\}$

- können zur Modellierung von Zuständen und Übergängen dazwischen verwendet werden
  z.B. Spiele, Ausführung von Programmen, Programmverifikation
  Beispiel: Münzenspiel, lineares Solitaire (ÜA)

- können Berechnungen simulieren
  (Bestimmung von erreichbaren Wörtern ohne Nachfolger)
  Beispiel: $\varepsilon \in L(\{||| \to |\}, ||||||)$

Wortproblem für durch Wortersetzungssysteme definierte Sprachen

Ist ein gegebenes Wort w in der Sprache $L(S, u)$ enthalten?

alternative Formulierung: Gilt $u \to_S^* w$?

Ableitungsrelation $\to_S$ als
(unendlicher) gerichteter Graph $G_S = (V, E)$ mit
Knoten: $V = A^*$
Kanten: $E = \{(u, v) \mid u \to_S v\}$

$u \to_S^* w$ gilt genau dann, wenn in G_S ein Pfad von u nach w existiert.

Beispiel: (Tafel) $S = \{ab \to ba\}$,
$abab \to_S^* baba$, aber $abab \not\to_S^* abaa$, $abab \not\to_S^* aabb$

Sprachen aus Wortersetzungssystemen

Lösung des Wortproblems und anderer Fragen zu Sprachen ist für endliche Sprachen einfach, für unendliche Sprachen oft nicht.

Darstellung der Sprache durch ein Wortersetzungssystem kann helfen.

Lösung des Wortproblem $w \in L(S, u)$ durch Standardverfahren:
Suche eines Pfades von u nach w im Ableitungsgraphen des Wortersetzungssystems S

Problem:

- Pfadsuche ist Standardverfahren für endliche Graphen.
- Ableitungsgraphen von Wortersetzungssystemen sind aber meist unendlich.

Standardverfahren ist in Spezialfällen anwendbar,
für welche die Suche in einem endlichen Teilgraphen genügt

Nichtverlängernde Wortersetzungssysteme

Ein Wortersetzungssystem S heißt genau dann nichtverlängernd, wenn für jede Regel $(l \to r) \in S$ gilt: $|l| \geq |r|$.

Wortproblem (L, w):

Eingabe : Sprache $L \subseteq A^*$, Wort $w \in A^*$
Frage: Gilt $w \in L$?
Ausgabe ja oder nein

Beispiel: $S = \{ab \to ba, ac \to a\}$, $u = abcac$, $v = aacb$

Satz

Es gibt einen Algorithmus, welcher für jedes nichtverlängernde Wortersetzungssystem $S \subseteq A^* \times A^*$ und beliebige Wörter $u, w \in A^*$ das Wortproblem $(L(S, u), w)$ in endlicher Zeit korrekt löst.

Idee:
Suche im endlichen Teilgraphen aller Wörter $v \in A^*$ mit $|v| \leq |u|$

Nichtverkürzende Wortersetzungssysteme

Ein Wortersetzungssystem S heißt genau dann nichtverkürzend, wenn für jede Regel $(l \to r) \in S$ gilt: $|l| \leq |r|$.

Beispiel: $S = \{a \to ba, b \to a\}$, $u = b$, $w = aba$, $w' = ab$

Satz

Es gibt einen Algorithmus, welcher für für jedes nichtverkürzende Wortersetzungssystem $S \subseteq A^* \times A^*$ und beliebige Wörter $u, w \in A^*$ das Wortproblem $(L(S, u), w)$ in endlicher Zeit korrekt löst.

Idee:
Suche im endlichen Teilgraphen aller Wörter $v \in A^*$ mit $|u| \leq |v| \leq |w|$

Wortersetzungssysteme mit verlängernden und verkürzenden Regeln

Beispiel:

$$S = \left\{ \begin{array}{lcl} c & \to & baaca, \\ c & \to & aacba, \\ c & \to & bbcabb, \\ aca & \to & d, \\ bcb & \to & d, \\ ada & \to & d, \\ bdb & \to & d \end{array} \right\}$$

Gilt $c \to_S^* d$?

Für Wortersetzungssysteme S mit verlängernden und verkürzenden Regeln existiert im Allgemeinen kein Algorithmus, der für beliebige Wörter $u, w \in A^*$ feststellt, ob $u \to_s^* w$ gilt.

Was bisher geschah

- Alphabet, Wort, Sprache
- Operationen und Relationen auf Wörtern und Sprachen
- interessante Fragen für Sprachen und Wörter

Reguläre Ausdrücke

- Syntax, Semantik
- endliche Darstellung evtl. unendlicher Sprachen

Wortersetzungssysteme $P \subseteq A^* \times A^*$

- Wortersetzungssregel $l \to r$ mit $l, r \in A^*$
- Ableitung in P: endliche Folge von Ersetzungsschritten
- Ausdrucksstärke:
  - Repräsentation von (auch unendlichen) Sprachen
  - Modellierung von Zustandsübergängen
  - Ausführen von Berechnungen

Łukasiewicz-Sprache $L(\{b \to abb\}, b)$

Ableitbare Wörter über Teilalphabet

Beispiele: $A = \{a, b, c\}$,

$$S = \left\{ \begin{array}{lcl} c & \to & aca, \\ c & \to & bcb, \\ c & \to & \varepsilon, \\ c & \to & a, \\ c & \to & b \end{array} \right\}$$

$$L(S, c) = \{u \circ d \circ u^R \mid u \in \{a, b\}^* \wedge d \in \{a, b, c, \varepsilon\}\}$$

Menge aller Wörter in $L(S, c) \cap \{a, b\}^*$:
alle Palindrome in $\{a, b\}^*$

c ist Hilfssymbol zur Erzeugung der Palindrome

Natürliche Sprache

Wortersetzungssystem S enthält die Regeln:

Satz	$\rightarrow$	Subjekt Prädikat .
Subjekt	$\rightarrow$	mArtikel mSubstantiv
Subjekt	$\rightarrow$	wArtikel wSubstantiv
mArtikel	$\rightarrow$	Der
wArtikel	$\rightarrow$	Die
mSubstantiv	$\rightarrow$	Hund
wSubstantiv	$\rightarrow$	Sonne
Prädikat	$\rightarrow$	bellt
Prädikat	$\rightarrow$	scheint

Alphabet $A = \{\text{Der}, \text{Die}, \text{Hund}, \text{Sonne}, \text{scheint}, \text{bellt}, .\} \cup$
{ Satz,Subjekt, Prädikat, wArtikel, mArtikel, wSubstantiv, mSubstantiv}

Ableitbare Wörter in $L(S,$ Satz) ohne Hilfssymbole aus der Menge
{Satz, Subjekt, Prädikat, mArtikel, wArtikel, mSubstantiv, wSubstantiv}:
Menge grammatisch korrekter deutscher Sätze (dieser einfachen Form mit ausschließlich den Worten Der, Die, Hund, Sonne, scheint, bellt).

Aussagenlogische Formeln

Wortersetzungssystem S enthält die Regeln

Formel	$\to$	Variable
Formel	$\to$	Konstante
Formel	$\to$	($\neg$ Formel)
Formel	$\to$	(Formel $\vee$ Formel)
Formel	$\to$	(Formel $\wedge$ Formel)
Variable	$\to$	p
Variable	$\to$	q
Konstante	$\to$	$\mathbb{t}$
Konstante	$\to$	$\mathbb{f}$

Alphabet $A = \{\mathbb{t}, \mathbb{f}, p, q, \neg, \vee, \wedge, (,)\} \cup \{$ Formel,Variable, Konstante$\}$

$$\text{Formel} \to_S (\text{Formel} \wedge \text{Formel}) \to_S^2 (\text{Formel} \wedge \mathbb{f}) \to_S^* ((p \vee (\neg q)) \wedge \mathbb{f})$$

Wörter in $L(S, \text{ Formel }) \cap \{\mathbb{t}, \mathbb{f}, p, q, \neg, \vee, \wedge, (,)\}^*$:
Menge $\text{AL}_{\{\neg,\vee,\wedge,\mathbb{t},\mathbb{f}\}}(\{p, q\})$ aller aussagenlogischen Formeln mit Aussagenvariablen aus der Menge $\{p, q\}$
und Junktoren aus der Menge $\{\neg, \vee, \wedge, \mathbb{t}, \mathbb{f}\}$

Aussagenlogische DNF

Wortersetzungssystem S enthält die Regeln

DNF	$\to$	Minterm $\vee$ DNF
DNF	$\to$	Minterm
Minterm	$\to$	Literal $\wedge$ Minterm
Minterm	$\to$	Literal
Literal	$\to$	$\neg$ Variable
Literal	$\to$	Variable
Variable	$\to$	p
Variable	$\to$	q

Alphabet $A =$ $\{p, q, \neg, \vee, \wedge\}$
$\cup\{$ DNF, Minterm, Literal,Variable$\}$

$$\text{DNF} \to_S \text{Minterm} \vee \text{DNF} \to_S^3 p \vee \text{DNF} \to_S^* p \vee q \wedge \neg p \vee \neg q$$

Wörter in $L(S, \text{DNF}) \cap \{p, q, \neg, \vee, \wedge\}^*$:
Menge AL($\{p, q\}$) aller disjunktiven Normalformen mit
Aussagenvariablen aus der Menge $\{p, q\}$

Dezimaldarstellung natürlicher Zahlen

Wortersetzungssystem S enthält die Regeln

Zahl	$\to$	0
Zahl	$\to$	1Ziffernfolge
	$\vdots$	
Zahl	$\to$	9Ziffernfolge
Ziffernfolge	$\to$	0Ziffernfolge
	$\vdots$	
Ziffernfolge	$\to$	9Ziffernfolge
Ziffernfolge	$\to$	ε

Alphabet $\{0, \ldots, 9\} \cup \{$ Zahl, Ziffernfolge $\}$

Zahl $\to_S$ 3Ziffernfolge $\to_S$ 32Ziffernfolge $\to_S$ 327Ziffernfolge $\to_S$ 327

Wörter in $L(S,$ Zahl $) \cap \{0, \ldots, 9\}^*$:
Menge aller Dezimaldarstellungen natürlicher Zahlen

Programmiersprachen

Java-Syntax (Ausschnitt) in Backus-Naur Form (BNF)
(John Backus, Peter Naur)

```
<while statement>::=while(<expression>)<statement>

<expression>::=<assignment expression>

<assignment expression>::=<conditional expression>|<assignment>

<assignment>::=<left hand side><assignment operator><assignment expression>

<left hand side>::=<expression name>|<field access>|<array access>

<assignment operator>::=  = | *= | /= | %= | += | -= | <<= | >>= | >>>=
                        | &= | ^= | |=
```

$a \to r_1|r_2|\ldots|r_n$ statt mehrerer Regeln $a \to r_1, \ldots, a \to r_n$
$::=$ statt $\to$ (in ASCII darstellbar)
Hilfssymbole markiert durch $<$ und $>$

Definition Grammatik

Grammatik $G = (N, T, P, S)$ ist definiert durch

Nichtterminalsymbole: endliche Menge N
(Hilfssymbole)

Terminalsymbole: endliche Menge T
(Alphabet der erzeugten Sprache)

Wortersetzungssystem: $P \subseteq (N \cup T)^+ \times (N \cup T)^*$ (Produktionen)

Startsymbol $S \in N$

Beispiel: $G = (N, T, P, S)$ mit

$$
\begin{aligned}
N &= \{S\}, \\
T &= \{0, 1\}, \\
P &= \left\{ \begin{array}{lcl} S & \rightarrow & 0S1 \\ S & \rightarrow & \varepsilon \end{array} \right\}
\end{aligned}
$$

Grammatiken: Beispiele

- $G = (N, T, P, E)$ mit $N = \{E, F, G\}$, $T = \{(,), a, +, \cdot\}$ und

$$P = \left\{ \begin{array}{lcl} E & \to & G, \\ E & \to & E + G, \\ G & \to & F, \\ G & \to & G \cdot F, \\ F & \to & a, \\ F & \to & (E) \end{array} \right\} = \left\{ \begin{array}{lcl} E & \to & G \mid E + G, \\ G & \to & F \mid G \cdot F, \\ F & \to & a \mid (E) \end{array} \right\}$$

- $G = (N, T, P, S)$ mit $N = \{S, A, B, C\}$, $T = \{a, b, c\}$

$$P = \left\{ \begin{array}{lcl} S & \to & aSBC, \\ S & \to & aBC, \\ CB & \to & BC, \\ aB & \to & ab, \\ bB & \to & bb, \\ bC & \to & bc, \\ cC & \to & cc \end{array} \right\}$$

Ableitungen in Grammatiken

Ableitung in Grammatik $G = (N, T, P, S)$:
Ableitung im Ersetzungssystem P mit Startwort S

Beispiel: $G = (N, T, P, S)$ mit

$$
\begin{aligned}
N &= \{S, A, B\} \\
T &= \{0, 1\} \\
P &= \left\{ \begin{array}{lcl} S & \to & 0SA \\ S & \to & 0A \\ A & \to & 1 \\ B & \to & A \end{array} \right\}
\end{aligned}
$$

Durch Grammatiken definierte Sprachen

Grammatik $G = (N, T, P, S)$ definiert die Sprache

$$L(G) = \{w \in T^* \mid S \rightarrow_P^* w\} = L(P, S) \cap T^*$$

Beispiel: $G = (N, T, P, S)$ mit

$$\begin{aligned} N &= \{S, Z\} \\ T &= \{0, 1\} \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & 1Z, \\ S & \rightarrow & 0, \\ Z & \rightarrow & 0Z, \\ Z & \rightarrow & 1Z, \\ Z & \rightarrow & \varepsilon \end{array} \right\} \end{aligned}$$

definiert die Sprache $L(G) = \ldots$

Äquivalenz von Grammatiken

Zwei Grammatiken G_1 und G_2 heißen genau dann äquivalent, wenn $L(G_1) = L(G_2)$ gilt.

Beispiel: $G_1 = (N_1, \{0,1\}, P_1, S)$ mit $N_1 = \{S, A, B\}$ und

$$P_1 = \left\{ \begin{array}{lcl} S & \to & 0SA \\ S & \to & 0A \\ A & \to & 1 \\ B & \to & A \end{array} \right\}$$

und $G_2 = (N_2, \{0,1\}, P_2, S')$ mit $N_2 = \{S'\}$ und

$$P_2 = \{S' \to 0S'1, S' \to 01\}$$

sind äquivalent wegen $L(G_1) = L(G_2) = \ldots$

Äquivalenz von Grammatiken ist eine Äquivalenzrelation. (ÜA)

Grammatiken mit ε-Regeln

ε-Regel : Grammatik-Regel der Form $l \to \varepsilon$

Beispiele:

- $G_1 = (\{A, B\}, \{0, 1\}, P_1, A)$ mit
  $P_1 = \{A \to 1B0, B \to 1B0, B \to \varepsilon\}$
  ist äquivalent zu $G_2 = (\{A\}, \{0, 1\}, P_2, A)$ mit
  $P_2 = \{A \to 1A0, A \to 10\}$
  (ohne ε-Regel)
- $G = (\{A\}, \{0, 1\}, P, A)$ mit $P = \{A \to 1A0, A \to \varepsilon\}$
  ist nicht äquivalent zu einer Grammatik ohne ε-Regel

Chomsky-Hierarchie

(Noam Chomsky)

Eine Grammatik $G = (N, T, P, S)$ heißt vom Chomsky-Typ

0 immer,

1 , falls für jede Regel $(l \to r) \in P$ gilt: $|l| \leq |r|$
(monoton, kontextsensitiv)

2 , falls Typ 1 und für jede Regel $(l \to r) \in P$ gilt: $l \in N$
(kontextfrei)

3 , falls Typ 2 und für jede Regel $(l \to r) \in P$ gilt:
$l \in N$ und $r \in (T \cup (T \circ N))$
(regulär)

Eine Sprache $L \subseteq T^*$ heißt vom (Chomsky-)Typ i für $i \in \{0, \ldots, 3\}$, falls i die größte Zahl ist, für die eine Grammatik G vom Typ i mit $L \setminus \{\varepsilon\} = L(G)$ existiert.

$\mathcal{L}_i$ bezeichnet die Menge aller Sprachen vom Typ i.

Achtung: Es gibt Sprachen, die nicht durch eine Grammatik erzeugt werden können, also keinen Chomsky-Typ haben (Warum?)

Beispiele

- $G_1 = (\{S\}, \{a, b\}, P, S)$ mit
  $P = \{S \to aSb, S \to ab, S \to a, S \to b\}$ ist vom Typ 2
- $G_2 = (\{S\}, \{a, b\}, P, S)$ mit
  $P = \{S \to aSb, S \to a, S \to b, S \to \varepsilon\}$ ist vom Typ 0
  aber $L(G_2)$ ist vom Typ 2,
  weil $L(G_2) \setminus \{\varepsilon\} = L(G_2')$ für $G_2' = (\{S\}, \{a, b\}, P', S)$ mit
  $P' = \{S \to aSb, S \to a, S \to b, S \to ab\}$
- $G_3 = (\{S, A, B\}, \{a, b\}, P, S)$ mit
  $P = \{S \to aA, A \to aB, B \to bB, B \to b\}$ ist vom Typ 3
- $G_4 = (\{A, B, C\}, \{a, b, c\}, P, A)$ mit

$$P = \left\{ \begin{array}{lcl} A & \to & aABC, \\ A & \to & aBC, \\ CB & \to & BC, \\ aB & \to & ab, \\ bB & \to & bb, \\ bC & \to & bc, \\ cC & \to & cc \end{array} \right\}$$

  ist vom Typ 1

Wortproblem für Typ-1-Sprachen

gegeben : Grammatik $G = (N, T, P, S)$ vom Chomsky-Typ 1,
Wort $w \in T^*$
Frage : Gilt $w \in L(G)$?

Satz

Es existiert ein Algorithmus, welcher für jede beliebige Eingabe (G, w), *wobei*

- T *ein endliches Alphabet,*
- $w \in T^*$ *und*
- G *eine monotone Grammatik (Chomsky-Typ 1) über* T *sind*

die Wahrheit der Aussage $w \in L(G)$ *(in endlicher Zeit) korrekt beantwortet.*

(folgt aus entsprechendem Satz für nichtverkürzende Wortersetzungssysteme)
Im weiteren Verlauf dieses Semesters:
spezielle (effizientere) Verfahren für Grammatiken von Typ 2 und 3

WH: Mächtigkeit unendlicher Mengen

Eine Menge A heißt

abzählbar gdw. $\exists\, f : \mathbb{N} \to A$ surjektiv

überabzählbar sonst

Abschlusseigenschaften der Menge aller abzählbaren Mengen:
Sind alle Mengen A_i mit $i \in \mathbb{N}$ abzählbar, dann sind auch

- jede Teilmenge $B \subseteq A_1$ abzählbar
- $B = A_1 \cup A_2$ abzählbar
- $B = A_1 \times A_2$ abzählbar
  (nach erstem Diagonalverfahren von Cantor)
  z.B. $B_k = ((A_1)_m, (A_2)_n)$ für $k = \sum_{i=0}^{m+n} i + m$ (ÜA)
- $\forall k \in \mathbb{N} : B = A_1^k$ abzählbar (ÜA)

Ist jede der abzählbar vielen Mengen $A_0, A_1, A_2, \ldots$ abzählbar,
dann sind auch

- $\bigcup_{i \in \mathbb{N}} A_i$ abzählbar (Hilberts Hotel)
- $\bigtimes_{i \in \mathbb{N}} A_i$ abzählbar

(formale Beweise durch Induktion nach i)

WH: Abzählbare Mengen – Beispiele

A abzählbar gdw. $\exists f : \mathbb{N} \to A$ surjektiv

Beispiele:

- jede endliche Menge $A = \{a_0, \ldots, a_n\}$ mit

$$\forall i \in \mathbb{N} : f(i) = \begin{cases} a_1 & \text{, falls } i \leq n \\ a_0 & \text{, falls } i > n \end{cases}$$

- $\mathbb{N}$ mit $f =$ Identität
- jede Menge $M \subseteq \mathbb{N}$
- $\mathbb{N} \times \mathbb{N}$ (nach erstem Diagonalverfahren von Cantor)
- $\mathbb{Z} = \{m - n \mid (m, n) \in \mathbb{N}^2\} = \mathbb{N} \cup \{-n \mid n \in \mathbb{N} \setminus \{0\}\}$
- $\mathbb{Q} = \{m/n \mid (m, n) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})\}$
- $\forall k \in \mathbb{N} : \mathbb{N}^k$
- $\{1\}^*$
- $\{0, 1\}^*$
- A^* für jedes endliche Alphabet A (ÜA)

WH: Überabzählbare Mengen

Menge $2^{\mathbb{N}}$ aller Mengen natürlicher Zahlen ist überabzählbar, also mächtiger als $\mathbb{N}$.
(Es gibt überabzählbar viele Mengen natürlicher Zahlen.)
(indirekter) Beweis, dass $2^{\mathbb{N}}$ überabzählbar
zweites Diagonalverfahren von Cantor
Annahme: Es existiert eine surjektive Funktion $M : \mathbb{N} \to 2^{\mathbb{N}}$, d.h. für jede Menge $L \subseteq \mathbb{N}$ gilt also $L = M(k)$ für (wenigstens) ein $k \in \mathbb{N}$.

	0	1	2	$\cdots$	k	$\cdots$
$M(0)$	$0 \in M(0)$	$1 \in M(0)$	$2 \in M(0)$	$\cdots$	$k \in M(0)$	$\cdots$
$M(1)$	$0 \in M(1)$	$1 \in M(1)$	$2 \in M(1)$	$\cdots$	$k \in M(1)$	$\cdots$
$M(2)$	$0 \in M(2)$	$1 \in M(2)$	$2 \in M(2)$	$\cdots$	$k \in M(2)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	
$M(k)$	$0 \in M(k)$	$1 \in M(k)$	$2 \in M(k)$	$\cdots$	$k \in M(k)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	

Definiere Menge $L = \{i \in \mathbb{N} \mid i \notin M(i)\} \subseteq \mathbb{N}$
Wegen $L \subseteq \mathbb{N}$ existiert nach Annahme ein $k \in \mathbb{N}$ mit $L = M(k)$ und damit $k \in M(k)$ gdw. $k \notin M(k)$ (Widerspruch),
d.h. Annahme falsch, also $2^{\mathbb{N}}$ nicht abzählbar

Beispiele überabzählbarer Mengen

Mit dem zweiten Diagonalverfahren von Cantor lässt sich zeigen:

- $\mathbb{R}$ ist überabzählbar (mächtiger als $\mathbb{N}$).
  (Es gibt überabzählbar viele reelle Zahlen.)
- $[0,1]$ $\subset \mathbb{R}$ ist überabzählbar.
  (Intervall $[0,1]$ enthält überabzählbar viele reelle Zahlen.)
- $2^{(\{0,1\}^*)}$ Menge aller Sprachen $L \subseteq \{0,1\}^*$ ist überabzählbar, also mächtiger als $\{0,1\}^*$.
- $2^{(A^*)}$ ist für jedes nichtleere endliche Alphabet A überabzählbar.
  (Für jedes nichtleere endliche Alphabet A ist die Menge aller Sprachen über A überabzählbar.) (ÜA)

Ist jede Sprache regulär ?

Existiert für jedes endliche Alphabet A zu jeder Sprache $L \subseteq A^*$ ein regulärer Ausdruck E mit $L = L(E)$?

Nein, Beweis (2 Möglichkeiten):

1. konstruktiv durch Gegenbeispiel
   z.B. $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ (Nachweis erst später möglich)

2. nichtkonstruktiv (mit schon bekannten Methoden möglich):

   - Wieviele Sprachen $L \subseteq A^*$ gibt es? (Mächtigkeit von $2^{(A^*)}$) überabzählbar viele
   - Wieviele reguläre Ausdrücke über dem endlichen Alphabet A gibt es? abzählbar viele, weil
     - Alphabet $A' = A \cup \{\emptyset, \varepsilon, +, ^*, (,), \}$ endlich
     - Menge $(A')^*$ aller Wörter über A' abzählbar
     - RegExp$(A) \subset (A')^*$ (endliche Beschreibung)
     - also RegExp(A) selbst abzählbar

   Damit existieren sogar sehr viel mehr (überabzählbar viele) nicht reguläre Sprachen.

Warum sind reguläre Ausdrücke trotzdem interessant?

Wird jede Sprache von einer Grammatik erzeugt?

Existiert für jedes endliche Alphabet A zu jeder Sprache $L \subseteq A^*$ eine Grammatik $G = (N, T, P, S)$ mit $L = L(G)$?

Nein (Gegenbeispiel später)

Begründung (nichtkonstruktiv):

1. Wieviele Sprachen $L \subseteq A^*$ gibt es? (Mächtigkeit von $2^{(A^*)}$)

   überabzählbar viele

2. Wieviele Grammatiken $G = (N, T, P, S)$ gibt es?
   abzählbar viele, weil

   - Alphabet $A' = N \cup T \cup \{(,), , , \{, \}, \rightarrow, \varepsilon\}$ endlich
   - Menge $(A')^*$ aller Wörter über A' abzählbar
   - jede Grammatik ist ein Wort aus $(A')^*$
     (endliche Beschreibung der Sprache)
   - Menge aller Grammatiken ist Teilmenge der abzählbaren Menge $(A')^*$, also selbst abzählbar

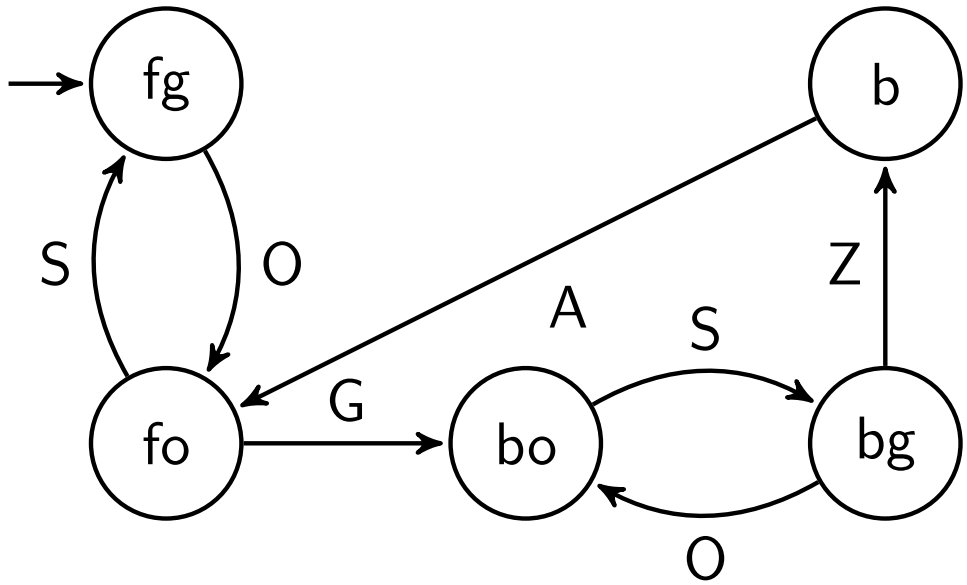
Also existieren überabzählbar viele Sprachen, die nicht durch Grammatiken erzeugt werden können.

Warum sind Grammatiken trotzdem interessant?

Was bisher geschah

- Alphabet, Wort, Sprache
- Operationen auf Wörtern und Sprachen
- Wortersetzungssysteme, $L(S, w)$
- Grammatiken, $L(G)$
- Äquivalenz von Grammatiken
- Besonderheit ε-Regeln
- Chomsky-Hierarchie
- Existenz nicht durch Grammatiken erzeugter Sprachen (Mächtigkeitsargument, nicht konstruktiv)

WH: Zustandsübergangssystem Münzschließfach



Aktionen: A aufschließen
Z zuschließen
O Tür öffnen
S Tür schließen
G Geld einwerfen

Zustände : fg frei, Tür zu
fo frei, Tür offen
bo bezahlt, Tür offen
bg bezahlt, Tür zu
b belegt

WH: Endliche Automaten – Definition

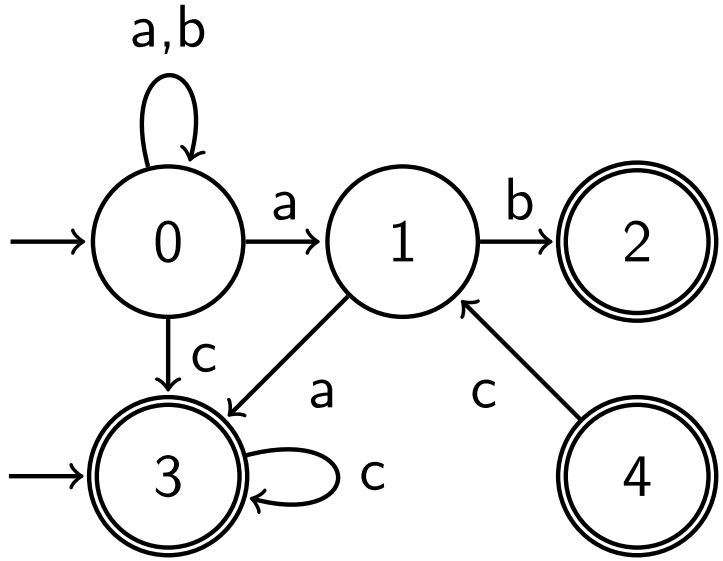
NFA (nondeterministic finite automaton)
$A = (X, Q, \delta, I, F)$ mit

- X endliches Alphabet,
- Q endliche Menge von Zuständen,
- δ Übergangsrelationen $\delta : X \to 2^{(Q \times Q)}$,
- $I \subseteq Q$ Startzustände,
- $F \subseteq Q$ akzeptierende Zustände.

WH: NFA – Beispiel

$A = (X, Q, \delta, \{0, 3\}, \{2, 3, 4\})$ mit

$$
\begin{aligned}
X &= \{a, b, c\} \\
Q &= \{0, 1, 2, 3, 4\} \\
\delta(a) &= \{(0, 0), (0, 1), (1, 3)\} \\
\delta(b) &= \{(0, 0), (1, 2)\} \\
\delta(c) &= \{(0, 3), (3, 3), (4, 1)\}
\end{aligned}
$$



Eigenschaften endlicher Automaten

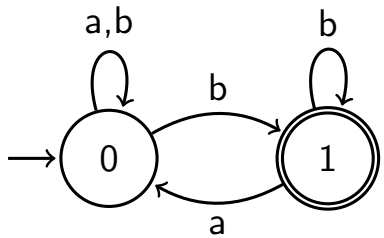
NFA $A = (X, Q, \delta, I, F)$ heißt

vollständig , falls $\forall a \in X \forall p \in Q : |\{q \mid (p, q) \in \delta(a)\}| \geq 1$

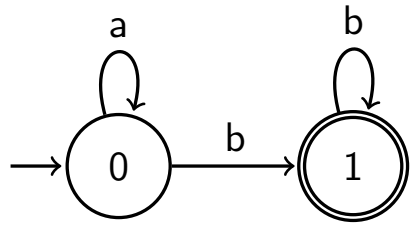
deterministisch (DFA) , falls

1. $|I| = 1$ und
2. $\forall a \in X \forall p \in Q : |\{q \mid (p, q) \in \delta(a)\}| \leq 1$

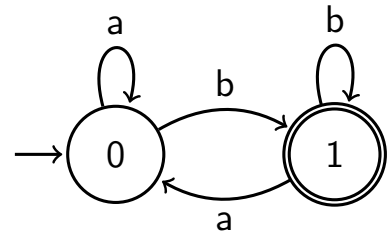
Beispiele:



vollständig
nicht deterministisch



nicht vollständig
deterministisch



vollständig
deterministisch

Isomorphie endlicher Automaten

Zwei NFA $A = (X, Q, \delta, I, F)$ und $B = (X, Q', \delta', I', F')$ sind genau dann isomorph,
wenn eine bijektive Funktion $h : Q \to Q'$ existiert, so dass

1. $\forall s \in Q((s \in I) \leftrightarrow (h(s) \in I'))$
2. $\forall f \in Q((f \in F) \leftrightarrow (h(f) \in F'))$
3. $\forall a \in X \; \forall (p, q) \in Q^2(((p, q) \in \delta(a)) \leftrightarrow ((h(p), h(q)) \in \delta'(a))$

Beispiel: NFA $A = (\{a, b\}, \{0, 1, 2\}, \delta, \{0\}, \{0\})$ mit
$\delta(a) = \{(0, 1), (1, 2), (2, 0)\}$ und $\delta(b) = \{(0, 2), (1, 0), (2, 1)\}$
und
NFA $B = (\{a, b\}, \{\alpha, \beta, \gamma\}, \delta, \{\beta\}, \{\beta\})$ mit
$\delta(a) = \{(\alpha, \beta), (\beta, \gamma), (\gamma, \alpha)\}$ und $\delta(b) = \{(\alpha, \gamma), (\beta, \alpha), (\gamma, \beta)\}$
sind isomorph durch die Bijektion
$h : \{0, 1, 2\} \to \{\alpha, \beta, \gamma\}$ mit $h(0) = \beta, h(1) = \gamma, h(2) = \alpha$.
(Graphisomorphie zwischen gerichteten Graphen mit markierten Kanten und Knoten)

Fakt

Isomorphie von NFA ist eine Äquivalenzrelation. *(ÜA)*

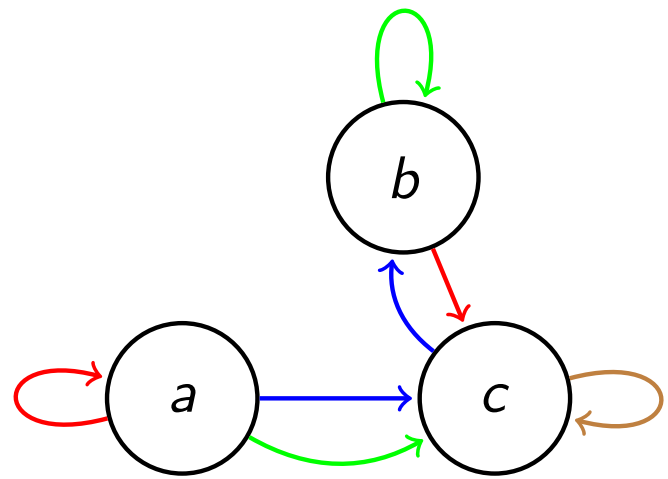
WH: Zweistellige Relationen

Verkettung der Relationen $R \subseteq M \times M$ und $S \subseteq M \times M$:

$$R \circ S = \{(a, b) \mid \exists c \in M : (a, c) \in R \wedge (c, b) \in S\}$$

Beispiel mit Darstellung als
gerichtetem Graphen $G = (V, E)$ mit $V = M$ und $E = R$

$$\begin{aligned} M &= \{a, b, c\} \\ R &= \{(a, a), (b, c)\} \\ S &= \{(a, c), (c, b)\} \\ R \circ S &= \{(a, c), (b, b)\} \\ S \circ R &= \{(c, c)\} \end{aligned}$$



in NFA:

- Übergangsrelation $\delta(a)$ zu jedem Symbol $a \in X$
- Verkettung der Symbole ab entspricht
  Verkettung der Relationen $\delta(a) \circ \delta(b)$

WH: Darstellung zweistelliger Relationen als Matrix

mit Booleschen Einträgen

$$
\begin{aligned}
M &= \{a, b, c\} \\
R &= \{(a, a), (b, c)\} \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\
S &= \{(a, c), (c, b)\} \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}
\end{aligned}
$$

Verkettung als Matrixmultiplikation mit Booleschen Operationen

$$
\begin{aligned}
R \circ S &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
S \circ R &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

Übergangsrelation auf Wörtern

Fortsetzung der Übergangsrelationen $\delta : X \to (Q \times Q)$
des NFA $A = (X, Q, \delta, I, F)$ auf Wörter $\delta^* : X^* \to (Q \times Q)$:

$$\begin{aligned} \delta^*(\varepsilon) &= \{(q,q) \mid q \in Q\} = I_Q \qquad \text{(Identität auf } Q\text{)} \\ \delta^*(wa) &= \delta^*(w) \circ \delta(a) \\ &= \{(p,q) \mid \exists r \in Q : (p,r) \in \delta^*(w) \wedge (r,q) \in \delta(a)\} \end{aligned}$$

für alle $w \in X^*, a \in X$

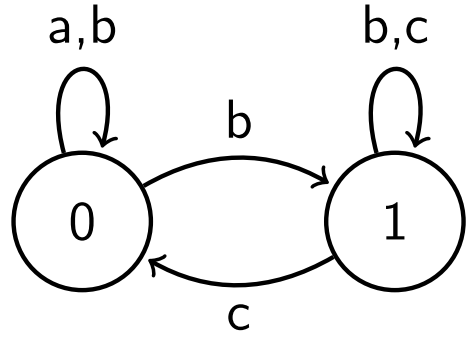
Es gilt also für alle

$$\begin{aligned} a \in X &: \delta^*(a) = \delta^*(\varepsilon) \circ \delta(a) = \delta(a) \\ w = w_1 \cdots w_n \in X^n &: \delta^*(w) = \delta(w_1) \circ \cdots \circ \delta(w_n) \end{aligned}$$

(Multiplikation der Matrizen $\delta(w_1), \ldots, \delta(w_n)$)

Beispiel

$A = (\{a, b, c\}, \{0, 1\}, \delta, I, F)$ mit
$\delta(a) = \{(0, 0)\}$ und
$\delta(b) = \{(0, 0), (0, 1), (1, 1)\}$
$\delta(c) = \{(1, 0), (1, 1)\}$



$$\delta(a) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \delta(b) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \delta(c) = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$$

$$\delta^*(ba) = \delta(b)\delta(a) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\delta^*(abb) = \delta(a)\delta(b)\delta(b) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

$$\delta^*(ac) = \delta(a)\delta(c) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Von einem NFA akzeptierte Wörter

NFA $A = (X, Q, \delta, I, F)$ akzeptiert ein Wort $w \in X^*$ genau dann, wenn

$$\delta^*(w) \cap (I \times F) \neq \emptyset$$

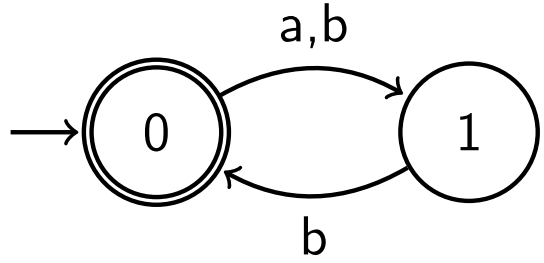
alternative Formulierung:
A akzeptiert w genau dann, wenn ein akzeptierender Weg für w in A existiert,
d.h. eine Folge $(q_0, \ldots, q_{|w|}) \in Q^*$ von Zuständen $q_i \in Q$, für die gilt:

1. $\forall i \in \{1, \ldots, |w|\} : (q_{i-1}, q_i) \in \delta(w_i)$,
2. $q_0 \in I$ (Startzustand) und
3. $q_{|w|} \in F$ (akzeptierender Zustand).

$$q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \cdots \xrightarrow{w_n} q_n \quad \text{mit} \quad q_0 \in I \text{ und } q_n \in F$$

Beispiel

$A = (\{a, b\}, \{0, 1\}, \delta, \{0\}, \{0\})$ mit
$\delta(a) = \{(0, 1)\}$, $\delta(b) = \{(0, 1), (1, 0)\}$



$$I \times F = \{0\} \times \{0\} = \{(0, 0)\} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\delta^*(abbb) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

A akzeptiert $abbb$ über den Weg $0 \xrightarrow{a} 1 \xrightarrow{b} 0 \xrightarrow{b} 1 \xrightarrow{b} 0 \in F$

$$\delta^*(bbb) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

A akzeptiert bbb nicht $0 \xrightarrow{b} 1 \xrightarrow{b} 0 \xrightarrow{b} 1 \notin F$

$$\delta^*(aaba) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

A akzeptiert $aaba$ nicht $0 \xrightarrow{a} 1 \xrightarrow{a} ?$

NFA-akzeptierbare Sprachen

NFA $A = (X, Q, \delta, I, F)$ akzeptiert die Sprache

$$L(A) = \{w \in X^* \mid \exists s \in I \; \exists f \in F : (s, f) \in \delta^*(w)\}$$

Beispiel: $A = (\{a, b\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 0), (2, 2)\}$ und $\delta(b) = \{(0, 0), (0, 1), (1, 2), (2, 2)\}$
akzeptiert $L(A) = \ldots$

Sprache $L \subseteq X^*$ heißt genau dann NFA-akzeptierbar, wenn ein NFA A mit $L = L(A)$ existiert.

Menge aller NFA-akzeptierbaren Sprachen: REC(NFA) (recognizable)

Beispiele:

- $L = \{w \in \{a, b\}^* \mid w$ enthält Infix aa oder Präfix $bba\}$ ist NFA-akzeptierbar.
- $L' = \{0^i 1 0^i \mid i \in \mathbb{N}\}$ ist nicht NFA-akzeptierbar (Nachweis später).

Beispiele NFA-akzeptierbarer Sprachen

- $L_1 = \{w \in \{0,1\}^* \mid |w|_0 \text{ ist ungerade } \}$
- $L_2 = \{w \in \{a,b\}^* \mid w \text{ beginnt mit } ab \text{ und endet mit } ba \}$
- $L_3 = L(G)$ mit $G = (\{S,T\},\{0,1\},P,S)$ mit $P = \{S \to 11T, T \to 0S, T \to 0\}$
- $L_4 = L(E)$ mit $E = (a+b)^*ab^*$
- $L_5 = L(F)$ mit $F = ((a+b)c)^*$

Äquivalenz endlicher Automaten

Zwei NFA A, B heißen genau dann äquivalent, wenn $L(A) = L(B)$.

Beispiel: Der NFA $A = (\{a, b\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1), (1, 1)\}$ und $\delta(b) = \{(1, 1), (1, 2)\}$

und der NFA $B = (\{a, b\}, \{0, 1, 2, 3\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1), (1, 1), (2, 1), (3, 3)\}$ und
$\delta(b) = \{(0, 3), (1, 2), (2, 2), (3, 3)\}$

sind äquivalent. Warum?

Fakt

Äquivalenz endlicher Automaten ist eine Äquivalenzrelation. *(ÜA)*

Fakt

Isomorphe endliche Automaten sind immer äquivalent. *(ÜA)*

(Achtung: Umkehrung gilt nicht, Gegenbeispiel oben)

Was bisher geschah

- Operationen auf Sprachen:
  - Mengenoperationen $\cup, \cap, \setminus, \overline{\ }, \Delta$
  - Sprachoperationen ${}^R, \circ, {}^*$
- reguläre Ausdrücke RegExp(X):

  IA: $\{\varepsilon, \emptyset\} \cup X \subseteq \text{RegExp}(X)$
  IS: $\forall E, F \in \text{RegExp}(X) : \{E + F, EF, E^*\} \subseteq \text{RegExp}(X)$

  Die Menge aller regulären Sprachen ist abgeschlossen unter $\cup, \circ, {}^*$
- Grammatiken, Chomsky-Hierarchie

endliche Automaten: NFA $A = (X, Q, \delta, I, F)$

- vollständige, deterministische NFA
- Isomorphie von NFA
- Akzeptanz von Wörtern durch NFA
- von NFA, DFA, akzeptierte Sprache $L(A)$
- Äquivalenz von NFA
- Menge REC(NFA) aller NFA-akzeptierbaren Sprachen

Vervollständigung von NFA

Motivation:
In vollständigen NFA existiert zu jedem Wort wenigstens ein Weg, Akzeptanz jedes Wortes durch Menge der mit dem letzten Symbol erreichten Zustände auf allen Wegen feststellbar.

Satz

Zu jedem NFA A existiert ein äquivalenter vollständiger NFA B.

Konstruktionsidee für $A = (X, Q_A, \delta_A, I, F)$ zu $B = (X, Q_B, \delta_B, I, F)$:

1. Hinzufügen eines zusätzlichen nichtakzeptierenden Zustandes e
   $Q_B = Q_A \cup \{e\}$
2. Hinzufügen zusätzlicher Kanten (p, e)
   $\forall a \in X : \delta_B(a) = \delta_A(a) \cup \{(p, e) \mid \neg\exists q \in Q_A : (p, q) \in \delta_A(a)\}$

Beispiel: $B = (\{a, b, c\}, \{0, 1, 2, 3, 4, 5\}, \delta, \{0\}, \{2, 4\})$ mit
$\delta(a) = \{(0, 0), (1, 2), (3, 4), (4, 4), (2, 5), (5, 5)\}$,
$\delta(b) = \{(0, 1), (0, 3), (1, 5), (2, 5), (3, 5), (4, 5), (5, 5)\}$
und $\delta(c) = \{(3, 0), (0, 5), (1, 5), (2, 5), (4, 5), (5, 5)\}$
(A enthält nur schwarze Elemente, Vervollständigung B auch alle roten)

Konstruktion von DFA aus NFA

Motivation: In DFA existiert zu jedem Wort w höchstens ein akzeptierender Weg, Feststellen der Akzeptanz in $|w|$ Schritten möglich.
Beispiel: $A = (\{a, b\}, \{0, 1, 2, 3, 4\}, \delta, \{0, 4\}, \{3\})$ mit
$\delta(a) = \{(0, 0), (4, 4)(0, 1), (1, 3), (3, 3)\}$ und
$\delta(b) = \{(0, 0), (4, 4), (4, 2), (2, 3), (3, 3)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q_A, \delta_A, I_A, F_A)$

gesucht: DFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit $L(A) = L(B)$

Potenzmengenkonstruktion: NFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit

$$
\begin{aligned}
Q_B &= 2^{Q_A} \quad \text{(Einschränkung auf von } I_B \text{ erreichbare genügt)} \\
I_B &= \{I_A\} \\
F_B &= \{M \subseteq Q_A \mid F_A \cap M \neq \emptyset\} \\
\forall a \in X: \ \delta_B(a) &= \{(M, N) \mid N = \{q \mid \exists p \in M : (p, q) \in \delta_A(a)\}\}
\end{aligned}
$$

Satz

Der nach der Potenzmengenkonstruktion aus dem NFA A konstruierte NFA B ist vollständig, deterministisch und äquivalent zu A.

Beispiel Potenzmengenkonstruktion

gegeben: NFA $A = (\{a, b\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1), (1, 1)\}$ und $\delta(b) = \{(1, 1), (1, 2)\}$

Potenzmengenkonstruktion: NFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit

$$
\begin{aligned}
Q_B &= 2^{Q_A} = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}\} \\
I_B &= \{\{0\}\} \\
F_B &= \{\{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}\} \\
\delta_B(a) &= \left\{ \begin{array}{l} (\emptyset,\emptyset), (\{0\},\{1\}), (\{1\},\{1\}), (\{1,2\},\{1\}), \\ (\{2\},\emptyset), (\{0,1\},\{1\}), (\{0,2\},\emptyset), (\{0,1,2\},\{1\}) \end{array} \right\} \\
\delta_B(b) &= \left\{ \begin{array}{l} (\emptyset,\emptyset), (\{0\},\emptyset), (\{1\},\{1,2\}), (\{1,2\},\{1,2\}), \\ (\{2\},\emptyset), (\{0,1\},\{1,2\}), (\{0,2\},\emptyset), (\{0,1,2\},\{1,2\}) \end{array} \right\}
\end{aligned}
$$

(rot: Einschränkung auf aus dem Startzustand erreichbare Zustände)
Einschränkung isomorph zum DFA $B' = (\{a, b\}, \{0, 1, 2, 3\}, \delta, \{0\}, \{2\})$
mit $\delta(a) = \{(0, 1), (1, 1), (2, 1), (3, 3)\}$ und
$\delta(b) = \{(0, 3), (1, 2), (2, 2), (3, 3)\}$
mit Isomorphismus $h : Q_B \to \{0, 1, 2, 3\}$ mit
$h(\{0\}) = 0$, $h(\{1\}) = 1$, $h(\{1, 2\}) = 2$, $h(\emptyset) = 3$

NFA für Spiegelung

Beispiel: DFA $A = (\{a, b\}, \{0, 1, 2, 3\}, \delta_A, \{0\}, \{2, 3\})$ mit $\delta_A(a) = \{(0, 1), (1, 3), (3, 3)\}$ und $\delta_A(b) = \{(1, 2), (2, 2)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q, \delta_A, I_A, F_A)$

gesucht: NFA B mit $L(B) = (L(A))^R$

Konstruktion: NFA $B = (X, Q, \delta_B, I_B, F_B)$ mit

1. $I_B = F_A$
2. $F_B = I_A$
3. $\forall a \in X : \delta_B(a) = \delta_A(a)^{-1} = \{(q, p) \mid (p, q) \in \delta_A(a)\}$

Fakt

Der oben definierte NFA B akzeptiert die Sprache $(L(A))^R$.

Die Menge REC(NFA) ist abgeschlossen unter Spiegelung.

NFA für Komplement NFA-akzeptierbarer Sprachen

Beispiel:
vollständiger DFA $A = (\{a, b\}, \{0, 1, 2, 3\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1), (1, 3), (2, 2), (3, 3)\}$ und
$\delta(b) = \{(0, 3), (1, 2), (2, 2), (3, 3)\}$

allgemeines Verfahren:

gegeben: vollständiger DFA $A = (X, Q, \delta, I, F)$
(falls notwendig vorher Potenzmengenkonstruktion)

gesucht: NFA B mit $L(B) = \overline{L(A)}$

Konstruktion: NFA $B = (X, Q, \delta, I, Q \setminus F)$

Satz

Der oben definierte NFA B akzeptiert die Sprache $\overline{L(A)}$.

Die Menge REC(NFA) ist abgeschlossen unter Komplementbildung.

NFA für Vereinigung NFA-akzeptierbarer Sprachen

Beispiel: NFA $A = (\{a, b\}, \{0, 1\}, \delta_A, \{0\}, \{0\})$ mit
$\delta_A(a) = \{(0, 1), (1, 0)\}$ und $\delta_A(b) = \emptyset$
NFA $B = (\{a, b\}, \{2, 3, 4\}, \delta_B, \{2\}, \{4\})$ mit
$\delta_B(a) = \{(2, 3), (4, 3)\}$ und $\delta_B(b) = \{(3, 4)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q_A, \delta_A, I_A, F_A)$ und
NFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit $Q_A \cap Q_B = \emptyset$
(Zustände disjunkt umbenennen, falls notwendig)

gesucht: NFA C mit $L(C) = L(A) \cup L(B)$

Konstruktion: NFA $C = (X, Q_A \cup Q_B, \delta_C, I_A \cup I_B, F_A \cup F_B)$ mit
$\forall a \in X : \delta_C(a) = \delta_A(a) \cup \delta_B(a)$

Satz

Der oben definierte NFA C akzeptiert die Sprache $L(A) \cup L(B)$.

Die Menge REC(NFA) ist abgeschlossen unter Vereinigung.

NFA für Schnitt NFA-akzeptierbarer Sprachen

WH (deMorgan):
Für alle Mengen M, N gilt $M \cap N = \overline{\overline{M} \cup \overline{N}}$

bisher: Konstruktionen für

1. NFA $A \mapsto$ vollst. DFA A' mit $L(A') = L(A)$
2. vollst. DFA $A \mapsto$ vollst. DFA A' mit $L(A') = \overline{L(A)}$
3. NFA $A, B \mapsto$ NFA C mit $L(C) = L(A) \cup L(B)$

Zu gegebenen NFA A und B lässt sich damit auch ein NFA C konstruieren mit

$$L(C) = L(A) \cap L(B) = \overline{\overline{L(A)} \cup \overline{L(B)}}$$

Die Menge REC(NFA) ist also abgeschlossen unter Schnitt.

Diese Konstruktion ist aber meist aufwendig. Es geht besser.

Was bisher geschah

- Operationen auf Sprachen:
  - Mengenoperationen $\cup, \cap, \setminus, \overline{\ }, \Delta$
  - Sprachoperationen ${}^R, \circ, {}^*$
- reguläre Ausdrücke RegExp(X):

  IA: $\{\varepsilon, \emptyset\} \cup X \subseteq \text{RegExp}(X)$

  IS: $\forall E, F \in \text{RegExp}(X) : \{E + F, EF, E^*\} \subseteq \text{RegExp}(X)$

  Die Menge aller regulären Sprachen ist abgeschlossen unter $\cup, \circ, {}^*$
- Grammatiken, Chomsky-Hierarchie

endliche Automaten: NFA $A = (X, Q, \delta, I, F)$

- vollständige, deterministische NFA
- Akzeptanz von Wörtern durch NFA
- von NFA, DFA, akzeptierte Sprache $L(A)$
- Menge REC(NFA) aller NFA-akzeptierbare Sprachen
- Zu jedem NFA existiert ein äquivalenter vollständiger NFA. (Vervollständigung)
- Zu jedem NFA existiert ein äquivalenter (vollständiger) DFA. (Potenzmengenkonstruktion)
- REC(NFA) ist abgeschlossen unter Mengenoperationen $\cup, \overline{\ }, \cap, \setminus, \Delta$

DFA für Schnitt – Produktkonstruktion

Beispiel: DFA $A = (\{a, b\}, \{0, 1\}, \delta_A, \{0\}, \{1\})$ mit
$\delta_A(a) = \{(0, 0)\}$ und $\delta_A(b) = \{(0, 1), (1, 1)\}$
DFA $B = (\{a, b\}, \{2, 3\}, \delta_B, \{2\}, \{3\})$ mit
$\delta_B(a) = \{(2, 3), (3, 3)\}$ und $\delta_B(b) = \{(3, 3)\}$
allgemeines Verfahren:

gegeben: DFA $A = (X, Q_A, \delta_A, I_A, F_A)$, DFA $B = (X, Q_B, \delta_B, I_B, F_B)$

gesucht: DFA $C = (X, Q_C, \delta_C, I_C, F_C)$ mit $L(C) = L(A) \cap L(B)$

Konstruktion: (Produktautomat)
DFA $C = (X, Q_A \times Q_B, \delta_C, I_A \times I_B, F_A \times F_B)$ mit

$$\forall a \in X : \ \delta_C(a) = \{((p, q), (r, s)) \mid (p, r) \in \delta_A(a) \wedge (q, s) \in \delta_B(a)\}$$

Satz

Der oben definierte DFA C akzeptiert die Sprache $L(A) \cap L(B)$.

Modifikation:
dieselbe Konstruktion des DFA C aus vollständigen DFA A und B

- mit akzeptierenden Zuständen $F_C = F_A \times F_B$
  akzeptiert $L(A) \cap L(B)$
- mit akzeptierenden Zuständen $F_C = (F_A \times Q_B) \cup (Q_A \times F_B)$
  akzeptiert $L(A) \cup L(B)$

NFA mit ε-Übergängen

ε-NFA: NFA ohne Einschränkung $\delta(\varepsilon) = I_Q$, stattdessen $I_Q \subseteq \delta(\varepsilon)$
(zusätzliche Kanten im Automatengraphen mit Beschriftung ε)

ε-NFA vereinfachen oft die Modellierung von Sprachen und Abläufen.

Beispiel: $A = (\{a, b, c\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 0)\}$, $\delta(b) = \{(1, 1)\}$, $\delta(c) = \{(2, 2)\}$, $\delta(\varepsilon) = \{(0, 1), (1, 2)\}$

Akzeptanz von Wörtern durch ε-NFA analog zu NFA:
ε-NFA $A = (X, Q, \delta, I, F)$ akzeptiert ein Wort $w \in X^*$ genau dann, wenn
$\delta^*(w) \cap (I \times F) \neq \emptyset$.

$$\delta^*(w) = \delta^*(\varepsilon) \circ \delta(w_1) \circ \delta^*(\varepsilon) \circ \cdots \circ \delta^*(\varepsilon) \circ \delta(w_n) \circ \delta^*(\varepsilon)$$

akzeptierender Pfad für w in ε-NFA A:
$q_0 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_i \xrightarrow{w_1} q_{i+1} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_k \xrightarrow{w_{|w|}} q_{k+1} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_n$

Der ε-NFA A im Beispiel oben akzeptiert die Sprache $L(A) = a^* b^* c^*$.

Fakt

Zu jedem NFA A existiert ein äquivalenter ε-NFA B mit genau einem Startzustand und genau einem akzeptierenden Zustand.

Eliminierung der ε-Übergänge aus ε-NFA

$\delta^*(\varepsilon)$ ist der transitive Abschluss der Relation $\delta(\varepsilon)$
ε-Hülle eines Zustandes $q \in Q$: $H_\varepsilon(q) = \{p \mid (q,p) \in \delta^*(\varepsilon)\}$

allgemeines Verfahren:

gegeben: ε-NFA $A = (X, Q, \delta, I, F)$ mit $\delta(\varepsilon) \supseteq I_Q$

gesucht: NFA B ohne ε-Übergänge mit $L(B) = L(A)$

Konstruktion (ε-Eliminierung): $B = (X, Q, \delta', I', F)$ mit

$$I' = \bigcup_{q \in I} H_\varepsilon(q)$$

$$\forall a \in X: \ \delta'(a) = \delta(a) \cup \left\{(p, q') \mid (p,q) \in \delta(a) \wedge q' \in H_\varepsilon(q)\right\}$$

Satz

Der durch ε-Eliminierung aus dem ε-NFA A konstruierte NFA B enthält keine ε-Übergänge und ist äquivalent zu A.

NFA für Verkettung NFA-akzeptierbarer Sprachen

Beispiel:
NFA $A = (\{a, b\}, \{0\}, \delta_A, \{0\}, \{0\})$ mit $\delta_A(a) = \emptyset$ und $\delta_A(b) = \{(0, 0)\}$
NFA $B = (\{a, b\}, \{1, 2, 3\}, \delta_B, \{1, 3\}, \{2\})$ mit
$\delta_B(a) = \{(3, 2), (3, 3)\}$ und $\delta_B(b) = \{(1, 2)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q_A, \delta_A, I_A, F_A)$ und
NFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit $Q_A \cap Q_B = \emptyset$
gesucht: NFA C mit $L(C) = L(A) \circ L(B)$
Konstruktion (eines ε-NFA C', anschließend ε-Eliminierung zu NFA C):

1. $(A' \equiv A \wedge |F_{A'}| = 1)$: ε-NFA $A' = (X, Q_A \cup \{f_A\}, \delta'_A, I_A, \{f_A\})$ mit
   $\forall a \in X : \delta'_A(a) = \delta_A(a)$ und $\delta'_A(\varepsilon) = \delta_A(\varepsilon) \cup \{(q, f_A) \mid q \in F_A\}$
2. $(B' \equiv B \wedge |I_{B'}| = 1)$: ε-NFA $B' = (X, Q_B \cup \{s_B\}, \delta'_B, \{s_B\}, F_B)$ mit
   $\forall a \in X : \delta'_B(a) = \delta_B(a)$ und $\delta'_B(\varepsilon) = \delta_B(\varepsilon) \cup \{(s_B, q) \mid q \in I_B\}$
3. $(\circ)$: ε-NFA $C' = (X, Q_A \cup Q_B \cup \{f_A, s_B\}, \delta_{C'}, I_A, F_B)$ mit
   $\forall a \in X : \delta_{C'}(a) = \delta'_A(a) \cup \delta'_B(a)$, $\delta_{C'}(\varepsilon) = \delta'_A(\varepsilon) \cup \delta'_B(\varepsilon) \cup \{(f_A, s_B)\}$

Fakt

Der oben definierte ε-NFA C' akzeptiert die Sprache $L(A) \circ L(B)$.

Folgerung: Die Menge REC(NFA) ist abgeschlossen unter Verkettung.

NFA für iterierte Verkettung

Beispiel: NFA $A = (\{a, b\}, \{0, 1, 2, 3\}, \delta_A, \{0\}, \{3\})$ mit
$\delta_A(a) = \{(0, 1), (2, 3)\}$ und $\delta_A(b) = \{(1, 2)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q, \delta_A, I, F)$

gesucht: NFA B mit $L(B) = L(A)^*$

Konstruktion (ε-NFA B', anschließend ε-Eliminierung zu NFA B):

1. $(A' \equiv A \wedge |I_{A'}| = |F_{A'}| = 1)$:
   ε-NFA $A' = (X, Q_A \cup \{s_A, f_A\}, \delta'_A, \{s_A\}, \{f_A\})$ mit
   $\forall a \in X : \delta'_A(a) = \delta_A(a)$ und
   $\delta'_A(\varepsilon) = \{(s_A, q) \mid q \in I_A\} \cup \{(q, f_A) \mid q \in F_A\}$
2. $(^*)$: ε-NFA $B' = (X, Q_A \cup \{s_A, f_A\}, \delta_B, \{s_A\}, \{f_A\})$ mit
   $\forall a \in X : \delta_{B'}(a) = \delta'_A(a)$ und
   $\delta_{B'}(\varepsilon) = \delta'_A(\varepsilon) \cup \{(s_A, f_A), (f_A, s_A)\}$

Fakt

Der oben definierte ε-NFA B' akzeptiert die Sprache $(L(A))^*$.

Folgerung: REC(NFA) ist abgeschlossen unter iterierter Verkettung.

Abschlusseigenschaften der Menge REC(NFA)

$$\text{REC(NFA)} = \text{REC(DFA)} = \text{REC}(\varepsilon\text{-NFA})$$

Sind L und L' NFA-akzeptierbare Sprachen, dann sind auch die Sprachen

- $L \cup L'$ (Vereinigung)
- $\overline{L}$ (Komplement)
- $L \cap L'$ (Schnitt)
- $L \circ L'$ (Verkettung)
- L^n, L^* (iterierte Verkettung)
- L^R (Spiegelung)

NFA-akzeptierbar.

dieselbe Aussage kürzer: REC(NFA) ist abgeschlossen unter

- Mengenoperationen: $\cup, \cap, \overline{\ }, \setminus, \Delta$
- Sprachoperationen $\circ$, *, ^{R}

Regulärer Ausdruck $\rightarrow$ NFA

Fakt

Jede durch einen regulären Ausdruck definierte Sprache ist NFA-akzeptierbar.

WH: REC(NFA) ist (u.A.) abgeschlossen unter $\cup, \circ, *$

induktiver Beweis:
(Induktion über die Struktur des regulären Ausdruckes)

IA $L(\emptyset)$, $L(\varepsilon)$ und $L(a)$ für alle $a \in X$ sind NFA-akzeptierbar

IS sind $L(E)$ und $L(F)$ NFA-akzeptierbar, dann sind auch

- $L(E + F) = L(E) \cup L(F)$,
- $L(EF) = L(E) \circ L(F)$ und
- $L(E^*) = L(E)^*$

NFA-akzeptierbar
(wegen Abgeschlossenheit von REC(NFA) unter $\cup, \circ, \ ^*$)

Dieser Beweis enthält ein Verfahren zur Konstruktion des NFA.

Beispiel: $ab^* + b$

DFA $\rightarrow$ regulärer Ausdruck

Beispiel: $A = (\{a, b\}, \{1, 2\}, \delta, \{1\}, \{1\})$ mit
$\delta(a) = \{(1, 2)\}$ und $\delta(b) = \{(1, 1)(2, 1)\}$
gegeben: DFA $A = (X, Q, \delta, I, F)$ mit $Q = \{1, \ldots, n\}$ und $I = \{1\}$
schrittweise Konstruktion regulärer Ausdrücke
(Dynamische Programmierung):

$R(k, i, j)$ beschreibt die Menge aller Wörter, für die ein Pfad
von i zu j nur über (Zwischen-)Zustände $\leq k$ existiert

$$
\begin{aligned}
R(0, i, j) &= \begin{cases} \{a \in X \mid (i, j) \in \delta(a)\} & \text{falls } i \neq j \\ \varepsilon + \{a \in X \mid (i, j) \in \delta(a)\} & \text{falls } i = j \end{cases} \\
R(k+1, i, j) &= R(k, i, j) + (R(k, i, k+1) R(k, k+1, k+1)^* R(k, k+1, j))
\end{aligned}
$$

Fakt

Für den so konstruierten regulären Ausdruck $E = \sum_{f \in F} R(n, 1, f)$
gilt $L(E) = L(A)$.

NFA und reguläre Ausdrücke

Schon gezeigt:

Fakt

Jede NFA-akzeptierbare Sprache wird durch einen regulären Ausdruck definiert.

Fakt

Jede durch einen regulären Ausdruck definierte Sprache ist NFA-akzeptierbar.

Das ergibt zusammen:

Satz

Eine Sprache ist genau dann NFA-akzeptierbar ($\in$ REC(NFA)), wenn sie regulär (durch einen regulären Ausdruck definiert) ist.

(REC(NFA) ist genau die Menge aller regulären Sprachen.)

Folgerung aus Satz und Abschlusseigenschaften von REC(NFA): Jeder erweiterte reguläre Ausdruck ist äquivalent zu einem (einfachen) regulären Ausdruck.

Was bisher geschah

Chomsky-Hierarchie für Sprachen:

- $\mathcal{L}_0$ Menge aller durch (beliebige) Grammatiken beschriebenen Sprachen
- $\mathcal{L}_1$ Menge aller monotonen (Kontextsensitive) Sprachen
- $\mathcal{L}_2$ Menge aller kontextfreien Sprachen
- $\mathcal{L}_3 = \mathrm{REG}$ Menge aller regulären Sprachen

- $\mathrm{REC(NFA)} = \mathrm{REC(DFA)} = \mathrm{REC}(\varepsilon - \mathrm{NFA})$
- REC(NFA) ist abgeschlossen unter $\cup, \cap, \overline{\ }, \circ, \ ^*, \ ^R$
- REC(NFA) = Menge aller regulären (durch reguläre Ausdrücke darstellbare) Sprachen
  (Konstruktion RegEx $\rightarrow$ NFA)

DFA-Minimierung

Ein vollständiger DFA A heißt genau dann minimal für die Sprache $L(A)$, wenn kein vollständiger DFA B mit $L(A) = L(B)$ und weniger Zuständen als A existiert.

Beispiel: Ist der folgende vollständige DFA minimal?
$A = (\{a, b\}, \{0, 1, 2, 3, 4\}, \delta, \{0\}, \{4\})$ mit
$\delta(a) = \{(0, 1), (1, 4), (2, 3), (3, 4), (4, 4)\}$ und
$\delta(b) = \{(0, 2), (1, 2), (2, 2), (3, 0), (4, 4)\}$

Idee (für vollständigen DFA $A = (X, Q, \delta, I, F)$):

- für jedes Paar (p, q) von Zuständen in Q:
  Suche nach einem (kürzesten) unterscheidenden Wort
- Zustände sind äquivalent, wenn kein unterscheidendes Wort existiert
- Zerlegung von Q in Äquivalenzklassen von Zuständen
- diese Klassen sind die Zustände des minimalen DFA.

Äquivalente Zustände in DFA

Ziel: Konstruktion eines minimalen DFA B mit $L(B) = L(A)$
zu gegebenem DFA A

Beispiel: $A' = (\{a, b\}, \{0, 1, 2, 3\}, \delta, \{0\}, \{3\})$ mit
$\delta(a) = \{(0, 1)\}$ und $\delta(b) = \{(0, 2), (1, 3), (2, 3)\}$
(vollständig ?)

Definition:
Jedes Paar aus DFA $A = (X, Q, \delta, I, F)$ und Zustand $q \in Q$
definiert den DFA $A_q = (X, Q, \delta, \{q\}, F)$
und damit die Sprache $L(A_p) \subseteq X^*$

Zustände $p, q \in Q$ heißen genau dann
äquivalent in A ($p \sim_A q$), wenn $L(A_p) = L(A_q)$ gilt.

Beispiel (oben):
$1 \sim_A 2$, weil $L(A_1) = L(A_2) = \{b\}$

Ein Wort $w \in X^*$ unterscheidet die Zustände $p \in Q$ und $q \in Q$
(beweist $p \not\sim_A q$) gdw. $w \in L(A_p) \Delta L(A_q)$

Beispiel (oben): $1 \not\sim_A 3$, weil $\varepsilon \in L(A_3) \setminus L(A_1)$

Algorithmus zur Bestimmung äquivalenter Zustände

Fakt

Falls das Wort $w \in X^*$ die Zustände $p', q' \in Q$ im DFA $A = (X, Q, \delta, I, F)$ unterscheidet und für ein $a \in X$ gilt $(p, p') \in \delta(a)$ und $(q, q') \in \delta(a)$, dann unterscheidet das Wort $aw \in X^*$ die Zustände $p, q \in Q$ in A.

Induktive Berechnung einer Folge T_i von Mengen von unterscheidbaren Zustandspaaren (als Zweiermengen):

$$
\begin{aligned}
T_0 &= \{\{p, q\} \mid p \in F, q \notin F\} \\
T_{i+1} &= T_i \cup \left\{ \{p, q\} \mid \exists a \in X \; \exists p', q' \in Q : \begin{pmatrix} \{p', q'\} \in T_i \\ \wedge (p, p') \in \delta(a) \\ \wedge (q, q') \in \delta(a) \end{pmatrix} \right\}
\end{aligned}
$$

(T_i: Menge aller durch ein w mit $|w| \leq i$ unterscheidbaren Paare)
Für jeden vollständigen DFA A existiert ein $n \in \mathbb{N}$ mit $T_n = T_{n+1}$.
Für jedes Paar $\{p, q\} \notin T_n$ sind p und q äquivalent in A.

Alternative Darstellung: Folge von Äquivalenzrelationen $\sim_i \subseteq Q^2$ mit
$p \sim_i q$ gdw. $\forall w \in X^* : |w| \leq i \rightarrow (w \in A_p \leftrightarrow w \in A_q)$
(auch als Folge der Zerlegungen in $\sim_i$-Äquivalenzklassen üblich)

Minimalautomat

gegeben: vollständiger DFA $A = (X, Q_A, \delta_A, I_A, F_A)$
Beispiel: vollst. DFA $A = (\{a, b\}, \{0, 1, 2, 3, 4\}, \delta, \{0\}, \{4\})$ mit
$\delta(a) = \{(0, 1), (1, 4), (2, 3), (3, 4), (4, 4)\}$ und
$\delta(b) = \{(0, 2), (1, 2), (2, 2), (3, 0), (4, 4)\}$

Minimalautomat für $L(A)$:
DFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit

$$\begin{aligned} Q_B &= \{[q]_A \mid q \in Q\} \quad \text{Äquivalenzklassen in } A \\ \forall a \in X : \delta_B(a) &= \{([p]_A, [q_A]) \mid (p, q) \in \delta_A(a)\} \\ I_B &= \{[q]_A \mid q \in I_A\} \\ F_B &= \{[q]_A \mid q \in F_A\} \end{aligned}$$

Satz

Zu jeder NFA-akzeptierbaren Sprache $L \subseteq X^*$ existiert ein (bis auf Isomorphie) eindeutiger vollständiger DFA A mit $L = L(A)$ und einer minimalen Anzahl von Zuständen (Minimalautomat für L).

Entscheidung der Äquivalenz von NFA

Wiederholung: NFA A und B äquivalent gdw. $L(A) = L(B)$

Eingabe:	NFA A, B	
Ausgabe:	1 (ja),	falls A und B äquivalent
	0 (nein),	sonst

Algorithmus:

1. Konstruktion vollständiger DFA A' und B' mit $L(A') = L(A)$ und $L(B') = L(B)$ (Potenzmengenkonstruktion),
2. Konstruktion der Minimalautomaten A'' zu A' und B'' zu B'
3. Test, ob A'' und B'' isomorph sind (Isomorphietest für Graphen).

Fakt

Die NFA A und B sind genau dann äquivalent, wenn die Automaten A'' und B'' isomorph sind.

Entscheidung der Äquivalenz regulärer Ausdrücke

Eingabe: Reguläre Ausdrücke E, F

Ausgabe: 1 (ja), falls E und F äquivalent ($L(E) = L(F)$)
0 (nein), sonst

Algorithmus:

1. Konstruktion der NFA A_E und A_F
   mit $L(A_E) = L(E)$ und $L(A_F) = L(F)$,
2. Konstruktion der DFA A'_E und A'_F mit
   $L(A'_E) = L(E)$ und $L(A'_F) = L(F)$
   (Potenzmengenkonstruktion),
3. Konstruktion der Minimalautomaten
   A''_E zu A'_E und A''_F zu A'_F
4. Test, ob A''_E und A''_F isomorph sind
   (Isomorphietest für Graphen)

Fakt

Die regulären Ausdrücke E und F sind genau dann äquivalent, wenn die Automaten A''_E und A''_F isomorph sind.

Beispiel: $(ab)^+a$ und $a(ba)^*ba$

Entscheidung der Äquivalenz von NFA

Wiederholung: NFA A und B äquivalent gdw. $L(A) = L(B)$

Eingabe:	NFA A, B	
Ausgabe:	1 (ja),	falls A und B äquivalent
	0 (nein),	sonst

Algorithmus:

1. Konstruktion vollständiger DFA A' und B' mit $L(A') = L(A)$ und $L(B') = L(B)$ (Potenzmengenkonstruktion),
2. Konstruktion der Minimalautomaten A'' zu A' und B'' zu B'
3. Test, ob A'' und B'' isomorph sind (Isomorphietest für Graphen).

Fakt

Die NFA A und B sind genau dann äquivalent, wenn die Automaten A'' und B'' isomorph sind.

NFA und reguläre Grammatiken

Beispiel: NFA $A = (\{a, b\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1)\}$ und $\delta(b) = \{(1, 0), (1, 2)\}$

reguläre Grammatik $G = (\{S, B\}, \{a, b\}, P, S)$ mit
$P = \{S \to aB, B \to bS, B \to b\}$

definieren beide die Sprache
$L(A) = L(G) = L((ab)^+) = \{(ab)^n \mid n \in \mathbb{N} \setminus \{0\}\}$

Satz

REC(NFA) $= \mathcal{L}_3$

dieselbe Aussage ausführlicher:
Eine Sprache $L \subseteq X^*$ ist genau dann NFA-akzeptierbar, wenn $L \setminus \{\varepsilon\}$ von einer regulären Grammatik (Chomsky-Typ 3) erzeugt wird.

Konstruktion: reguläre Grammatik $\rightarrow$ NFA

gegeben: reguläre Grammatik $G = (N, X, P, S)$ (Chomsky 3)
Konstruktion: NFA $A = (X, Q, \delta, I, F)$ mit

$$\begin{aligned} Q &= N \cup \{f\} \quad \text{mit } f \notin N \\ I &= \{S\} \\ F &= \{f\} \\ \text{für jedes } a \in X\text{: } \delta(a) &= \{(A, B) \mid (A \rightarrow aB) \in P\} \\ &\quad \cup \{(A, f) \mid (A \rightarrow a) \in P\} \end{aligned}$$

Beispiel: Konstruktion ergibt für die reguläre Grammatik
$G = (\{S, B\}, \{0, 1\}, P, S)$ mit
$P = \{S \rightarrow 0B, B \rightarrow 0B, B \rightarrow 1S, B \rightarrow 0\}$
den NFA $A = (\{0, 1\}, \{S, B, f\}, \delta, \{S\}, \{f\})$ mit
$\delta(0) = \{(S, B), (B, B), (B, f)\}$ und $\delta(1) = \{(B, S)\}$

Fakt (REC(NFA) $\subseteq \mathcal{L}_3$): Für den wie oben zur Grammatik G konstruierten NFA A gilt $L(G) = L(A)$.

Konstruktion: NFA $\rightarrow$ reguläre Grammatik

gegeben: NFA $A = (X, Q, \delta, I, F)$
Konstruktion: reguläre Grammatik $G = (N, X, P, S)$ mit

$$
\begin{aligned}
N &= Q \,\dot\cup\, \{S\} \quad (\text{mit } S \notin Q) \\
P &= \bigcup_{a \in X} \{p \to aq \mid (p,q) \in \delta(a)\} \\
&\quad \cup \bigcup_{a \in X} \{S \to aq \mid \exists s \in I : (s,q) \in \delta(a)\} \\
&\quad \cup \bigcup_{a \in X} \{p \to a \mid \exists f \in F : (p,f) \in \delta(a)\} \\
&\quad \cup \bigcup_{a \in X} \{S \to a \mid \exists s \in I \; \exists f \in F : (s,f) \in \delta(a)\}
\end{aligned}
$$

Beispiel: $A = (\{a,b\}, \{0,1,2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0,1),(1,2)\}$ und $\delta(b) = \{(1,1),(2,1)\}$
konstruierte Grammatik $G = (\{S,0,1,2\}, \{a,b\}, P, S)$ mit
$P = \{0 \to a1, 1 \to b1, 1 \to a2, 2 \to b1, S \to a1, 1 \to a\}$

Fakt ($\mathcal{L}_3 \subseteq$ REC(NFA)): Für die wie oben aus dem NFA A konstruierte Grammatik G gilt $L(G) = L(A) \setminus \{\varepsilon\}$.

Ergebnisse über reguläre Sprachen

Für jede Sprache $L \subseteq X^*$ sind die folgenden Aussagen äquivalent:

- $L = L(E)$ für einen regulären Ausdruck E.
- L hat den Chomsky-Typ 3.
- Es existiert eine reguläre Grammatik G mit $L \setminus \{\varepsilon\} = L(G)$.
- Es existiert eine reguläre Grammatik G mit ε-Regel und $L = L(G)$.
- L ist NFA-akzeptierbar.
- Es existiert ein vollständiger DFA A mit $L = L(A)$.

Die Menge aller Sprachen vom Chomsky-Typ 3 ist abgeschlossen unter

- Mengenoperationen: $\cup, \cap, \overline{\ }, \setminus, \Delta$
- Sprachoperationen: $\circ$, *, ^{R}

Algorithmische Lösungen für reguläre Sprachen

alle Sprachen in Eingaben endlich beschrieben, z.B. als NFA

Wortproblem Eingabe: (L, w),
Ausgabe: ja, falls $w \in L$, sonst nein
(Suche nach mit w markiertem Pfad im Automatengraphen)

Leerheit Eingabe: L, Ausgabe: ja, falls $L = \emptyset$, sonst nein
(Erreichbarkeit akzeptierender Zustände in NFA für L)

Vollständigkeit Eingabe: L, Ausgabe: ja, falls $L = X^*$, sonst nein
(Erreichbarkeit im Automaten für $\overline{L}$)

(Un-)Endlichkeit Eingabe: L, Ausgabe: ja, falls L (un-)endlich, sonst nein
(Suche nach einem akzeptierenden Weg mit Schleife)

Inklusion Eingabe: L_1, L_2,
Ausgabe: ja, falls $L_1 \subseteq L_2$, sonst nein
(Test $L_1 \cap \overline{L_2} = \emptyset$)

Gleichheit Eingabe: L_1, L_2,
Ausgabe: ja, falls $L_1 = L_2$, sonst nein
(isomorphe Minimalautomaten oder Test $L_1 \Delta L_2 \stackrel{?}{=} \emptyset$)

Disjunktheit Eingabe: L_1, L_2,
Ausgabe: ja, falls $L_1 \cap L_2 = \emptyset$, sonst nein

Anwendung von NFA und regulären Sprachen

- Modellierung vom Verhalten von Zustandsübergangssystemen, z.B.
  - reale Geräte und Anlagen
  - Softwaresysteme

  Überprüfung, ob Zustandsübergangssystem bestimmte Anforderungen erfüllt
- Beschreibung der Syntax von Programmiersprachen z.B.
  - endlicher Mengen (z.B. Mengen aller Schlüsselwörter)
  - Folgen von Zeichen und Zeichenketten
  - Zahlendarstellungen
- lexikalische Analyse im Compiler
- Suche nach Zeichenketten in Texten (string matching)
- Textverarbeitung, z.B. Wortvervollständigung