Eigenschaften von zweistelligen Operationen

für 2-stellige Operation $*\colon A \times A \longrightarrow A$
$a, b \longmapsto a * b$ (infixe Notation)

assoziativ, falls für alle $a, b, c \in A$: $(a * b) * c = a * (b * c)$.

kommutativ, falls für alle $a, b \in A$: $a * b = b * a$.

neutrales Element: $e \in A$ neutrales Element für $*$
gdw für alle $a \in A$: $a * e = e * a = a$.

inverse Elemente (bzgl. $*$ mit neutralem Element e):
$a' \in A$ inverses Element zu $a \in A$, falls
$a * a' = a' * a = e$.

Beispiel Konkatenation: assoziativ, neutrales Element ε,
$w \neq \varepsilon$ hat kein inverses Element

(algebraische) Strukturen

→ Abschnitt 1.1.4

Struktur =

Trägermenge mit ausgezeichneten $\begin{cases} \textbf{Konstanten}, \\ \textbf{Operationen}, \\ \textbf{Relationen} \end{cases}$

typische Beispiele:

- **Standardstrukturen der Algebra**
  $(\mathbb{N}, +, 0)$, $(\mathbb{N}, +, \cdot, <, 0, 1)$, $(\mathbb{Z}, +, \cdot, 0, 1)$, ...
- **Graphen (Transitionssysteme)**
- **Wortmonoide**
- **Boolesche Algebren**
- später: Wortstrukturen, relationale Datenbanken, u.v.a.m.

Strukturtypen: Beispiele

Graphen (Transitionssysteme) als relationale Strukturen

(V, E) mit Knotenmenge V, Kantenrelation E
$E \subseteq V \times V$ eine 2-stellige Relation
$(a, b) \in E$ zu deuten als $a \xrightarrow{E} b$

Monoide als algebraische Strukturen

Monoid: assoziative 2-stellige Operation mit neutralem Element

Beispiel Wort-Monoid

das Wort-Monoid $(\boldsymbol{\Sigma^*}, \cdot, \varepsilon)$ über $\boldsymbol{\Sigma}$
$\cdot$, Konkatenation, als 2-stellige Operation
ε, das leere Wort, als Konstante

Beispiel: Boolesche Algebren

Axiome für Boolesche Algebra $(\mathbf{B}, \cdot, +, {}', 0, 1)$:

BA1: $+$ und $\cdot$ assoziativ und kommutativ.

Für alle x, y, z: $(x + y) + z = x + (y + z)$ $\quad x + y = y + x$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$ $\quad x \cdot y = y \cdot x$

BA2: $+$ und $\cdot$ distributiv.

Für alle x, y, z: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
$x + (y \cdot z) = (x + y) \cdot (x + z)$

BA3: 0 und 1 als neutrale Elemente.

Für alle x: $x \cdot 1 = x$ $\quad x + 0 = x$

BA4: Komplement.

$0 \neq 1$ und für alle x: $x \cdot x' = 0$ $\quad x + x' = 1$

Beispiele: $(\boldsymbol{\mathcal{P}(M)}, \cap, \cup, {}^-, \emptyset, \mathbf{M})$ für $M \neq \emptyset$; $(\mathbb{B}, \wedge, \vee, \neg, \mathbf{0}, \mathbf{1})$

Homomorphismen $\rightarrow$ Abschnitt 1.1.5

strukturerhaltende Abbildungen zw. Strukturen desselben Typs

z.B. für Strukturen $(A, *^A, e^A)$ und $(B, *^B, e^B)$
mit einer zweistelligen Operation $*$ und einer Konstanten e

$$\left.\begin{array}{rcl} F\colon A & \longrightarrow & B \\ a & \longmapsto & f(a) \end{array}\right\} \quad \textbf{Homomorphismus}\text{, falls}$$

(i) $F(e^A) = e^B$ (verträglich mit Konstante e)

(ii) $F(a_1 *^A a_2) = F(a_1) *^B F(a_2)$ (verträglich mit Operation $*$)

$$\begin{tikzcd} e^A \arrow[d, "F"] & A \times A \arrow[r, "*^A"] \arrow[d, shift right, "F"'] \arrow[d, shift left, "F"] & A \arrow[d, "F"] \\ e^B & B \times B \arrow[r, "*^B"] & B \end{tikzcd}$$

Homomorphismen: Beispiele

(1) $\begin{array}{rcl} h\colon \Sigma^* & \longrightarrow & \mathbb{N} \\ w & \longmapsto & |w| \end{array}$

Homomorphismus von $(\Sigma^*, \cdot, \varepsilon)$ nach $(\mathbb{N}, +, 0)$.

(2) $\begin{array}{rcl} \hat{f}\colon \Sigma_1^* & \longrightarrow & \Sigma_2^* \\ w = a_1 \dots a_n & \longmapsto & a_1' \dots a_n' \end{array}$

wobei $a_i' = f(a_i)$ für eine vorgeg. Funktion $f\colon \Sigma_1 \to \Sigma_2$

Homomorphismus von $(\Sigma_1^*, \cdot, \varepsilon)$ nach $(\Sigma_2^*, \cdot, \varepsilon)$.

(3) analog zu (2), zu $f\colon \Sigma_1 \to \Sigma_2^*$:
ersetze $a \in \Sigma_1$ durch ein Wort $f(a) \in \Sigma_2^*$.

Bemerkung: $\hat{f}$ in (2) und (3) eindeutig bestimmt durch f und die Forderung, dass $\hat{f}\colon (\Sigma_1^*, \cdot, \varepsilon) \xrightarrow{\text{hom}} (\Sigma_2^*, \cdot, \varepsilon)$
und dass $\hat{f}$ Fortsetzung von f ist: $\hat{f}(a) := f(a)$ f.a. $a \in \Sigma_1$.

Isomorphie – Isomorphismen

Isomorphismus: bijektiver Homomorphismus, dessen
Umkehrung auch ein Homomorphismus ist.

Beispiel

Für eine Bijektion $\begin{array}{rcl} f\colon \Sigma_1 & \longrightarrow & \Sigma_2 \\ a & \longmapsto & f(a) =: a' \end{array}$

ist $\begin{array}{rcl} \hat{f}\colon \Sigma_1^* & \longrightarrow & \Sigma_2^* \\ w = a_1 \dots a_n & \longmapsto & a_1' \dots a_n' \end{array}$

ein Isomorphismus zwischen $(\Sigma_1^*, \cdot, \varepsilon)$ und $(\Sigma_2^*, \cdot, \varepsilon)$.

Schreibweise: $\hat{f}\colon (\Sigma_1^*, \cdot, \varepsilon) \simeq (\Sigma_2^*, \cdot, \varepsilon)$

Beobachtung: $(\Sigma_1^*, \cdot, \varepsilon) \simeq (\Sigma_2^*, \cdot, \varepsilon)$ gdw. $|\Sigma_1| = |\Sigma_2|$

elementare Beweistechniken $\rightarrow$ Abschnitt 1.2

teilweise Vorgriff auf Teil II (Logik)

primäres Anliegen hier:

- normierte Verknüpfung von Aussagen, Aussagenlogik (AL)
- mathematische Präzision für Quantoren, Quantorenlogik
- Beweistechniken/-muster, insbesondere: Induktionsbeweise

Präzision des Ausdrucks / Strenge des Argumentierens mathematische Grunddisziplin für den Werkzeugkasten

aussagenlogische Junktoren → Abschnitt 1.2.1

normierte Wahrheitswerte für aussagenlogische Operationen

Wahrheitswerte (wahr bzw. falsch; 1 bzw. 0) zusammengesetzter Aussagen als Funktion der Wahrheitswerte der Teilaussagen

und $\wedge \colon \mathbb{B} \times \mathbb{B} \to \mathbb{B}$

$\wedge$	0	1
0	0	0
1	0	1

oder $\vee \colon \mathbb{B} \times \mathbb{B} \to \mathbb{B}$

$\vee$	0	1
0	0	1
1	1	1

Negation $\neg \colon \mathbb{B} \to \mathbb{B}$

	0	1
$\neg$	1	0

vgl. Boolesche Algebra $(\mathbb{B}, \wedge, \vee, \neg, 0, 1)$

weitere aussagenlogische Verknüpfungen

abgeleitete Junktoren , z.B.

Implikation $\to \colon \mathbb{B} \times \mathbb{B} \to \mathbb{B}$

$\to$	0	1
0	**1**	1
1	0	1

Biimplikation $\leftrightarrow \colon \mathbb{B} \times \mathbb{B} \to \mathbb{B}$

$\leftrightarrow$	0	1
0	1	0
1	0	1

sodass

$$\begin{aligned} (p \to q) &\equiv (\neg p) \vee q \\ (p \leftrightarrow q) &\equiv (p \wedge q) \vee ((\neg p) \wedge (\neg q)) \\ &\equiv (p \to q) \wedge (q \to p) \end{aligned}$$

aussagenlogische Äquivalenzen und Schlussregeln

Kontraposition: $(p \to q) \equiv ((\neg q) \to (\neg p))$

- beweise "$A \Rightarrow B$" über "$\neg B \Rightarrow \neg A$"

Indirekter Beweis/Widerspruchsbeweis: $p \equiv (\neg p \to 0)$

- beweise "A" über "$(\neg A)$ unmöglich"

Biimplikation/Äquivalenz: $(p \leftrightarrow q) \equiv ((p \to q) \wedge (q \to p))$

- beweise "$A \Leftrightarrow B$" über "$A \Rightarrow B$ und $B \Rightarrow A$"

Implikationsketten:

- beweise "$A \Rightarrow B$" z.B. über "$A \Rightarrow C$ und $C \Rightarrow B$" (Zwischenbehauptungen)

Quantoren: All- und Existenzaussagen → Abschnitt 1.2.2

$(\forall n \in \mathbb{N})A(n)$ für die **Allaussage** "für alle $n \in \mathbb{N}$ gilt $A(n)$"

$(\exists n \in \mathbb{N})A(n)$ für die **Existenzaussage** "$A(n)$ gilt für mindestens ein $n \in \mathbb{N}$"

Negationen von Allaussagen sind äquivalent zu Existenzaussagen und umgekehrt.

Beispiel

$\neg$("alle Schnurze beissen") $\equiv$ "es gibt mindestens einen Schnurz, der nicht beisst"

beachte: "alle Schnurze beissen" ist wahr,wenn es keine Schnurze gibt!

wichtig:

Allaussagen kann man durch ein Gegenbeispiel widerlegen, aber nicht durch Beispiele beweisen!

Induktionsbeweise

→ Abschnitt 1.2.3

Prinzip der vollständigen Induktion über $\mathbb{N}$:

beweise die Allaussage $(\forall n \in \mathbb{N}) A(n)$ anhand von

(i) **Induktionsanfang**: $A(0)$.

(ii) **Induktionsschritt**: für alle $n \in \mathbb{N}$ gilt: $A(n) \Rightarrow A(n+1)$

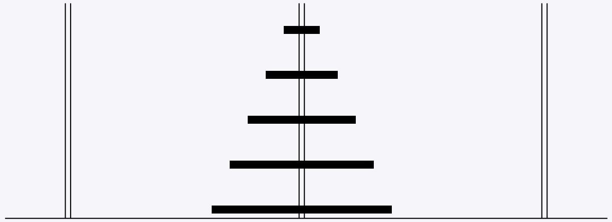
Rechtfertigung:

für jedes feste n ergibt sich aus (ii) eine Implikationskette

$A(0) \Rightarrow A(1) \Rightarrow A(2) \Rightarrow \ldots \Rightarrow A(n-1) \Rightarrow A(n)$

Beispiel: Induktionsbeweis über $\mathbb{N}$

Beispiel 1.2.2



$A(n)$: n Scheiben lassen sich in $2^n - 1$ Schritten gemäß der Regeln umschichten, und nicht in weniger Schritten

Induktionsanfang: $A(0)$

Induktionsschritt: für alle $n \in \mathbb{N}$ gilt: $A(n) \Rightarrow A(n+1)$

Induktionsprinzipien für andere Bereiche

Beispiel 1.2.4

betrachte Menge M der Terme mit 2-st. Fktn $*$ und Konst. c
als Menge von Wörtern über $\Sigma = \{*, c, (,)\}$, $M \subseteq \Sigma^*$

$$M = \{c, c*c, c*(c*c), \ldots, (c*c)*(c*(c*c)), \ldots\}$$

systematische Erzeugung aller $t \in M$:

ausgehend vom Startelement $c \in M$
mit Operation $F \colon M \times M \longrightarrow M$

$$F(t_1, t_2) := \begin{cases} (t_1) * (t_2) & \text{für } t_1, t_2 \neq c \\ c * (t_2) & \text{für } t_1 = c, t_2 \neq c \\ (t_1) * c & \text{für } t_1 \neq c, t_2 = c \\ c * c & \text{für } t_1 = t_2 = c \end{cases}$$

Beweise damit z.B.: $\boxed{(\forall t \in M)\big(|t|_c = |t|_* + 1\big)}$

Induktionsprinzipien für andere Bereiche

M werde, ausgehend von $M_0 \subseteq M$,
durch Operationen $F \in \mathcal{F}$ erzeugt; dann lässt sich

$$(\forall x \in M)\, A(x)$$

beweisen anhand von

(i) **Induktionsanfang**: $A(x)$ gilt für alle $x \in M_0$.

(ii) **Induktionsschritt(e)** für $F \in \mathcal{F}$ (n-stellig):
aus $A(x_i)$ für $i = 1, \ldots, n$ folgt, dass auch $A(F(x_1, \ldots, x_n))$.

Induktionsprinzipien für andere Bereiche

Beispiele

Bereich M	$M_0 \subseteq M$	erzeugende Operationen
$\mathbb{N}$	$\{0\}$	$\mathrm{S}\colon n \longmapsto n+1$
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

$$A(n)\colon \begin{cases}\text{jede Gruppe von } n \text{ Personen besteht aus}\\ \text{gleichaltrigen Personen.}\end{cases}$$

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

$$\left.\begin{array}{ll}\textbf{Komplement} & L \longmapsto \overline{L} := \Sigma^* \setminus L\\ \textbf{Schnitt} & (L_1, L_2) \longmapsto L_1 \cap L_2\\ \textbf{Vereinigung} & (L_1, L_2) \longmapsto L_1 \cup L_2\end{array}\right\}\ \text{Boolesche Operationen}$$

Konkatenation von Sprachen

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

Stern-Operation

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

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: $\Sigma, \Sigma^*, \Sigma^+, \varepsilon$]

Beispiel: $(\mathtt{b}^* \,\mathtt{a}\, \mathtt{b}^* \,\mathtt{a}\, \mathtt{b}^* \,\mathtt{a})^* \,\mathtt{b}^*$

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 $\alpha \in \mathrm{REG}(\Sigma)$

Beispiel: $L(\mathtt{b}^* \,\mathtt{a}\, \mathtt{b}^* \,\mathtt{a}\, \mathtt{b}^* \,\mathtt{a})^* \,\mathtt{b}^* =$
$= \big\{$alle Wörter mit einer durch 3 teilbaren Anzahl 'a's$\big\}$

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

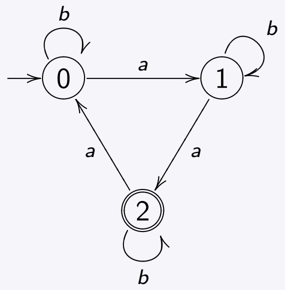
Transitionssysteme (mit endl. Zustandsmenge)

$\mathcal{S} = \big(\Sigma, Q, \Delta\big)$ 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 $\delta\colon Q \times \Sigma \rightarrow Q$

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 $\begin{array}{rcl} \delta\colon Q \times \Sigma & \longrightarrow & Q \\ (q, a) & \longmapsto & \delta(q, a) \in Q \end{array}$

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*
$\delta\colon Q \times \Sigma \rightarrow Q$	*Übergangsfunktion.*

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

die eindeutige Zustandsfolge $q_0, \ldots, q_n$ 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:

$$\begin{array}{rcl} \hat{\delta}\colon Q \times \Sigma^* & \longrightarrow & Q \\ (q, w) & \longmapsto & \hat{\delta}(q, w) \in Q \end{array} \quad \begin{cases} \text{der (!) Endzustand des} \\ \text{Laufs auf } w \text{ von } q \text{ aus} \\ \textit{induktiv definiert} \end{cases}$$

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 $q_0, \dots, q_n$ $\qquad q_n = \hat{\delta}(q_0, w)$

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

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 $q_n \in A$

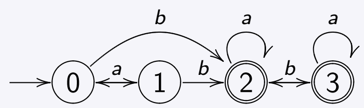
die von $\mathcal{A}$ **akzeptierte/erkannte Sprache**:

$$L(\mathcal{A}) := \{w \in \Sigma^* : \mathcal{A} \textit{ hat eine } \text{akzeptierende Berechnung auf } w\}$$

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

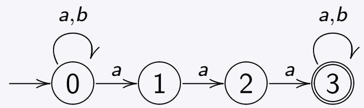
Beispiele — $\Sigma = \{a, b\}$

DFA $\mathcal{A}_1$



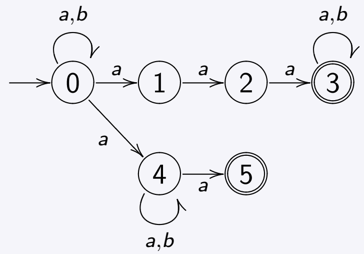
$L(\mathcal{A}_1) = ?$

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 $\mathcal{A}^{\mathrm{det}}$ (effektiv) konstruieren, der dieselbe Sprache erkennt: $L(\mathcal{A}) = L(\mathcal{A}^{\mathrm{det}})$.

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

Potenzmengen-Trick

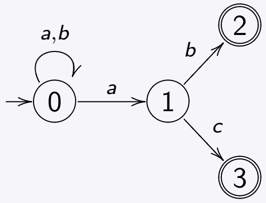
deterministische Simulation des NFA in DFA mittels Potenzmengen-Trick

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

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

Beispiel: Determinisierung

NFA $\mathcal{A}$



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

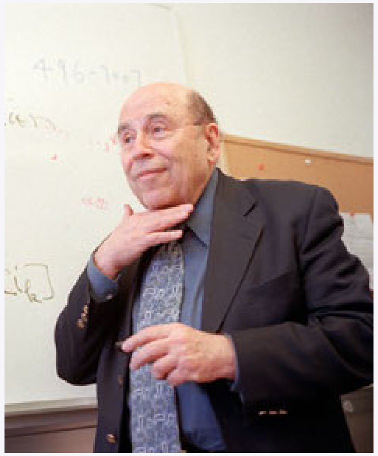
DFA $\mathcal{A}^{\mathrm{det}}$ mit $L(\mathcal{A}^{\mathrm{det}}) = L(\mathcal{A})$:

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

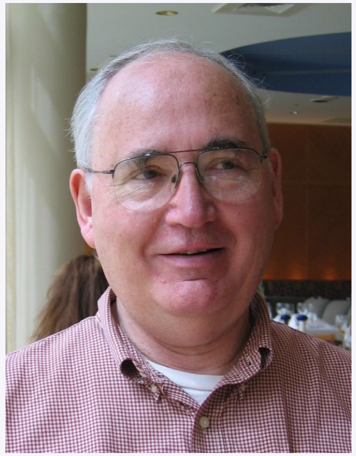
(aktive) Zustände: $\{0\}, \{0,1\}, \{0,2\}, \{3\}, \emptyset$

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

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



Michael O. Rabin



Dana Scott

"*Finite Automata and Their Decision Problem*" (1959)