

Abschlusseigenschaften

Durchschnitt und Vereinigung (für DFA)

$$\text{zu } \mathcal{A}_1 = (\Sigma, Q^{(1)}, q_0^{(1)}, \delta^{(1)}, A^{(1)})$$

$$\mathcal{A}_2 = (\Sigma, Q^{(2)}, q_0^{(2)}, \delta^{(2)}, A^{(2)})$$

Produktautomat $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$ mit

$$Q := Q^{(1)} \times Q^{(2)}$$

$$q_0 := (q_0^{(1)}, q_0^{(2)})$$

$$\delta((q_1, q_2), a) := (\delta^{(1)}(q_1, a), \delta^{(2)}(q_2, a))$$

simuliert $\mathcal{A}_1/\mathcal{A}_2$ parallel in erster/zweiter Komponente

$$A := \begin{cases} A^{(1)} \times A^{(2)} & \text{für Durchschnitt} \\ (A^{(1)} \times Q^{(2)}) \cup (Q^{(1)} \times A^{(2)}) & \text{für Vereinigung} \end{cases}$$

Abschlusseigenschaften

Konkatenation (für NFA)

$$\text{aus NFA } \mathcal{A}_1 = (\Sigma, Q^{(1)}, q_0^{(1)}, \Delta^{(1)}, A^{(1)})$$

$$\mathcal{A}_2 = (\Sigma, Q^{(2)}, q_0^{(2)}, \Delta^{(2)}, A^{(2)})$$

mit $Q^{(1)} \cap Q^{(2)} = \emptyset$ und $q_0^{(1)} \notin A^{(1)}$ (*)

gewinne **Hintereinanderschaltung** als NFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$

$$Q := Q^{(1)} \cup Q^{(2)}$$

$$q_0 := q_0^{(1)}$$

$$A := A^{(2)}$$

$$\Delta := \Delta^{(1)} \cup \Delta^{(2)} \cup \Delta^{(1) \rightarrow (2)}$$

$$\Delta^{(1) \rightarrow (2)} := \{(q, a, q_0^{(2)}) : q \in Q^{(1)}, (q, a, q') \in \Delta^{(1)} \text{ für ein } q' \in A^{(1)}\}$$

(*): was ist andernfalls zu tun?

Abschlusseigenschaften

Korollar 2.2.16

alle regulären Sprachen von NFA/DFA erkannt

per Induktion über reguläre Ausdrücke zeige:

$(\forall \alpha \in \text{REG}(\Sigma)) L(\alpha)$ Automaten-erkennbar

Induktionsanfang: $\alpha = \emptyset$ und $\alpha = \mathbf{a}$ für $a \in \Sigma$.

$L(\emptyset) = \emptyset$ und $L(\mathbf{a}) = \{a\}$ Automaten-erkennbar. (Übung!)

Induktionsschritte: von α_1, α_2 zu $\begin{cases} \alpha_1 + \alpha_2, \\ \alpha_1 \alpha_2, \\ \alpha_1^* \end{cases}$

wenn $L(\alpha_1), L(\alpha_2)$ Automaten-erkennbar sind, so auch

$$\begin{cases} L(\alpha_1 + \alpha_2) = L(\alpha_1) \cup L(\alpha_2) \\ L(\alpha_1 \alpha_2) = L(\alpha_1) \cdot L(\alpha_2) \\ L(\alpha_1^*) = (L(\alpha_1))^* \end{cases}$$

Satz von Kleene

→ Abschnitt 2.3

Satz 2.3.1 (Kleene's Theorem)

L regulär $\Leftrightarrow L$ DFA/NFA-erkennbar

reguläre Ausdrücke	—	Automaten-Berechnung
erzeugen (Sprache)	—	erkennen (Zugehörigkeit)
deskriptiv	—	prozedural
Syntax	—	Semantik

Übersicht

reguläre Σ -Sprachen

NFA/DFA erkennbare Σ -Sprachen

$L = L(\alpha)$: $\alpha \in \text{REG}(\Sigma)$

$L(\emptyset) = \emptyset$, $L(a) = \{a\}$, ...

abgeschlossen unter

Vereinigung \cup ja (triv)
 Konkatenation \cdot ja (triv)
 Stern-Operation $*$ ja (triv)
 Durchschnitt \cap ?
 Komplement $-$?

$L = L(\mathcal{A})$: Σ -NFA/DFA \mathcal{A}

\emptyset , $\{a\}$, ...

abgeschlossen unter

Vereinigung \cup ja
 Konkatenation \cdot ja
 Stern-Operation $*$ ja
 Durchschnitt \cap ja
 Komplement $-$ ja

Satz von Kleene: dies sind alternative Beschreibungen derselben Sprachklasse

DFA/NFA erkennbare Sprachen sind regulär

zum Beweis vom Satz von Kleene (Satz 2.3.1)

Aufgabe:

gewinne systematisch zu Σ -DFA/NFA \mathcal{A}
 regulären Ausdruck $\alpha \in \text{REG}(\Sigma)$ mit $L(\alpha) = L(\mathcal{A})$

Idee:

sukzessive Berechnung von α' für Hilfssprachen L' so,
 dass kompliziertere α'/L' sich einfach
 aus einfacheren zusammensetzen
 (algorithmisch vgl. Idee des dynamischen Programmierens)

o.B.d.A. betrachte DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$ mit $Q = \{1, \dots, n\}$

zum Beweis vom Satz von Kleene

DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$

$Q = \{1, \dots, n\}$

zu $0 \leq k \leq n$ und $1 \leq \ell, m \leq n$ sei

$L_{\ell, m}^k := \left\{ w \in \Sigma^* : \mathcal{A} \text{ hat Lauf von Zustand } \ell \text{ nach Zustand } m \right.$
 $\left. \text{auf } w \text{ über Zwischenzustände } q \in \{1, \dots, k\} \right\}$

$L_{\ell, m}^0 = \begin{cases} \{a \in \Sigma : \delta(\ell, a) = m\} & \text{falls } \ell \neq m \\ \{\varepsilon\} \cup \{a \in \Sigma : \delta(\ell, a) = \ell\} & \text{falls } \ell = m \end{cases}$ (endlich)

$L_{\ell, m}^{k+1} = \underbrace{L_{\ell, m}^k}_{(1)} \cup \underbrace{L_{\ell, k+1}^k}_{(2)} \cdot \underbrace{(L_{k+1, k+1}^k)^*}_{(3)} \cdot \underbrace{L_{k+1, m}^k}_{(4)}$

(1) Läufe ohne Zustand $k+1$;

(2) Läufe von Zustand ℓ zum ersten $k+1$;

(3) Schleifen durch Zustand $k+1$;

(4) Läufe vom letzten $k+1$ nach m .

Folgerungen aus dem Satz von Kleene

Korollar 2.3.2

die Klasse der regulären Sprachen ist abgeschlossen unter allen
 Booleschen Operationen sowie Konkatenation und Stern

alle Automaten-erkennbaren Sprachen lassen sich allein mit

- Vereinigung,
- Konkatenation und
- Stern

aus (einfachsten) endlichen Sprachen gewinnen

wieviele Zustände sind notwendig? → Abschnitt 2.4

Zustandszahlen von DFA \mathcal{A} mit $L = L(\mathcal{A})$
als Maß für Komplexität von L

Grundidee zu *minimalem* DFA für L :

jeder Zustand beschreibt notwendige Information
verschiedene Zustände : *notwendige Unterscheidungen*

Methode: betrachte induzierte **Äquivalenzrelationen** auf Σ^*

\sim_L zu gegebenem L $w \not\sim_L w'$: "notwendige Unterscheidung"

$\sim_{\mathcal{A}}$ zu gegebenem \mathcal{A} $w \not\sim_{\mathcal{A}} w'$: "verschiedene Berechnungen"

die Äquivalenzrelation \sim_L

$L \subseteq \Sigma^*$ DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$

\sim_L zu $L \subseteq \Sigma^*$:

$$w \sim_L w' \text{ gdw } (\forall x \in \Sigma^*) (wx \in L \Leftrightarrow w'x \in L)$$

- \sim_L ist Äquivalenzrelation auf Σ^* :
reflexiv, symmetrisch, transitiv
- \sim_L ist *rechts-invariant*: $w \sim_L w' \Rightarrow wu \sim_L w'u$
- L besteht aus ganzen \sim_L -Äquivalenzklassen

die Äquivalenzrelation $\sim_{\mathcal{A}}$

$L \subseteq \Sigma^*$ DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$

$\sim_{\mathcal{A}}$ zu DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$

$$w \sim_{\mathcal{A}} w' \text{ gdw } \hat{\delta}(q_0, w) = \hat{\delta}(q_0, w')$$

- $\sim_{\mathcal{A}}$ ist Äquivalenzrelation auf Σ^* :
reflexiv, symmetrisch, transitiv
- $\sim_{\mathcal{A}}$ ist *rechts-invariant*: $w \sim_{\mathcal{A}} w' \Rightarrow wu \sim_{\mathcal{A}} w'u$
- $\sim_{\mathcal{A}}$ hat *endlichen Index*: $\text{index}(\sim_{\mathcal{A}}) \leq |Q|$.

\sim_L und $\sim_{\mathcal{A}}$

Korollare 2.4.5/6

für $L = L(\mathcal{A})$: $\sim_{\mathcal{A}}$ Verfeinerung von \sim_L :

$$(\forall w, w' \in \Sigma^*) w \sim_{\mathcal{A}} w' \Rightarrow w \sim_L w'$$

$$\text{index}(\sim_L) \leq \text{index}(\sim_{\mathcal{A}}) \leq |Q|$$

Korollare aus dem Vergleich von \sim_L und $\sim_{\mathcal{A}}$

- L regulär $\Rightarrow \sim_L$ hat endlichen Index
- für reguläres L : jeder DFA, der L erkennt, hat mindestens $\text{index}(\sim_L)$ viele Zustände

Ziel: Satz von Myhill-Nerode

- \sim_L hat endlichen Index $\Rightarrow L$ regulär
- für reguläres L : ex. DFA mit $\text{index}(\sim_L)$ Zuständen für L

Myhill-Nerode

→ Abschnitt 2.4.1

der Äquivalenzklassen-Automat

Idee: assoziiere je einen Zustand mit jeder \sim_L -Äquivalenzklasse und erhalte minimalen DFA, der L erkennt

$[w] := \{v \in \Sigma^* : v \sim_L w\}$ die \sim_L -Äquivalenzklasse von w

$$\mathcal{A} = (\Sigma, Q, q_0, \delta, A) \quad Q := \Sigma^* / \sim_L$$

$$q_0 := [\varepsilon]$$

$$\delta([w], a) := [wa] \quad (\text{wohldefiniert!})$$

$$A := \{[w] : w \in L\}$$

$L = L(\mathcal{A})$ folgt aus: $(\forall w \in \Sigma^*) \hat{\delta}(q_0, w) = [w]$ (Induktion!)

Satz von Myhill-Nerode

Satz 2.4.7

für $L \subseteq \Sigma^*$ sind äquivalent:

- (i) \sim_L hat endlichen Index.
- (ii) L ist regulär.

Korollar aus dem Beweis:

kleinste DFA für reguläre L mit genau $\text{index}(\sim_L)$ vielen Zuständen

Folgerung aus dem Satz:

$L \subseteq \Sigma^*$ nicht-regulär \Leftrightarrow

es gibt eine Folge $(w_n)_{n \in \mathbb{N}}$ in Σ^* mit $w_n \not\sim_L w_m$ für $n \neq m$

Beispiel: $L = \{a^n b^n : n \in \mathbb{N}\} \subseteq \{a, b\}^*$ nicht regulär

das syntaktische Monoid

→ Abschnitt 2.4.2

anstelle von \sim_L betrachte die Verfeinerung \approx_L :

$$w \approx_L w' \quad \text{gdw} \quad (\forall x, y \in \Sigma^*) (xwy \in L \Leftrightarrow xw'y \in L)$$

- \approx_L ist Äquivalenzrelation auf Σ^*
- \approx_L ist Verfeinerung von \sim_L : $w \approx_L w' \Rightarrow w \sim_L w'$
- $\text{index}(\sim_L) \leq \text{index}(\approx_L)$
- $\text{index}(\approx_L) \leq n^n$ wenn L von DFA mit n Zuständen erkennbar
- \approx_L ist verträglich mit Konkatenation (*Kongruenzrelation*):
 $u \approx_L u'$ und $v \approx_L v' \Rightarrow uv \approx_L u'v'$

$$\left. \begin{array}{l} (\Sigma^* / \approx_L, \cdot, [\varepsilon]_{\approx_L}) \text{ heisst } \mathbf{\text{syntaktisches Monoid zu } L} \\ \Sigma^* \longrightarrow \Sigma^* / \approx_L \\ w \longmapsto [w]_{\approx_L} \end{array} \right\} \text{ ist Monoid-Homomorphismus}$$

Minimalautomat für reguläre Sprache → Abschnitt 2.4.3

$L \subseteq \Sigma^*$ regulär

\Rightarrow der Äquivalenzklassen-Automat zu \sim_L ist
 ein DFA mit minimaler Zustandszahl (= $\text{index}(\sim_L)$)
 unter allen DFA, die L akzeptieren

Satz**(Lemma 2.4.13)**

jeder DFA mit $\text{index}(\sim_L)$ Zuständen, der L erkennt,
 ist isomorph zum Äquivalenzklassen-Automat zu \sim_L

Äquivalenzklassen-Automat zu \sim_L : der Minimalautomat zu L

Minimierung eines gegebenen DFA

wie kann man redundante Zustände erkennen/eliminieren?

- zunächst eliminiere alle nicht erreichbaren Zustände
- dann: $q \not\sim q'$ wesentlich verschieden, wenn für ein $x \in \Sigma^*$ nicht $(\hat{\delta}(q, x) \in A) \leftrightarrow (\hat{\delta}(q', x) \in A)$
- Zusammenfassung von \sim -Klassen von Zuständen liefert minimierten DFA, isomorph zum Minimalautomat

algorithmisch: induktive Verfeinerung

Tests für x der Länge $i = 0, 1, \dots$

$q \not\sim_0 q'$ gdw. nicht $(q \in A) \leftrightarrow (q' \in A)$

$q \not\sim_{i+1} q'$ gdw. $q \not\sim_i q'$ oder $(\exists a \in \Sigma) \delta(q, a) \not\sim_i \delta(q', a)$

sobald $\sim_{i+1} = \sim_i$ ($=: \sim$), fasse Zustände in \sim -Klassen zusammen

Das "pumping lemma" → Abschnitt 2.5

Pumping Lemma

(Lemma 2.5.2)

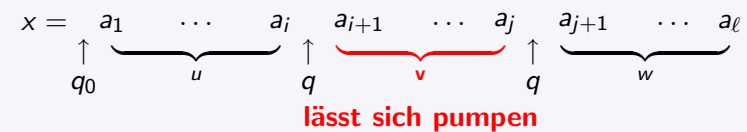
$L \subseteq \Sigma^*$ regulär \Rightarrow

existiert $n \in \mathbb{N}$ sodass sich jedes $x \in L$ mit $|x| \geq n$ zerlegen lässt in $x = uvw$, $v \neq \epsilon$, $|uv| \leq n$ und für alle $m \in \mathbb{N}$

$$u \cdot v^m \cdot w = u \cdot \underbrace{v \cdots v}_{m \text{ mal}} \cdot w \in L$$

Beweis: $L = L(A)$, DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$, $n := |Q|$

$|x| \geq n$: existiert Zustandswiederholung $q_i = q_j = q$ im Lauf auf x :



Beispiele nicht-regulärer Sprachen

$L = \{a^n b^n : n \in \mathbb{N}\}$ für $a, b \in \Sigma, a \neq b$

$L = \{w \in \{(,)\}^* : w \text{ korrekte Kammerschachtelung}\}$

$\text{PALINDROM} = \{w \in \Sigma^* : w = w^{-1}\}$ für $|\Sigma| \geq 2$

$L = \{u\#v\#w : u, v, w \in \{0, 1\}^*, u + v = w \text{ (binäre Addition)}\}$

Nachweis: Pumping Lemma!

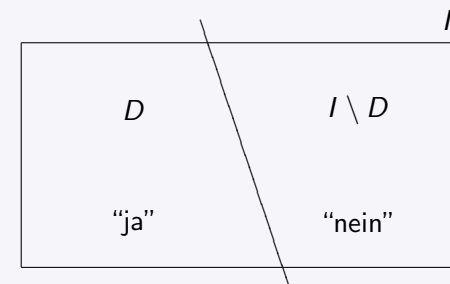
Algorithmische Entscheidungsprobleme → Abschnitt 2.6

Entscheidungsproblem (ja/nein Problem)

spezifiziert durch

- Menge von zugelassenen Eingaben $x \in I$ (Instanzen)
- Teilmenge $D \subseteq I$ der positiven Instanzen (Antwort "ja")

d.h.: wohldefinierte ja/nein Antwort für alle $x \in I$



Entscheidungsproblem:

gegeben $x \in I$
entscheide, ob $x \in D$

Entscheidungsprobleme für reguläre Sprachen Beispiele

Wortproblem zu Sprache $L \subseteq \Sigma^*$:

für $w \in \Sigma^*$ entscheide ob $w \in L$

$I = \Sigma^*$, $D = L$

Leerheitsproblem:

für DFA \mathcal{A} entscheide ob $L(\mathcal{A}) = \emptyset$

$I = \{\mathcal{A}: \mathcal{A} \text{ DFA}\}$, $D = \{\mathcal{A}: L(\mathcal{A}) = \emptyset\}$

analog mit $I = \text{REG}(\Sigma)$

Sprachgleichheit/Automatenäquivalenz:

für $\alpha, \beta \in \text{REG}(\Sigma)$ entscheide ob $L(\alpha) = L(\beta)$

$I = (\text{REG}(\Sigma))^2$, $D = \{(\alpha, \beta) \in (\text{REG}(\Sigma))^2: L(\alpha) = L(\beta)\}$

analog für DFA/NFA

das Wichtigste aus Kapitel 2

reguläre Sprachen

REG DFA NFA

Abschlusseigenschaften / Automatenkonstruktionen

Satz von Kleene

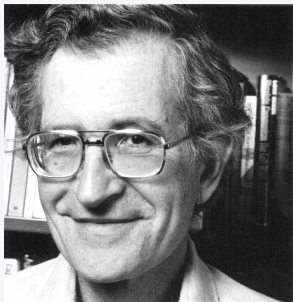
Satz von Myhill-Nerode

Pumping Lemma für reguläre Sprachen

Minimalautomat

Kapitel 3: Grammatiken

**Chomsky Hierarchie: Komplexitätsniveaus
für formale Sprachen**



Noam Chomsky (geb. 1928)

Grammatiken

→ [Abschnitt 3.1](#)

Idee: Spezifikation einer Sprache durch *Erzeugungsprozess*

$G = (\Sigma, V, P, X_0)$

Σ Terminalalphabet,

V endliche Menge von Variablen, $V \cap \Sigma = \emptyset$

$X_0 \in V$ Startvariable/Startsymbol

P endliche Menge von Produktionen/Regeln

$P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$

$(v, v') \in P$ eine Produktion/Regel von G

Regel

$v \rightarrow v'$

linke Seite: $v \in (V \cup \Sigma)^+ = (V \cup \Sigma)^* \setminus \{\varepsilon\}$

rechte Seite: $v' \in (V \cup \Sigma)^*$

erlaubt in $(V \cup \Sigma)$ -Wörtern $\left\{ \begin{array}{l} \text{Ersetzung von } v \text{ durch } v' \\ \text{Übergang von } uvw \text{ nach } uv'w \end{array} \right.$