Induktionsprinzipien für andere Bereiche

Beispiele

Bereich M	$M_0 \subseteq M$	erzeugende Operationen
$\mathbb{N}$	$\{0\}$	S: $n \longmapsto n+1$
Σ^*	$\{\varepsilon\}$	($w \longmapsto wa$) für $a \in \Sigma$
$\{*, c\}$-Terme	$\{c\}$	$(t_1, t_2) \longmapsto (t_1 * t_2)$
endl. Teilmengen von A	$\{\emptyset\}$	($B \longmapsto B \cup \{a\}$) für $a \in A$

falscher Induktionsbeweis über $\mathbb{N}$ Übung 1.2.7

$A(n)$: { jede Gruppe von n Personen besteht aus gleichaltrigen Personen.

Induktionsanfang: $A(n)$ wahr für $n = 0$ und $n = 1$.

Induktionsschritt: $A(n) \Rightarrow A(n+1)$.

Sei $n \geqslant 1$, $|P| = n+1$; $p_1 \neq p_2$ beliebig aus P ausgewählt.

Betrachte $P_1 := P \setminus \{p_1\}$ und $P_2 := P \setminus \{p_2\}$. $|P_1| = |P_2| = n$.

Nach Induktionsannahme $A(n)$ bestehen also P_1 und P_2 jeweils aus gleichaltrigen Personen.

Jedes $p \in P \setminus \{p_1, p_2\}$ ist in P_1 *und* in P_2 vorhanden.

Also sind alle in P gleichaltrig. Also gilt auch $A(n+1)$.

Also gilt $(\forall n \in \mathbb{N})A(n)$?

Kapitel 2: Endliche Automaten Reguläre Sprachen

Reguläre Σ-Sprachen → Abschnitt 2.1

Operationen auf Σ-Sprachen

Komplement	$L \longmapsto \overline{L} := \Sigma^* \setminus L$	Boolesche Operationen
Schnitt	$(L_1, L_2) \longmapsto L_1 \cap L_2$	
Vereinigung	$(L_1, L_2) \longmapsto L_1 \cup L_2$	

Konkatenation von Sprachen

$(L_1, L_2) \longmapsto L_1 \cdot L_2 := \{u \cdot v : u \in L_1, v \in L_2\}$

Stern-Operation

$L \longmapsto L^* := \{u_1 \cdot \ldots \cdot u_n : u_1, \ldots, u_n \in L, n \in \mathbb{N}\}$

Reguläre Ausdrücke — Definition 2.1.2

Die Menge $\mathrm{REG}(\Sigma)$ der *regulären Ausdrücke* über Σ, wird erzeugt gemäß:

(i) $\emptyset$ ist ein regulärer Ausdruck.
(ii) **a** ist ein regulärer Ausdruck, für $a \in \Sigma$.
(iii) für $\alpha, \beta \in \mathrm{REG}(\Sigma)$ ist $(\alpha + \beta) \in \mathrm{REG}(\Sigma)$.
(iv) für $\alpha, \beta \in \mathrm{REG}(\Sigma)$ ist $(\alpha\beta) \in \mathrm{REG}(\Sigma)$.
(v) für $\alpha \in \mathrm{REG}(\Sigma)$ ist $\alpha^* \in \mathrm{REG}(\Sigma)$.

[evtl. auch zugelassen: $\Sigma, \Sigma^*, \Sigma^+, \varepsilon$]

Reguläre Sprachen — Definition 2.1.3

Semantik für $\alpha \in \mathrm{REG}(\Sigma)$: $L(\alpha) \subseteq \Sigma^*$ die durch α bezeichnete reguläre Sprache

Induktiv/rekursiv über $\alpha \in \mathrm{REG}(\Sigma)$ definiere $L(\alpha)$:

(i) $L(\emptyset) := \emptyset$.
(ii) $L(\mathbf{a}) := \{a\}$.
(iii) $L(\alpha + \beta) := L(\alpha) \cup L(\beta)$.
(iv) $L(\alpha\beta) := L(\alpha) \cdot L(\beta)$.
(v) $L(\alpha^*) := \big(L(\alpha)\big)^*$.

Definition

Die *regulären* $\boldsymbol{\Sigma}$*-Sprachen* sind genau die $L(\alpha)$ für $\alpha \in \mathrm{REG}(\Sigma)$

Die **regulären Σ-Sprachen** werden erzeugt aus den **Ausgangssprachen** $\emptyset$ und $\{a\}$ für $a \in \Sigma$ durch die Operationen **Vereinigung**, **Konkatenation** und **Stern**.

Endliche Automaten — → Abschnitt 2.2

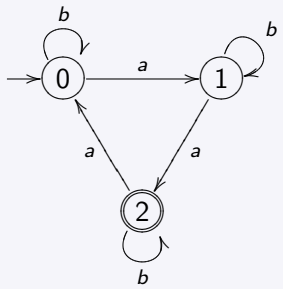
Transitionssysteme (mit endl. Zustandsmenge)

$\mathcal{S} = (\Sigma, Q, \Delta)$ mit den Komponenten:

Σ: Alphabet (Kantenbeschriftungen)
Q: Zustandsmenge, endlich, $\neq \emptyset$
$\Delta \subseteq Q \times \Sigma \times Q$: Transitionsrelation

$(q, a, q') \in \Delta$ steht für die Transition $q \xrightarrow{a} q'$

Beispiel: Transitionssystem mit Zusatzstruktur, DFA



modulo-3 Zähler für a, $|w|_a \bmod 3$.

Zusatzstruktur:
$\rightarrow$: Initialisierung;
(2): ausgezeichneter Zustand, hier für $|w|_a \equiv 2 \,(\mathrm{mod}\, 3)$

deterministisch:
Δ beschreibbar durch Funktion $\delta\colon Q \times \Sigma \to Q$

Endliche Automaten, DFA und NFA

Idee: Transitionssysteme zur *Erkennung von Sprachen*
deterministische Transitionssysteme/Automaten

anstelle der
Transitionsrelation $\Delta \subseteq Q \times \Sigma \times Q$

Transitionsfunktion $\delta\colon Q \times \Sigma \longrightarrow Q$
$(q, a) \longmapsto \delta(q, a) \in Q$

jeweils genau ein *eindeutig bestimmter Nachfolgezustand*
kein deadlock, keine Auswahl

nicht-deterministische Transitionssysteme/Automaten

Transitions*relation* bietet u.U. bei Eingabe a in Zustand q

$$\begin{cases} \text{kein } q' \text{ mit } (q, a, q') \in \Delta. & \text{deadlock} \\ \text{verschiedene } q' \text{ mit } (q, a, q') \in \Delta. & \text{Auswahl} \end{cases}$$

Deterministische endliche Automaten, DFA

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

Q	endliche, nicht-leere *Zustandsmenge*
$q_0 \in Q$	*Anfangszustand*
$A \subseteq Q$	Menge der *akzeptierenden Zustände*
$\delta\colon Q \times \Sigma \to Q$	*Übergangsfunktion.*

Berechnung von $\mathcal{A}$ auf $w = a_1 \dots a_n \in \Sigma^*$

die eindeutige Zustandsfolge $q_0, \dots, q_n$ mit
$q_{i+1} = \delta(q_i, a_{i+1})$ für $0 \leqslant i < n$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \quad \cdots \quad q_{n-1} \xrightarrow{a_n} q_n$$

DFA: Läufe und Berechnungen

Definition 2.2.4

analog zu Berechnung (vom Startzustand q_0 aus)
definiere Lauf auf w von $q \in Q$ aus

$$q \xrightarrow{a_1} q' \xrightarrow{a_2} \quad \cdots$$

führt zu eindeutiger Fortsetzung $\hat{\delta}$ von δ:

$$\hat{\delta}\colon Q \times \Sigma^* \longrightarrow Q, \quad (q, w) \longmapsto \hat{\delta}(q, w) \in Q$$

$\left\{ \begin{array}{l} \text{der (!) Endzustand des} \\ \text{Laufs auf } w \text{ von } q \text{ aus} \\ \textit{induktiv definiert} \end{array} \right.$

Berechnungen sind Läufe von q_0 aus;
Endzustand der Berechnung von $\mathcal{A}$ auf w: $\hat{\delta}(q_0, w)$
Läufe beschreiben auch Teilabschnitte von Berechnungen

erkannte/akzeptierte Sprache — Definition 2.2.3

DFA: von $\mathcal{A}$ erkannte/akzeptierte Sprache

$w = a_1 \dots a_n$ mit Berechnung $q_0, \dots, q_n$ $\quad q_n = \hat\delta(q_0, w)$

$$\mathcal{A} \quad \begin{cases} \textit{akzeptiert } w & \text{falls } q_n \in A \\ \textit{verwirft } w & \text{falls } q_n \notin A \end{cases}$$

die von $\mathcal{A}$ **akzeptierte/erkannte Sprache**:

$$L(\mathcal{A}) := \{ w \in \Sigma^* : \ \mathcal{A} \text{ akzeptiert } w \}$$
$$= \{ w \in \Sigma^* : \hat\delta(q_0, w) \in A \}$$

Nicht-deterministische endliche Automaten, NFA

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

Q	endliche, nicht-leere Zustandsmenge
$q_0 \in Q$	Anfangszustand
$A \subseteq Q$	Menge der akzeptierenden Zustände
$\Delta \subseteq Q \times \Sigma \times Q$	Übergangs*relation*.

Berechnung von $\mathcal{A}$ auf $w = a_1 \dots a_n \in \Sigma^*$

jede (!) Zustandsfolge $q_0, \dots, q_n$ mit $(q_i, a_{i+1}, q_{i+1}) \in \Delta$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \quad \cdots \quad q_{n-1} \xrightarrow{a_n} q_n$$

Vorsicht: i.d.R. nicht eindeutig, nicht notwendig existent!

erkannte/akzeptierte Sprache — Definition 2.2.6

NFA: von $\mathcal{A}$ erkannte/akzeptierte Sprache

eine Berechnung $q_0, \dots, q_n$ von $\mathcal{A}$ auf $w = a_1 \dots a_n$
ist eine *akzeptierende Berechnung* auf w falls $q_n \in A$

die von $\mathcal{A}$ **akzeptierte/erkannte Sprache**:

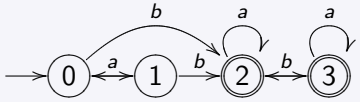
$L(\mathcal{A}) :=$

$\{ w \in \Sigma^* : \ \mathcal{A}$ *hat eine* akzeptierende Berechnung auf $w \}$

beachte: Existenz mindestens einer akzeptierenden Berechnung
Asymmetrie bzgl. akzeptieren/verwerfen

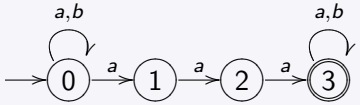
Beispiele — $\Sigma = \{a, b\}$

DFA $\mathcal{A}_1$



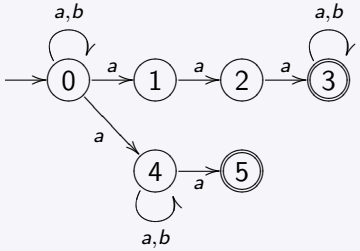
$L(\mathcal{A}_1) = ?$

NFA $\mathcal{A}_2$



$L(\mathcal{A}_2) = ?$

NFA $\mathcal{A}_3$



$L(\mathcal{A}_3) = ?$

Determinisierung → Abschnitt 2.2.3

von NFA zu äquivalentem DFA; Determinisierung

deterministische Simulation des NFA durch Potenzmengen-Trick

Satz 2.2.9

Zu NFA $\mathcal{A}$ lässt sich ein DFA $\mathcal{A}^{\text{det}}$ (effektiv) konstruieren, der dieselbe Sprache erkennt: $L(\mathcal{A}) = L(\mathcal{A}^{\text{det}})$.

Idee: Zustände von $\mathcal{A}^{\text{det}}$ geben an, in welchen Zuständen $\mathcal{A}$ *sein könnte*

Potenzmengen-Trick

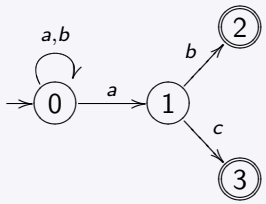
deterministische Simulation des NFA in DFA mittels Potenzmengen-Trick

	$\mathcal{A} = (\Sigma, Q, q_0, \Delta, A)$	$\mathcal{A}^{\text{det}} = (\Sigma, \hat{Q}, \hat{q}_0, \delta, \hat{A})$
Zustände	$q \in Q$	$S \subseteq Q$
Zust.-M.	Q	$\hat{Q} := \mathcal{P}(Q) = \{S \colon S \subseteq Q\}$
Start-Z.	q_0	$\hat{q}_0 := \{q_0\}$
akz.	A	$\hat{A} := \{S \colon S \cap A \neq \emptyset\}$
Trans.	$\Delta \subseteq Q \times \Sigma \times Q$	$\delta \colon \hat{Q} \times \Sigma \to \hat{Q}$

$$\delta(S, a) = \{q' \in Q \colon (q, a, q') \in \Delta \text{ für mindestens ein } q \in S\}$$

Beispiel: Determinisierung

NFA $\mathcal{A}$



$\Sigma = \{a, b, c\}$

DFA $\mathcal{A}^{\text{det}}$ mit $L(\mathcal{A}^{\text{det}}) = L(\mathcal{A})$:

δ	a	b	c
$\{0\}$	$\{0,1\}$	$\{0\}$	$\emptyset$
$\{0,1\}$	$\{0,1\}$	$\{0,2\}$	$\{3\}$
$\{0,2\}$	$\{0,1\}$	$\{0\}$	$\emptyset$
$\{3\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

(aktive) Zustände: $\{0\}, \{0,1\}, \{0,2\}, \{3\}, \emptyset$

akzeptierende Zustände: $\{0,2\}$ und $\{3\}$

$$L(\mathcal{A}) = L((a+b)^* a(b+c))$$

Abschlusseigenschaften → Abschnitt 2.2.4

Abschlusseigenschaften für NFA/DFA erkennbare Sprachen

Nachweis: Automatenkonstruktionen Lemmata 2.2.11/14

Vereinigung
zu DFA $\mathcal{A}_1, \mathcal{A}_2$ existiert DFA $\mathcal{A}$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Durchschnitt
zu DFA $\mathcal{A}_1, \mathcal{A}_2$ existiert DFA $\mathcal{A}$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Komplement
zu DFA $\mathcal{A}_1$ existiert DFA $\mathcal{A}$ mit $L(\mathcal{A}) = \overline{L(\mathcal{A}_1)}$.

Konkatenation
zu NFA $\mathcal{A}_1, \mathcal{A}_2$ existiert NFA $\mathcal{A}$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cdot L(\mathcal{A}_2)$.

Stern-Operation
zu NFA $\mathcal{A}_1$ existiert NFA $\mathcal{A}$ mit $L(\mathcal{A}) = (L(\mathcal{A}_1))^*$.

Abschlusseigenschaften

Durchschnitt und Vereinigung (für DFA)

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)

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

$$Q := Q^{(1)} \cup Q^{(2)}$$
$$q_0 := q_0^{(1)}$$
$$A := A^{(2)}$$
$$\Delta := \Delta^{(1)} \cup \Delta^{(2)} \cup \Delta^{(1)\to(2)}$$

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

$(*)$: was ist andernfalls zu tun?

Abschlusseigenschaften Korollar 2.2.16

alle regulären Sprachen von NFA/DFA erkannt

per Induktion über reguläre Ausdrücke zeige:

$(\forall \alpha \in \mathrm{REG}(\Sigma))$ $L(\alpha)$ 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**
$L = L(\alpha)$: $\alpha \in \mathrm{REG}(\Sigma)$	$L = L(\mathcal{A})$: Σ-NFA/DFA $\mathcal{A}$
$L(\emptyset) = \emptyset$, $L(a) = \{a\}$, ...	$\emptyset$, $\{a\}$, ...
abgeschlossen unter	abgeschlossen unter
Vereinigung $\cup$ ja (triv)	Vereinigung $\cup$ ja
Konkatenation $\cdot$ ja (triv)	Konkatenation $\cdot$ ja
Stern-Operation * ja (triv)	Stern-Operation * ja
Durchschnitt $\cap$ **?**	Durchschnitt $\cap$ ja
Komplement $^-$ **?**	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 $\alpha \in \mathrm{REG}(\Sigma)$ 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 $Q = \{1, \ldots, n\}$

zum Beweis vom Satz von Kleene

DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, a)$ $\qquad Q = \{1, \ldots, n\}$

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

$L^k_{\ell,m} := \Big\{ w \in \Sigma^* : \begin{array}{l} \mathcal{A} \text{ hat Lauf von Zustand } \ell \text{ nach Zustand } m \\ \text{auf } w \text{ über Zwischenzustände } q \in \{1, \ldots, k\} \end{array} \Big\}$

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

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

(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