3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

26.3.2019

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
   - 3.1. Was ist Logik?
   - 3.2. Aussagenlogische Funktionen
   - 3.3. Syntax und Semantik der Aussagenlogik
   - 3.4. Von der Funktion zur Formel
   - 3.5. Normalformen
   - 3.6. Das Erfüllbarkeitsproblem
   - 3.7. House
   - 3.8. Dualität von Funktionen, Operatoren und Formeln
   - 3.9. Gone Maggie gone

The Simpsons – aussagenlogische Modellierung

If all you have is a hammer, everything looks like a nail.

M_i ... Maggie
K_i ... Knecht Ruprecht
G_i ... Gift
H_i ... Homer

} ... befindet sich zum Zeitpunkt i auf der anderen Seite des Flusses.

- Zum Zeitpunkt i befinden sich alle auf dieser Flussseite.

$$\text{AlleHier}(i) := \neg M_i \wedge \neg K_i \wedge \neg G_i \wedge \neg H_i$$

- Zum Zeitpunkt i befinden sind alle auf der anderen Flussseite.

$$\text{AlleDort}(i) := M_i \wedge K_i \wedge G_i \wedge H_i$$

- Wenn sich Maggie und KR oder Maggie und das Gift am selben Flussufer befinden, muss Homer bei Maggie sein.

$$\text{Sicher}(i) := ((M_i \equiv K_i) \vee (M_i \equiv G_i)) \supset (M_i \equiv H_i)$$

MH_i ... Maggie
KH_i ... Knecht Ruprecht
GH_i ... Gift
} ... fährt mit Homer über den Fluss (zw. den Zeitpunkten $i-1$ und i).

HH_i ... Homer fährt alleine über den Fluss (zwischen $i-1$ und i).

- Genau eine Überfahrt zwischen den Zeitpunkten $i-1$ und i.

Überfahrt(i) :=

$$(MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee$$
$$(\neg MH_i \wedge \neg KH_i \wedge GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge HH_i)$$

- Definition der Überfahrten:

DefÜberfahrt(i) :=

$$\begin{aligned}
&(MH_i \supset ((M_{i-1} \not\equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv M_i))) \\
\wedge\ &(KH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \not\equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv K_i))) \\
\wedge\ &(GH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \not\equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv G_i))) \\
\wedge\ &(HH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i))
\end{aligned}$$

Gesamtformel: Nach n Überfahrten sollen alle auf der anderen Seite sein.

$$\text{Simpsons}(n) := \text{AlleHier}(0) \wedge \text{AlleDort}(n) \wedge \bigwedge_{i=0}^{n} \text{Sicher}(i)$$
$$\wedge \bigwedge_{i=1}^{n} \text{Überfahrt}(i) \wedge \bigwedge_{i=1}^{n} \text{DefÜberfahrt}(i)$$

Methode zum Lösen des Rätsels:

1. Errate die benötigte Zahl n der Überfahrten.
2. Finde eine erfüllende Interpretation für die Formel Simpsons(n) (z.B. mit Hilfe eines SAT-Solvers).

Eine mögliche Lösung:

$n = 7$,

$I(MH_1) = I(HH_2) = I(GH_3) = I(MH_4) = I(KH_5) = I(HH_6) = I(MH_7) = 1$

Vor- und Nachteile dieser aussagenlogischen Modellierung

Vorteile:

- deklarativ-statisch, nicht prozedural-dynamisch
  Welche Eigenschaften sollen gelten?
  Nicht: Welche Schritte sind für Lösung erforderlich?
- modular
  Neue Bedingungen werden durch zusätzliche Formeln berücksichtigt.

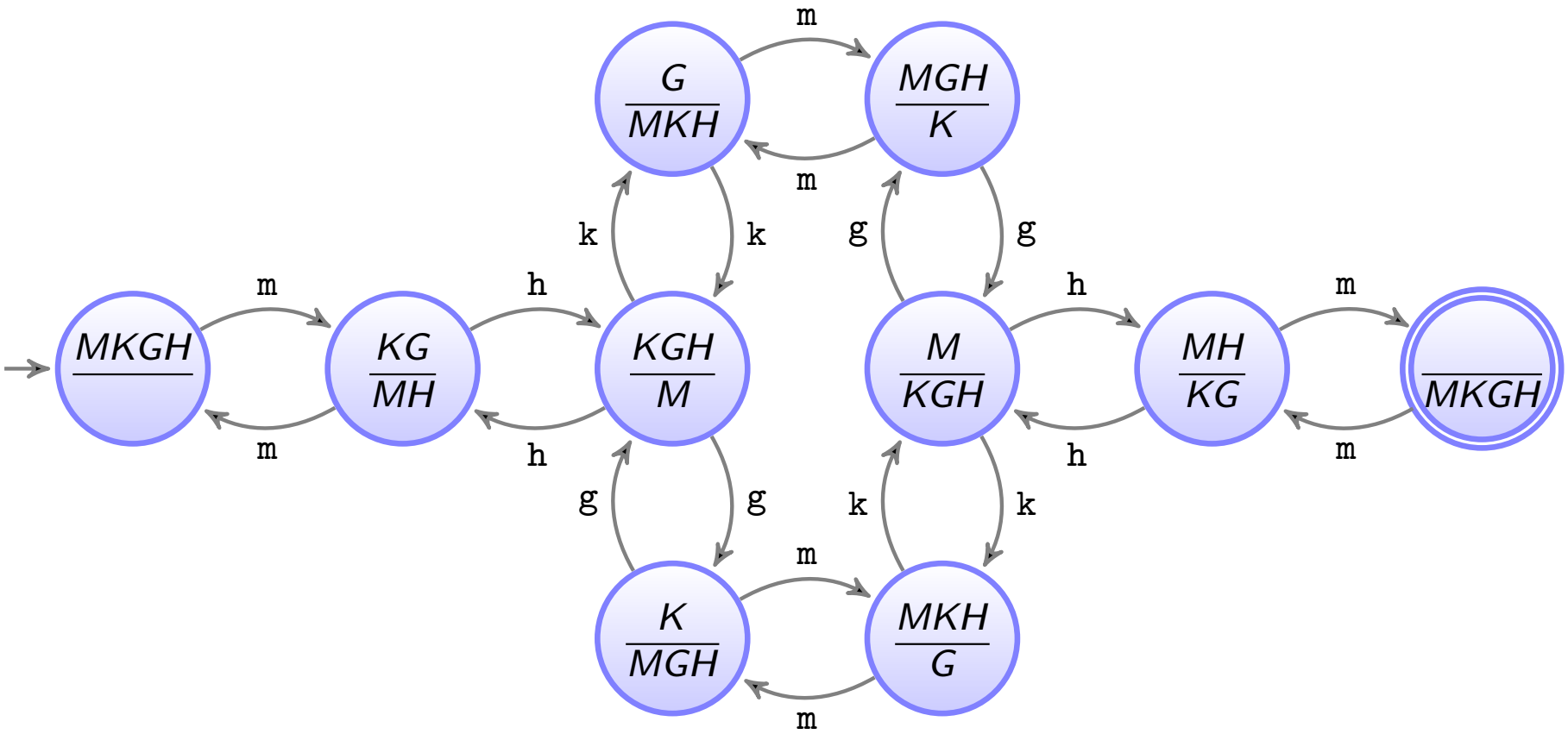
Nachteile:

- Erraten von Parametern
  n muss durch Probieren gefunden werden
- Große Zahl an Variablen und Formeln
  Dynamik muss durch indizierte Variablen simuliert werden.
- Frame Problem
  Bei jeder Aktion muss auch definiert werden, was sich nicht ändert.
- unintuitiv
  Bei der Modellierung von Abläufen denkt man an Zustände und Übergänge, nicht an statische Bedingungen.

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten

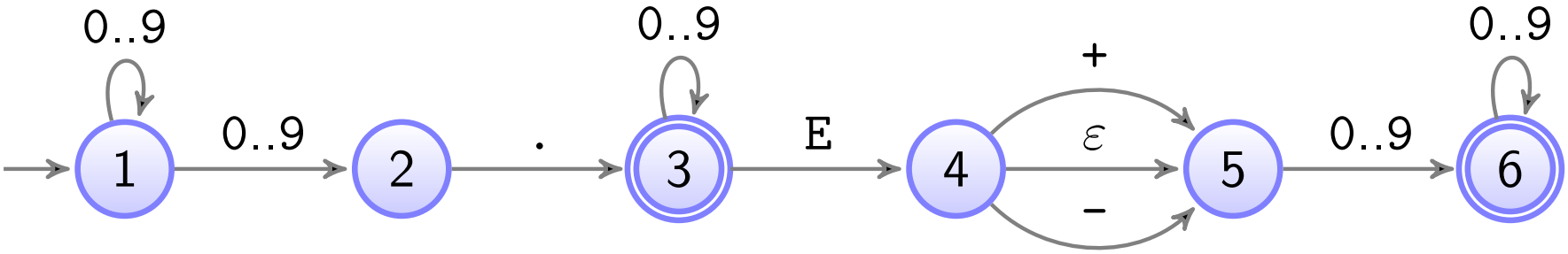
Beispiel: Flussüberquerung als Automat



Mögliche Lösungen: $\{\texttt{mhkmghm}, \texttt{mmmhhhgmkhm}, \ldots, \texttt{mhkmgkmgkmghm}, \ldots\}$

„Sprache des Automaten“

Beispiel: Reelle Numerale



- Zustandsbeschriftungen 1–6 dienen nur der Bezugnahme, irrelevant für das Verhalten des Automaten
- Kanten sind mit Symbolen beschriftet, die gelesen/geschrieben werden.
- Anfangszustand (1) ist durch einen Pfeil aus dem Nichts markiert.
- Endzustände (3, 6) sind durch einen Doppelkreis markiert.

Zwei Sichtweisen:

- Akzeptor: Der Automat liest Symbole und akzeptiert alle Zeichenketten, die vom Anfangs- zu einem der Endzustände führen.
- Generator: Der Automat schreibt Symbole und generiert jene Zeichenketten, die vom Anfangs- zu einem der Endzustände führen.

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten

Endliche Automaten modellieren Systeme bzw. Abläufe, die nur eine begrenzte, feste Zahl an unterscheidbaren Zuständen besitzen.

(Klassischer) Endlicher Automat:

- Anfangs- und Endzustände
- nur Eingaben (bzw. nur Ausgaben)
- verarbeitet endliche Symbolfolgen
- Unterarten: deterministisch, nichtdeterministisch mit/ohne ε-Übergängen

Transducer: wie endlicher Automat, aber mit Ein- und Ausgaben.
Unterarten: Mealy-Automaten, Moore-Automaten

Büchi-Automat: wie endlicher Automat, verarbeitet aber unendliche Symbolfolgen

Spezifikation von Automaten:

- Graphisch: Zustände sind Knoten, Übergänge sind Kanten, Ein- und Ausgaben sind Beschriftungen von Knoten und Kanten.
- Tabellarisch: Zu jedem Zustand und Eingabesymbol gibt es einen Eintrag mit zugehöriger Ausgabe und den Folgezuständen.

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten

Formale Sprachen

Σ Alphabet, d.h., endliche, nicht-leere Menge atomarer Symbole

w Wort über Σ, (endliche) Folge von Zeichen aus dem Alphabet Σ

ε Leerwort

Σ^+ Menge aller nicht-leeren endlichen Wörter über Σ

Σ^* Menge aller endlichen Wörter über Σ (inklusive Leerwort)

$w_1 \cdot w_2 = w_1 w_2$ Verkettung der Wörter $w_1, w_2 \in \Sigma^*$

$L \subseteq \Sigma^*$ formale Sprache über Σ

2^{Σ^*} Menge aller Sprachen über Σ

$\langle \Sigma^*, \cdot, \varepsilon \rangle$ bildet ein Monoid.

$\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$

$10 \cdot \varepsilon \cdot 11101 \cdot 000 = 1011101000$ (Klammerung irrelevant, Assoziativität!)

$\varepsilon \cdot \varepsilon \cdot \varepsilon = \varepsilon$

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten

Endliche Automaten

Bezeichnungen

Q	endliche Menge von Zuständen
q_0	Anfangszustand
F	Menge von Endzuständen
Σ	Eingabealphabet
δ	Übergangsfunktion/-relation
δ^*	erweiterte Übergangsfunktion/-relation
$\mathcal{A}$	Automat
$\mathcal{L}(\mathcal{A})$	die von $\mathcal{A}$ akzeptierte Sprache

Übergangsfunktionen und -relationen

$\delta\colon Q \times \Sigma \mapsto Q$	det. endlicher Automat (DEA)
$\delta \subseteq Q \times \Sigma \times Q$ $\delta\colon Q \times \Sigma \mapsto 2^Q$	nichtdet. endl. Aut. (NEA) ohne ε-Ü.
$\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$	nichtdet. endl. Aut. (NEA) mit ε-Ü.

ε-Übergänge, Relation oder Potenzmenge $\Longrightarrow$ nichtdeterm. Automat

Akzeptierte Wörter

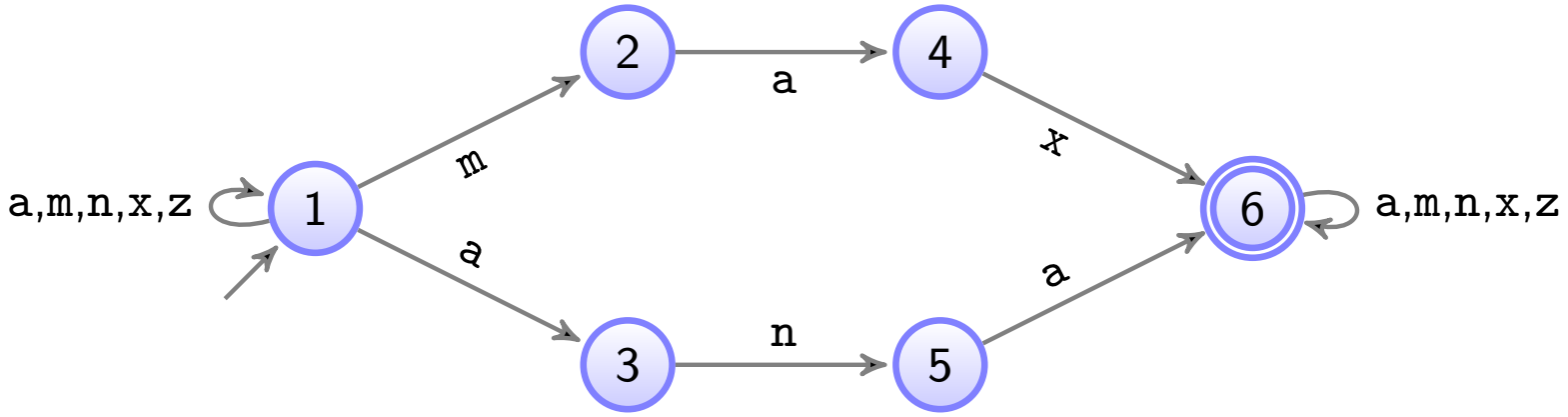
Alle Wörter, mit denen man vom Anfangszustand aus einen der Endzustände erreicht.

Beispiel: Suche nach Max und Ana

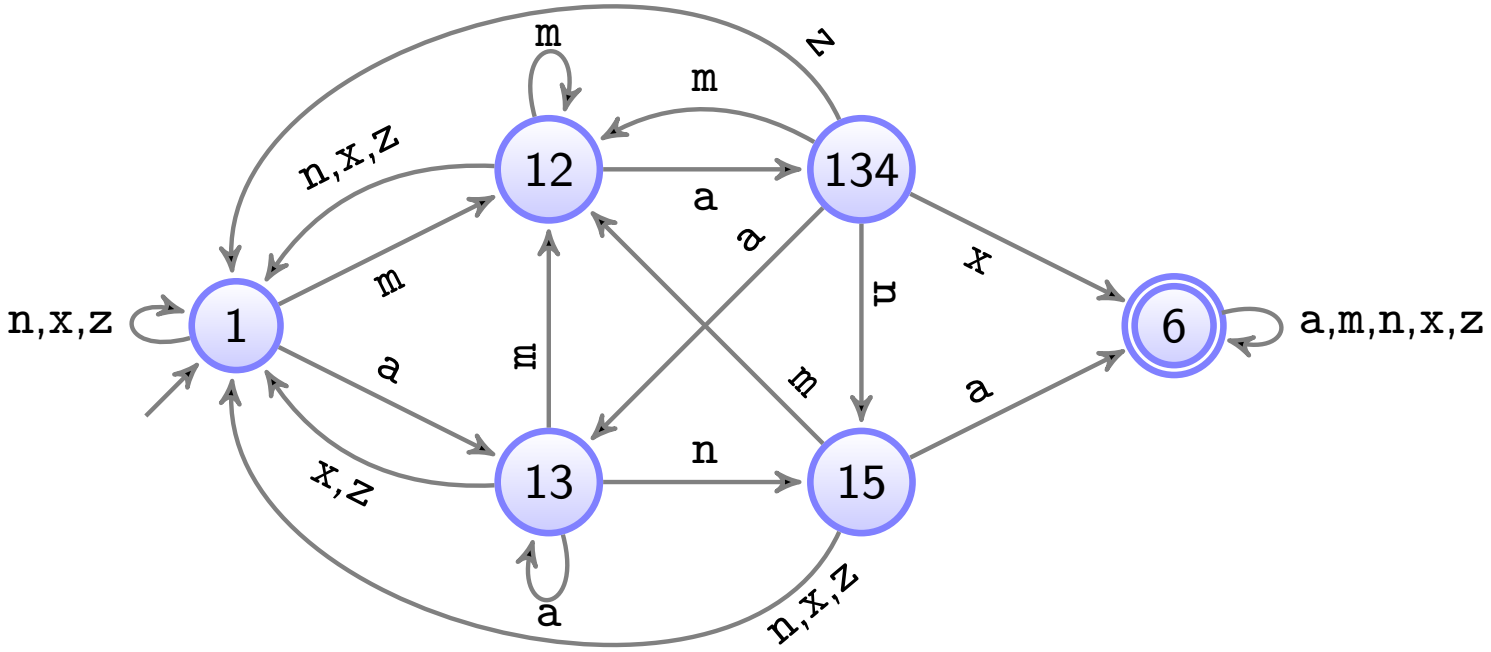
Gesucht: Automat zur Suche nach „max“ und „ana“ in einem Text

$\Sigma = \{\mathtt{a}, \mathtt{m}, \mathtt{n}, \mathtt{x}, \mathtt{z}\}$ (z ... Stellvertreter für b–l, o–w, y, z, ...)

NEA:



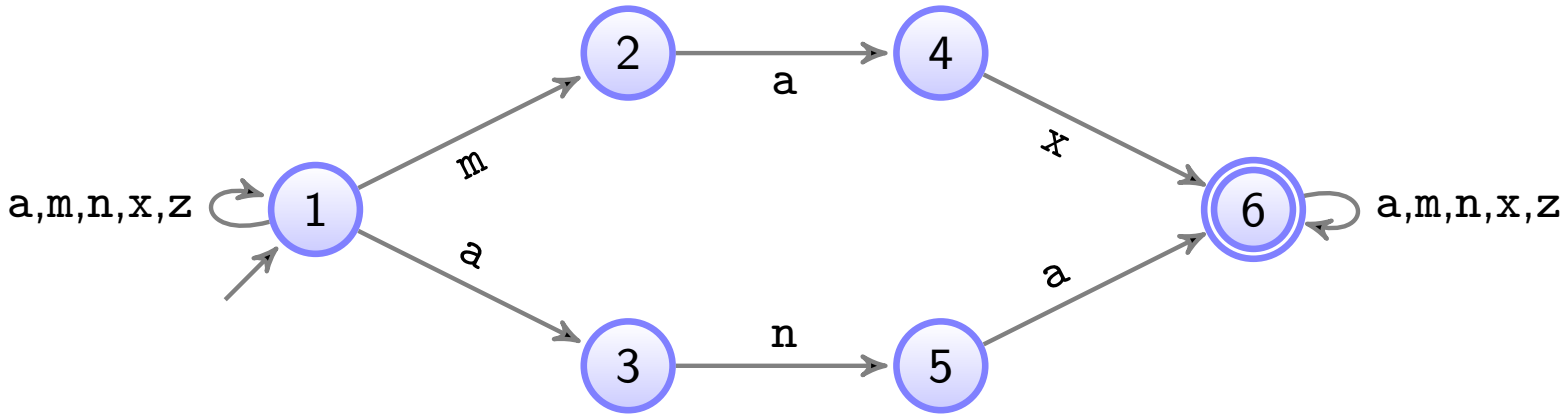
DEA:



Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten
   - 4.6. Determinisierung
   - 4.7. Transducer
   - 4.8. Büchi-Automaten
5. Reguläre Sprachen

Beispiel: Suche nach Max und Ana

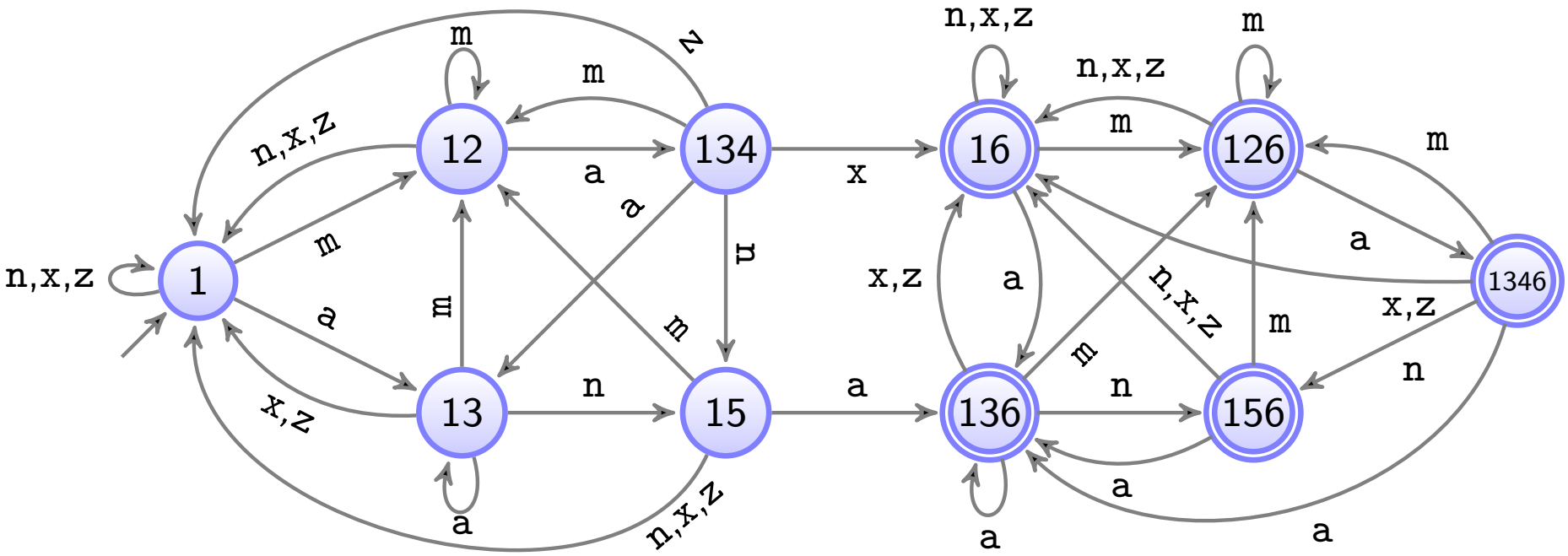


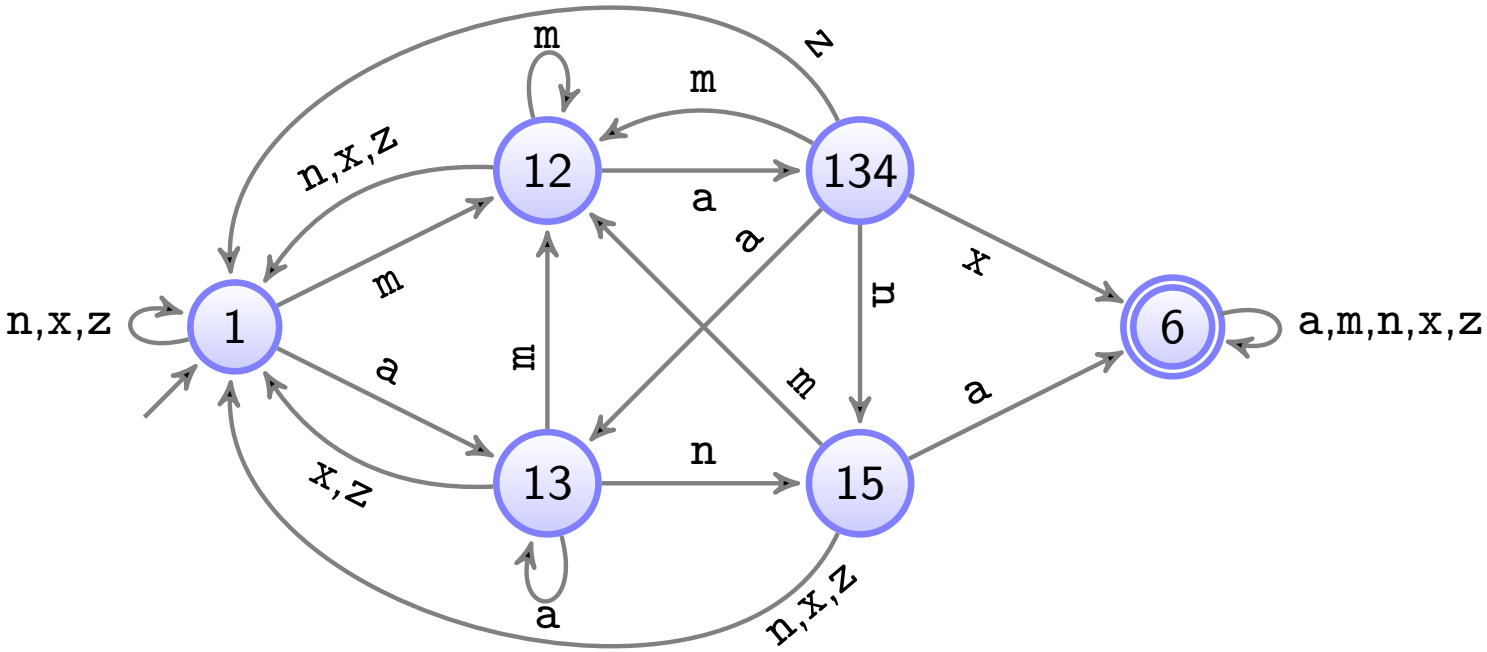
	δ^*	a	m	n	x	z
SZ	1	$\{1,3\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	2	$\{4\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
	3	$\{\}$	$\{\}$	$\{5\}$	$\{\}$	$\{\}$
	4	$\{\}$	$\{\}$	$\{\}$	$\{6\}$	$\{\}$
	5	$\{6\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
EZ	6	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$

Identisch mit der Tabelle für δ, wenn es keine ε-Kanten gibt.

	δ^*	a	m	n	x	z
SZ	1	$\{1,3\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	2	$\{4\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
	3	$\{\}$	$\{\}$	$\{5\}$	$\{\}$	$\{\}$
	4	$\{\}$	$\{\}$	$\{\}$	$\{6\}$	$\{\}$
	5	$\{6\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
EZ	6	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$

	$\widehat{\delta}$	a	m	n	x	z
SZ	$\{1\}$	$\{1,3\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	$\{1,2\}$	$\{1,3,4\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	$\{1,3\}$	$\{1,3\}$	$\{1,2\}$	$\{1,5\}$	$\{1\}$	$\{1\}$
	$\{1,3,4\}$	$\{1,3\}$	$\{1,2\}$	$\{1,5\}$	$\{1,6\}$	$\{1\}$
	$\{1,5\}$	$\{1,3,6\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
EZ	$\{1,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,3,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,5,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,2,6\}$	$\{1,3,4,6\}$	$\{1,2,6\}$	$\{1,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,5,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,3,4,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,5,6\}$	$\{1,6\}$	$\{1,6\}$


n,x,z
1
12
134
13
15
16
126
1346
136
156
m
a
n
x
z
x,z
n,x,z


n,x,z
1
12
134
13
15
6
a,m,n,x,z
m
a
n
x
z
x,z
n,x,z

Determinisierung

Gegeben: NEA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ mit $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

Gesucht: DEA $\widehat{\mathcal{A}} = \langle \widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q_0}, \widehat{F} \rangle$ mit $\widehat{\delta}\colon \widehat{Q} \times \Sigma \mapsto \widehat{Q}$,
sodass $\mathcal{A}$ und $\widehat{\mathcal{A}}$ dieselbe Sprache akzeptieren.

Wir definieren den deterministischen Automaten $\widehat{\mathcal{A}}$ folgendermaßen:

- $\widehat{Q} = 2^Q$
- $\widehat{q_0} = \{q_0\}$
- $\widehat{F} = \begin{cases} \{\, \widehat{q} \in \widehat{Q} \mid \widehat{q} \cap F \neq \emptyset \,\} \cup \{\widehat{q_0}\} & \text{falls } \varepsilon \in \mathcal{L}(\mathcal{A}) \\ \{\, \widehat{q} \in \widehat{Q} \mid \widehat{q} \cap F \neq \emptyset \,\} & \text{sonst} \end{cases}$
- Für alle Zustände $\widehat{q} \in \widehat{Q}$ und alle Symbole $s \in \Sigma$:

  $\widehat{\delta}(\widehat{q}, s) = \{\, q' \in Q \mid \text{es gibt } q \in \widehat{q}, \text{ sodass } (q, s, q') \in \delta^* \,\}$

$\mathcal{A}$ und $\widehat{\mathcal{A}}$ sind äquivalent, d.h., sie akzeptieren dieselbe Sprache:
$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$.

Anmerkungen

Neue Endzustände: Ein neuer Zustand $\widehat{q}$ ist Endzustand, ...

- wenn seine Bezeichnung einen der alten Endzustände enthält, d.h., wenn $\widehat{q} \cap F \neq \emptyset$, oder
- wenn es sich um den neuen Startzustand handelt und der alte Automat das Leerwort akzeptiert,
  d.h., wenn $\widehat{q} = \widehat{q_0}$ und $\varepsilon \in \mathcal{L}(\mathcal{A})$,
  d.h., wenn $\widehat{q} = \widehat{q_0}$ und $(q_0, \varepsilon, f) \in \delta^*$ für einen Endzustand $f \in F$.

Neue Übergangsfunktion: Wegen

$$\widehat{\delta}(\widehat{q}, s) = \bigcup_{q \in \widehat{q}} \{\, q' \in Q \mid (q, s, q') \in \delta^* \,\} = \bigcup_{q \in \widehat{q}} \delta^*(q, s)$$

spart es Arbeit, wenn man zuerst $\delta^*(q, s)$ für alle $q \in Q$ und alle $s \in \Sigma$ berechnet. Danach müssen nur mehr die Zeilen, die $\widehat{q}$ entsprechen, vereinigt werden.

Neue Zustände: Betrachte nur jene Zustände aus 2^Q, die von $\widehat{q_0}$ aus erreichbar sind.

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten
   - 4.6. Determinisierung
   - 4.7. Transducer
     - 4.7.1. Endliche Transducer
     - 4.7.2. Mealy-Automaten
     - 4.7.3. Moore-Automaten
   - 4.8. Büchi-Automaten
5. Reguläre Sprachen

Transducer

Endlicher Transducer

... wird beschrieben durch ein 6-Tupel $\mathcal{A} = \langle Q, \Sigma, \Gamma, \delta, I, F \rangle$, wobei

- Q, Σ, F ... siehe DEAs
- Γ ... Ausgabealphabet (*output alphabet*)
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ ... Übergangsrelation
- $I \subseteq Q$ ... Anfangszustände

Erweiterte Übergangsrelation $\delta^* \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$

δ^* ist die kleinste Menge mit folgenden Eigenschaften:

- $(q, \varepsilon, \varepsilon, q) \in \delta^*$ für alle $q \in Q$
- $(q_1, w, w', q_2) \in \delta^*,\ (q_2, s, s', q_3) \in \delta \quad \Longrightarrow \quad (q_1, ws, w's', q_3) \in \delta^*$

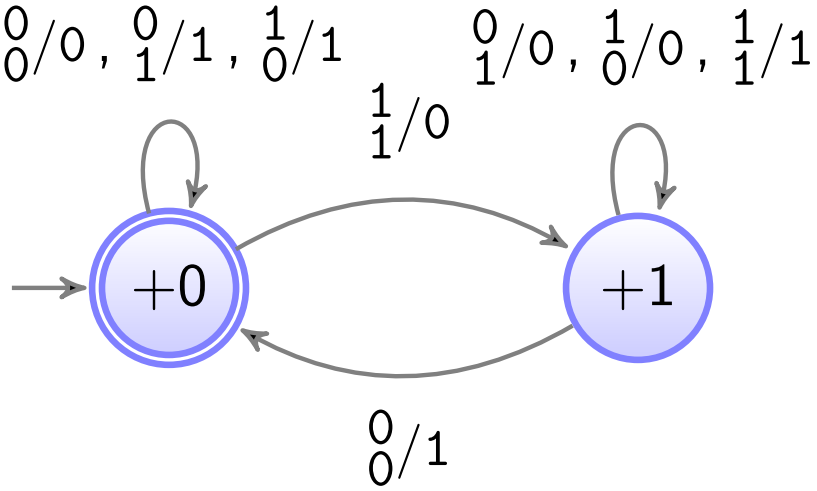
Übersetzungsrelation $[\mathcal{A}] \subseteq \Sigma^* \times \Gamma^*$

$[\mathcal{A}] = \{\, (w, w') \in \Sigma^* \times \Gamma^* \mid (i, w, w', f) \in \delta^* \text{ für ein } i \in I \text{ und ein } f \in F \,\}$

Beispiel: Binäraddition von rechts nach links

$$\begin{array}{cccccc|l} 0 & 0 & 1 & 0 & 1 & 1 & = 11_{10} \\ 0 & 0 & 0 & 1 & 0 & 1 & = 5_{10} \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & = 16_{10} \end{array}$$

$\longleftarrow$



$\mathcal{A} = \langle \{+0, +1\}, \{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}, \{0, 1\}, \delta, \{+0\}, \{+0\} \rangle$, wobei

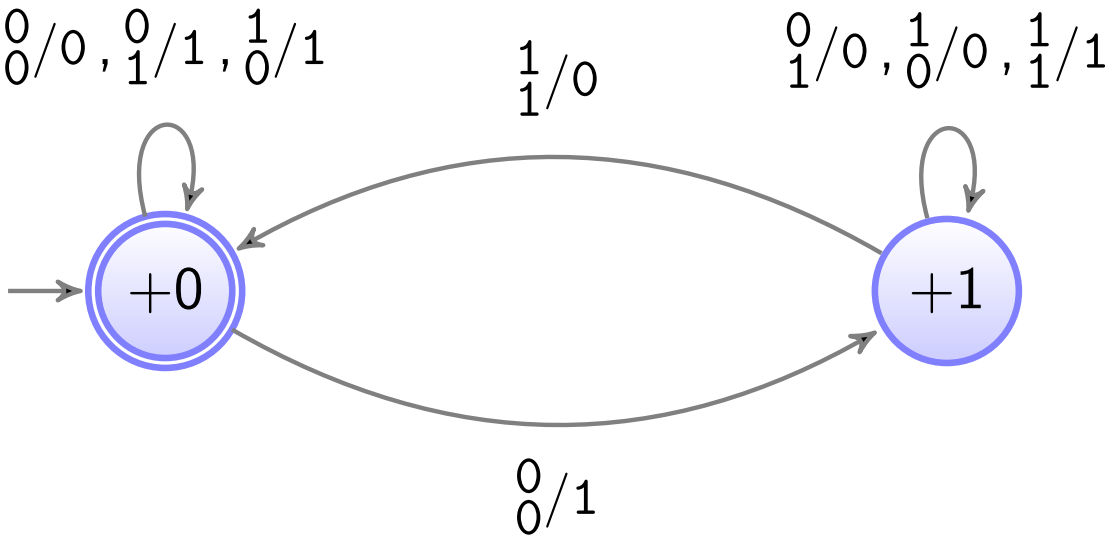
δ	$\binom{0}{0}$	$\binom{0}{1}$	$\binom{1}{0}$	$\binom{1}{1}$
$+0$	$\{(0, +0)\}$	$\{(1, +0)\}$	$\{(1, +0)\}$	$\{(0, +1)\}$
$+1$	$\{(1, +0)\}$	$\{(0, +1)\}$	$\{(0, +1)\}$	$\{(1, +1)\}$

$$[\mathcal{A}] = \{ (\varepsilon, \varepsilon), (\tbinom{0}{0}, 0), (\tbinom{0}{1}, 1), (\tbinom{1}{0}, 1), (\tbinom{0\,0}{0\,0}, 00), (\tbinom{0\,0}{0\,1}, 01), \ldots, (\tbinom{1\,0}{1\,0}, 01), \ldots, (\tbinom{1\,0\,1\,0}{1\,0\,1\,0}, 0101), \ldots \}$$

Binäraddition von links nach rechts

$$\begin{array}{cccccc|l} 0 & 0 & 1 & 0 & 1 & 1 & = 11_{10} \\ 0 & 0 & 0 & 1 & 0 & 1 & = 5_{10} \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & = 16_{10} \end{array}$$

$\longrightarrow$



+0 ... kein Übertrag
+1 ... Übertrag

Achtung, Indeterminismus! Zustand „+0“ besitzt bei Eingabe $\frac{0}{0}$ zwei Folgezustände, ebenso Zustand „+1“ bei Eingabe $\frac{1}{1}$.

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten
   - 4.6. Determinisierung
   - 4.7. Transducer
     - 4.7.1. Endlicher Transducer
     - 4.7.2. Mealy-Automaten
     - 4.7.3. Moore-Automaten
   - 4.8. Büchi-Automaten
5. Reguläre Sprachen

Mealy-Automat

... wird beschrieben durch ein 6-Tupel $\mathcal{A} = \langle Q, \Sigma, \Gamma, \delta, \gamma, q_0 \rangle$, wobei

- Q, Σ, δ, q_0 ... siehe DEAs
- Γ ... Ausgabealphabet (*output alphabet*)
- $\gamma\colon Q \times \Sigma \mapsto \Gamma$ ... Ausgabefunktion (*output function*)

Erweiterte Übergangsfunktion $\delta^*\colon Q \times \Sigma^* \mapsto Q$ siehe DEA.

Erweiterte Ausgabefunktion $\gamma^*\colon Q \times \Sigma^* \mapsto \Gamma^*$

$\gamma^*(q, \varepsilon) = \varepsilon$ für alle $q \in Q$, $s \in \Sigma$, $w \in \Sigma^*$

$\gamma^*(q, sw) = \gamma(q, s) \cdot \gamma^*(\delta(q, s), w)$

Übersetzungsfunktion $[\mathcal{A}]\colon \Sigma^* \mapsto \Gamma^*$

$[\mathcal{A}](w) = \gamma^*(q_0, w)$

Mealy-Automaten sind ein Spezialfall von Transducern:

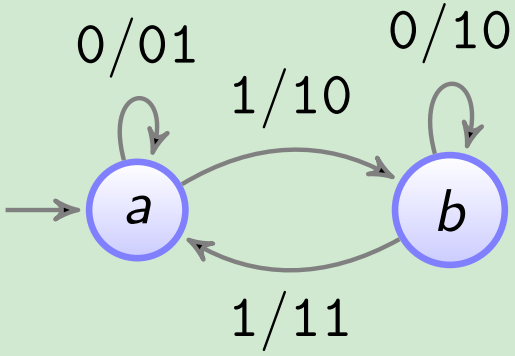
- Nur ein Anfangszustand: $I = \{q_0\}$
- Die Übergangsrelation ist deterministisch:
  - Der Folgezustand $\delta(q, s)$ ist eindeutig durch q und s bestimmt.
  - Keine ε-Übergänge
- Relationstupel: $(q, s, \gamma(q, s), \delta(q, s))$
- Alle Zustände sind Endzustände: $F = Q$

$(1:2)$-$(0,1)$-RLL-Encoder als Mealy-Automat

$\mathcal{A} = \langle \{a, b\}, \{0, 1\}, \{01, 10, 11\}, \delta, \gamma, a \rangle$

δ	0	1
a	a	b
b	b	a

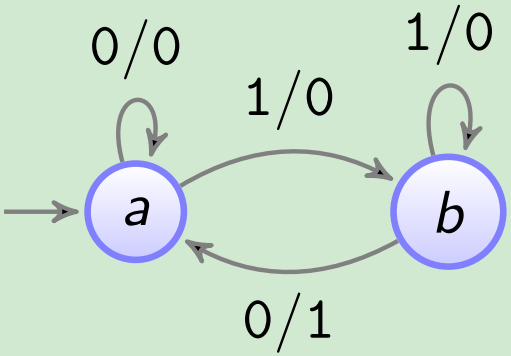
γ	0	1
a	01	10
b	10	11



w:	ε	0	1	00	10	01	11	000	100	$\cdots$
$[\mathcal{A}](w)$:	ε	01	10	0101	1010	0110	1011	010101	101010	$\cdots$

Detektor für fallende Flanken

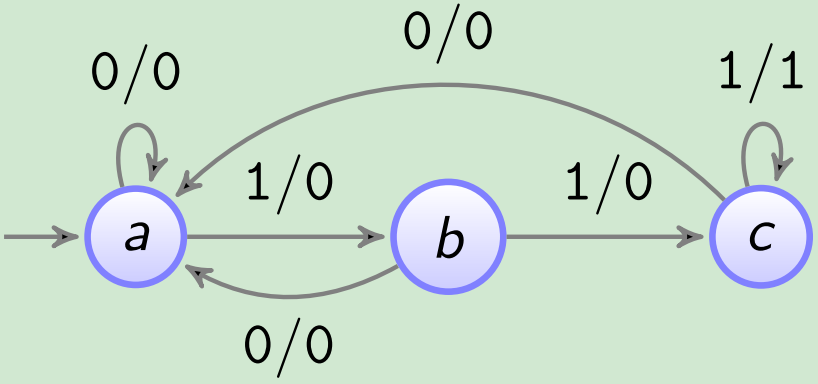
Ausgabe 1, wenn in der Eingabe ein Wechsel von 1 auf 0 stattfindet.



w:	$00110001010\cdots$
$[\mathcal{A}](w)$:	$00001000101\cdots0$

Detektor für 111-Blöcke

Ausgabe 1, wenn in der Eingabe drei 1er aufeinander folgen.



w:	$00110111110101110\cdots$
$[\mathcal{A}](w)$:	$00000001110000010\cdots$

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten
   - 4.6. Determinisierung
   - 4.7. Transducer
     - 4.7.1. Endlicher Transducer
     - 4.7.2. Mealy-Automaten
     - 4.7.3. Moore-Automaten
   - 4.8. Büchi-Automaten
5. Reguläre Sprachen

Moore-Automat

... wird beschrieben durch ein 6-Tupel $\mathcal{A} = \langle Q, \Sigma, \Gamma, \delta, \gamma, q_0 \rangle$, wobei

- Q, Σ, δ, q_0 ... siehe DEAs
- Γ ... Ausgabealphabet (*output alphabet*)
- $\gamma\colon Q \mapsto \Gamma$ ... Ausgabefunktion (*output function*)

(Mealy: $\gamma\colon Q \times \Sigma \mapsto \Gamma$)

Erweiterte Übergangsfunktion $\delta^*\colon Q \times \Sigma^* \mapsto Q$ siehe DEA.

Erweiterte Ausgabefunktion $\gamma^*\colon Q \times \Sigma^* \mapsto \Gamma^*$

$\gamma^*(q, \varepsilon) = \gamma(q)$ für alle $q \in Q$, $s \in \Sigma$, $w \in \Sigma^*$

$\gamma^*(q, sw) = \gamma(q) \cdot \gamma^*(\delta(q, s), w)$

(Mealy: $\gamma^*(q, sw) = \gamma(q, s) \cdot \gamma^*(\delta(q, s), w)$)

Übersetzungsfunktion $[\mathcal{A}]\colon \Sigma^* \mapsto \Gamma^*$

$[\mathcal{A}](w) = \gamma^*(q_0, w)$

Moore-Automaten sind (fast) ein Spezialfall von Transducern:

- Nur ein Anfangszustand: $I = \{q_0\}$
- Alle Übergange in einen Zustand geben dasselbe Symbol aus.
- Die Übergangsrelation ist deterministisch:
  - Der Folgezustand $\delta(q, s)$ ist eindeutig durch q und s bestimmt.
  - Keine ε-Übergänge
- Relationstupel: $(q, s, \gamma(q), \delta(q, s))$
- Alle Zustände sind Endzustände: $F = Q$

Vergleich von Moore- und Mealy-Automaten

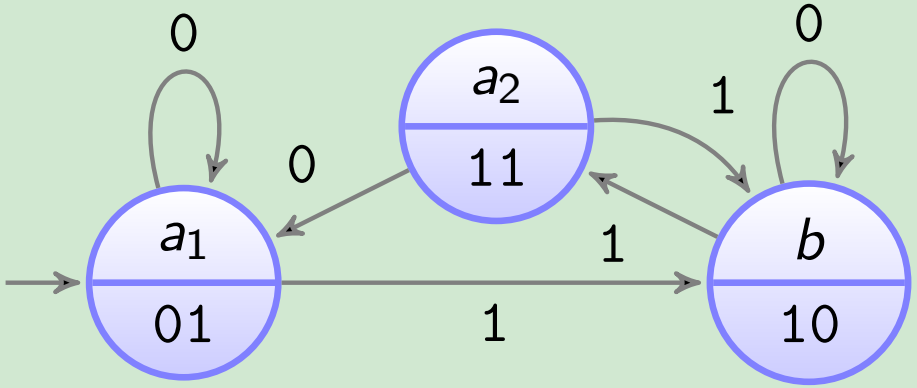
- Die Ausgabe erfolgt
  - ... bei Moore-Automaten durch den momentanen Zustand.
  - ... bei Mealy-Automaten beim Zustandswechsel, der durch Ursprungszustand und Eingabe festgelegt ist.
- Moore- und Mealy-Automaten besitzen dieselbe Ausdrucksstärke, sind aber schwächer als Transducer.
- Moore-Automaten haben in der Regel mehr Zustände als Mealy-Automaten.

$(1:2)$-$(0,1)$-RLL-Encoder als Moore-Automat

$\mathcal{A} = \langle \{a_1, a_2, b\}, \{0,1\}, \{01, 10, 11\}, \delta, \gamma, a \rangle$

δ	0	1
a_1	a_1	b
a_2	a_1	b
b	b	a_2

γ	
a_1	01
a_2	11
b	10

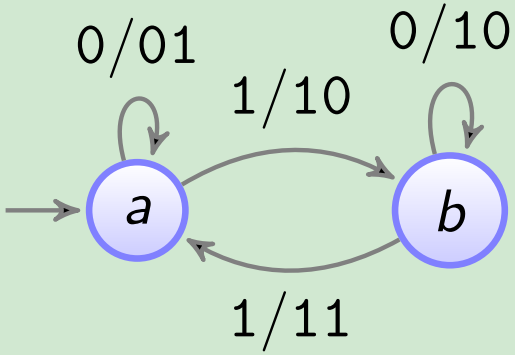


w:	ε	0	1	00	10	01	$\cdots$	100	$\cdots$
$[\mathcal{A}](w)$:	01	0101	0110	010101	011010	0110	$\cdots$	01101010	$\cdots$

Zum Vergleich: Mealy-Automat

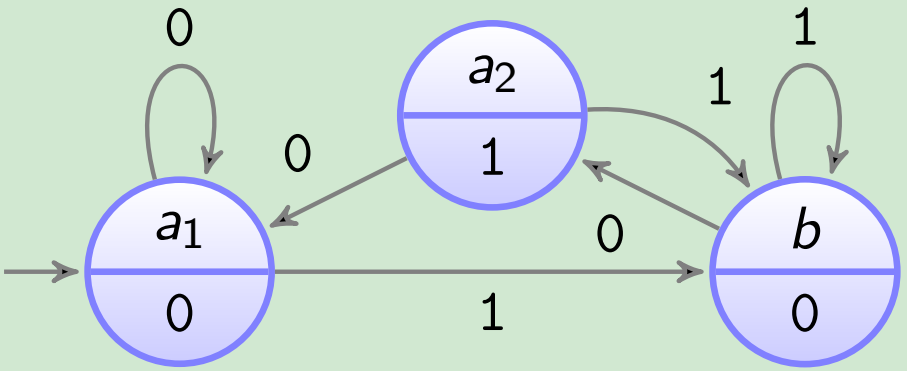
δ	0	1
a	a	b
b	b	a

γ	0	1
a	01	10
b	10	11



Detektor für fallende Flanken

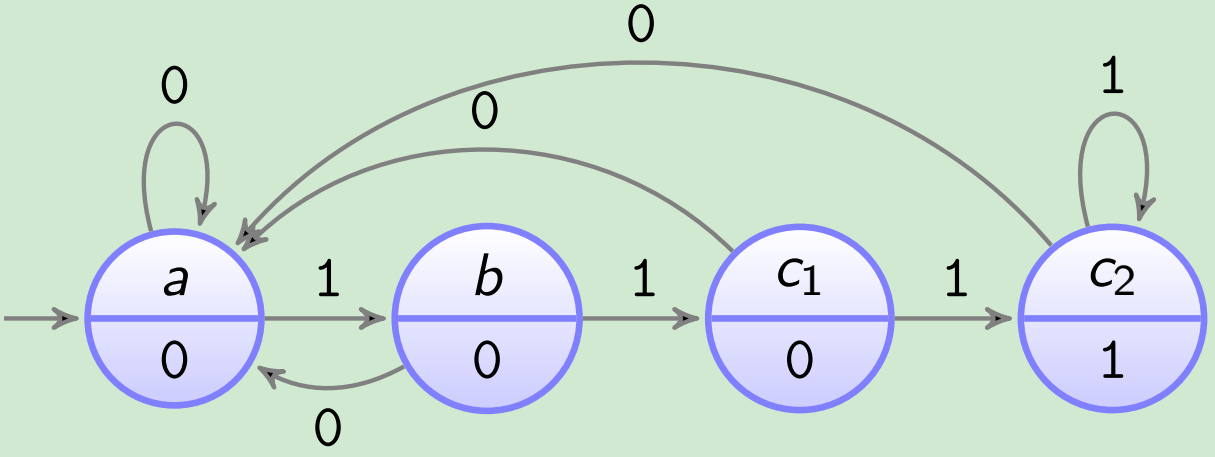
Ausgabe 1, wenn in der Eingabe ein Wechsel von 1 auf 0 stattfindet.



w:	$00110001010\cdots$
$[\mathcal{A}](w)$:	$000001000101\cdots$

Detektor für 111-Blöcke

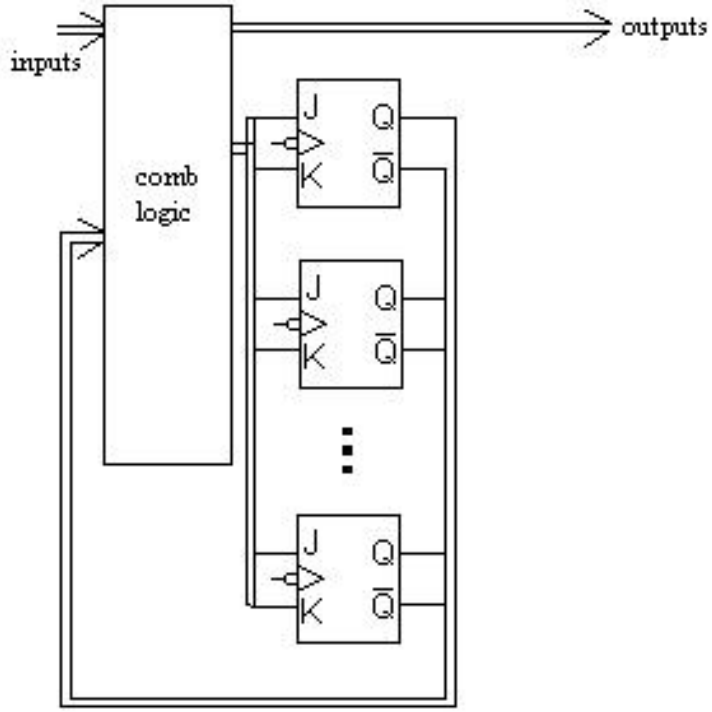
Ausgabe 1, wenn in der Eingabe drei 1er aufeinander folgen.



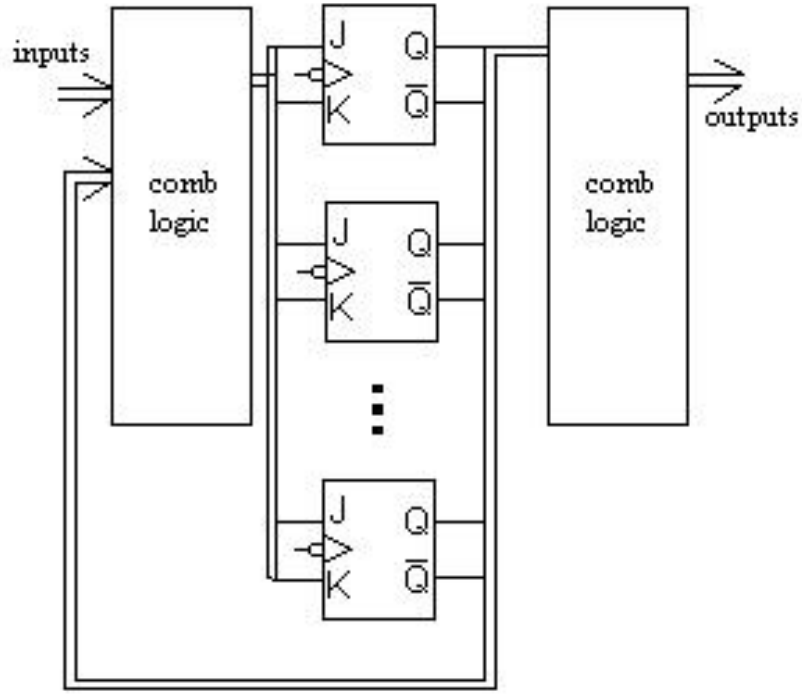
w:	$00110111110101110\cdots$
$[\mathcal{A}](w)$:	$000000001110000010\cdots$

Relevanz von Mealy- und Moore-Automaten

Mealy-Schaltwerk



Moore-Schaltwerk



- „inputs“ stammen aus dem Eingabealphabet Σ
- „outputs“ stammen aus dem Ausgabealphabet Γ
- Flip-Flops speichern den Zustand aus Q; Reset: Anfangszustand q_0
- „combination logic“: realisiert die Übergangsfunktionen δ und γ.

Graphiken: Ken Reid, Indiana University – Purdue University Indianapolis

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
   - 4.1. Beispiele
   - 4.2. Klassifikation
   - 4.3. Grundlagen formaler Sprachen
   - 4.4. Deterministische endliche Automaten
   - 4.5. Nichtdeterministische endliche Automaten
   - 4.6. Determinisierung
   - 4.7. Transducer
   - 4.8. Büchi-Automaten
5. Reguläre Sprachen

Büchi-Automaten

Deterministischer Büchi-Automat

... wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, wobei

- Q, Σ, δ, q_0, F wie bei DEAs definiert sind.

Σ^ω ... Menge aller unendlichen Wörter ($= \omega$-Wörter) über Σ

Akzeptanz von Wörtern

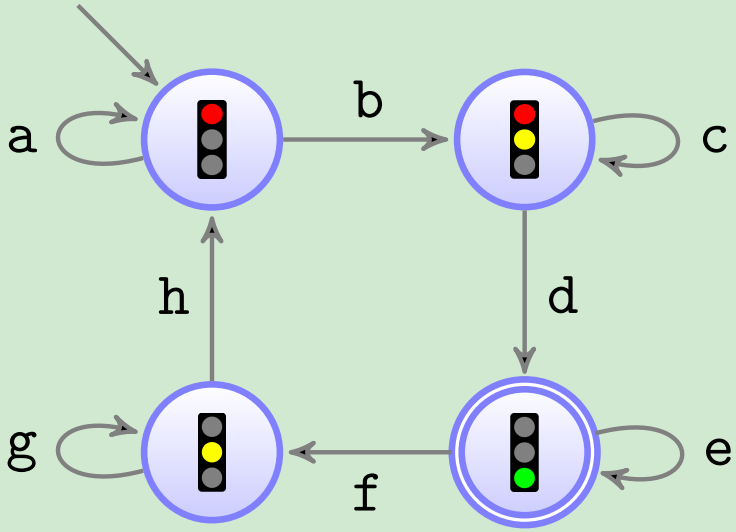
Ein deterministischer Büchi-Automat $\mathcal{A}$ akzeptiert ein Wort $s_1 s_2 s_3 \cdots \in \Sigma^\omega$, wenn es Zustände $q_0, q_1, q_2, q_3, \cdots \in Q$ gibt, sodass

- $q_0 \in Q$ der Startzustand ist,
- $\delta(q_{i-1}, s_i) = q_i$ für alle $i \in \mathbb{N}$ gilt und
- es unendlich viele i gibt, sodass q_i ein Endzustand ist ($q_i \in F$).

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^\omega \mid w \text{ wird von } \mathcal{A} \text{ akzeptiert} \}$

Faire Verkehrsampel



Der Automat akzeptiert genau jene Wörter aus $\{\mathtt{a}, \ldots, \mathtt{h}\}^\omega$, bei denen es immer wieder grün wird.

Nichtdeterministischer Büchi-Automat

... wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$, wobei

- Q, Σ, F ... siehe DEA
- $I \subseteq Q$ ... Anfangszustände
- $\delta \subseteq Q \times \Sigma \times Q$ ... Übergangsrelation

Akzeptanz von Wörtern

Ein nichtdeterministischer Büchi-Automat $\mathcal{A}$ akzeptiert ein Wort $s_1 s_2 s_3 \cdots \in \Sigma^\omega$, wenn es Zustände $q_0, q_1, q_2, q_3, \cdots \in Q$ gibt, sodass

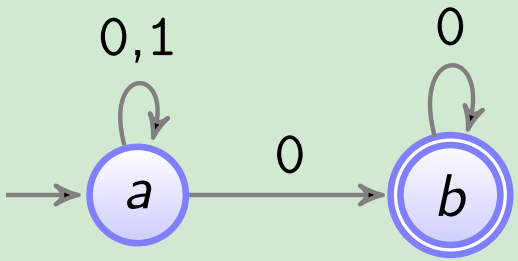
- $q_0 \in Q$ ein Startzustand ist ($q_0 \in I$),
- $(q_{i-1}, s_i, q_i) \in \delta$ für alle $i \in \mathbb{N}$ gilt und
- es unendlich viele i gibt, sodass q_i ein Endzustand ist ($q_i \in F$).

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^\omega \mid w \text{ wird von } \mathcal{A} \text{ akzeptiert} \}$

Nur endlich viele 1er

Gesucht: Ein Büchi-Automat, der genau jene ω-Wörter über $\{0, 1\}$ akzeptiert, die nur eine endliche Anzahl an 1ern enthalten.



Kein deterministischer Büchi-Automaten akzeptiert diese Sprache.
$\Longrightarrow$ Nichtdeterministische Büchi-Automaten sind ausdrucksstärker als deterministische!

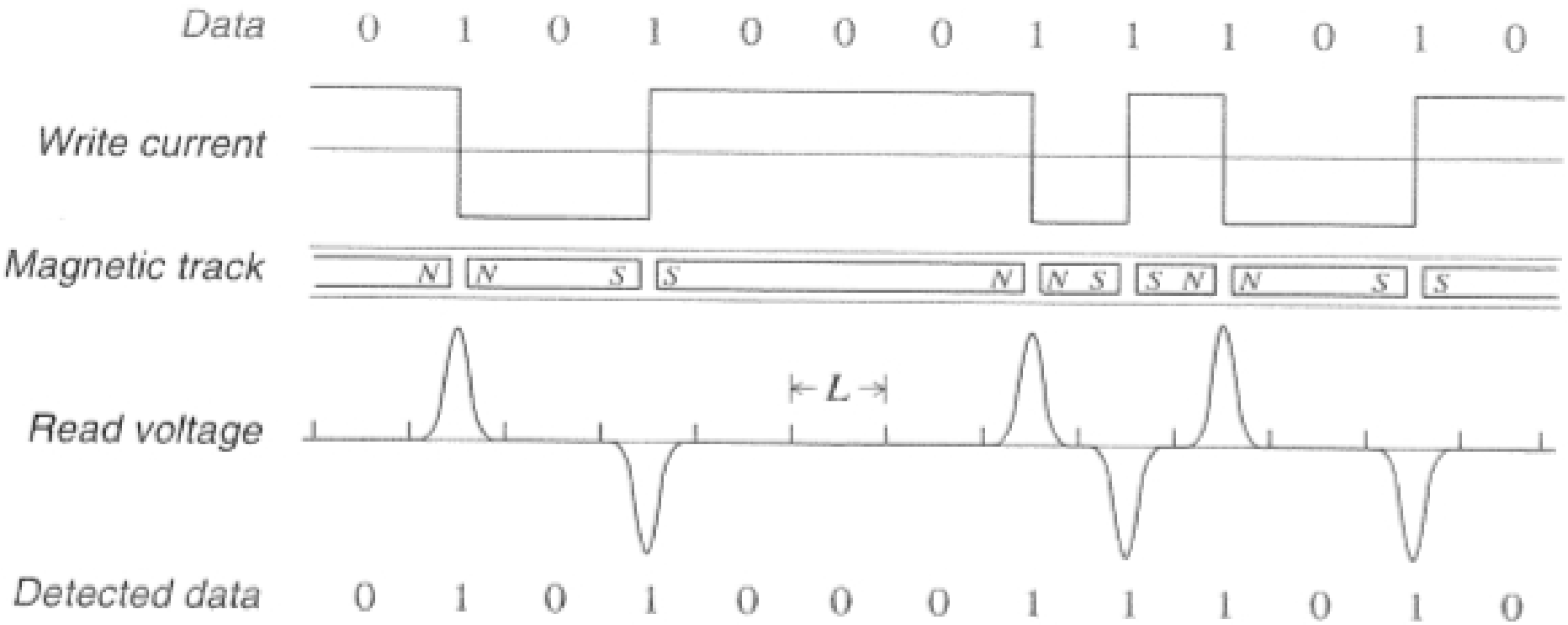
Modellierung mit endlichen Automaten

Vorgangsweise:

1. Was sind die Zustände des Systems? Wieviele sind notwendig? Zustandsbezeichnungen?
2. Startzustand? Endzustände?
3. Was sind die Aktionen/Eingaben, die zu Zustandsübergängen führen? Bezeichnung?
4. Was sind die Aktionen/Ausgaben, die bei Zustandsübergängen stattfinden? Bezeichnung?
5. Lege für jeden Zustand und jede Eingabe die Folgezustände und die Ausgaben fest.

Anwendung: RLL (run-length limited) Codes

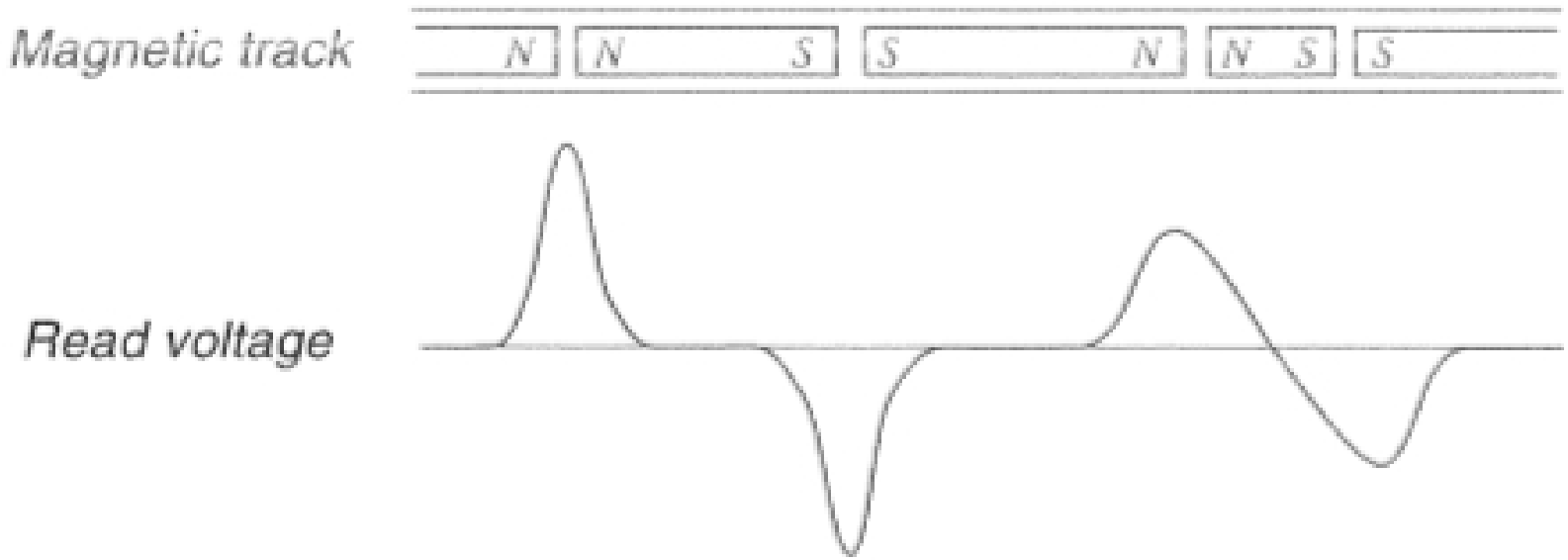
- Daten werden auf Festplatten, CDs und DVDs binär kodiert.
- Binär-1: Magnetisierungswechsel bzw. Vertiefung („pit“)



Graphik aus: D.Lind, B.Marcus: An Introduction to Symbolic Dynamics and Coding, Cambridge Unversity Press 1995.

Probleme:

- 1er zu weit auseinander: Synchronisation geht verloren.
- 1er zu dicht: werden ununterscheidbar bzw. schwächen sich ab.



Graphik aus: D.Lind, B.Marcus: An Introduction to Symbolic Dynamics and Coding, Cambridge Unversity Press 1995.

Einfache Lösung:

- Füge nach jedem Datenbit ein zusätzliches Synchronisationsbit ein.
- Wähle Bitabstand so groß, dass 1er-Folgen richtig erkannt werden.

$(m : n)$-(d, k)-RLL-Codes

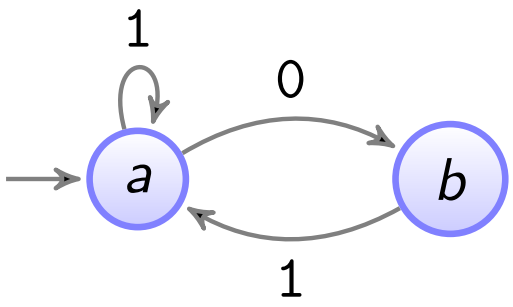
Nachteil der einfachen Lösung: Benötigt mehr Platz als notwendig.

Idee: Nütze die „natürlichen“ 0er-Strecken und 1er im Datenstrom.
m Datenbits werden so in n Codebits übersetzt, dass zwischen zwei 1ern mindestens d und höchstens k 0er auftreten.

- Existiert nur für bestimmte Werte von m, n, d, k
- Ziel: minimiere n/m, maximiere d

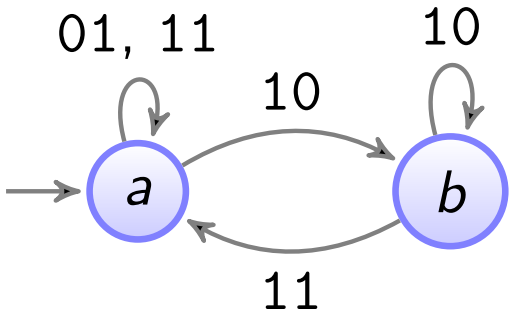
DVD: $(8 : 16)$-$(2, 10)$-RLL-Code

$(1:2)$-$(0,1)$-RLL-Code: 2 Codebits/Datenbit, max. ein 0er zwischen 1ern.



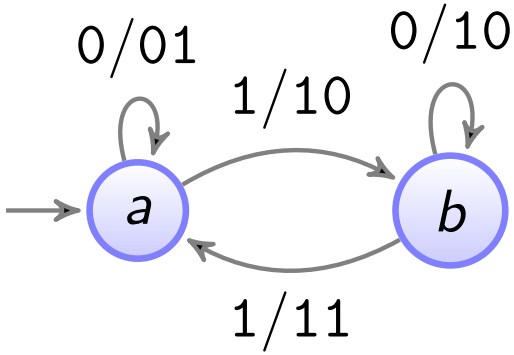
Zustand a: zwei Codeworte für zwei Datenbits – reicht.
Zustand b: ein Codewort für zwei Datenbits – reicht nicht.

„Potenzieren“ des Automaten:

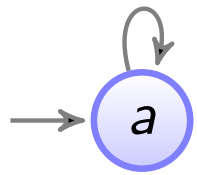


Zustand a: drei Codeworte für zwei Datenbits – reicht.
Zustand b: zwei Codeworte für zwei Datenbits – reicht.

Codierungsautomat (Mealy-Automat):



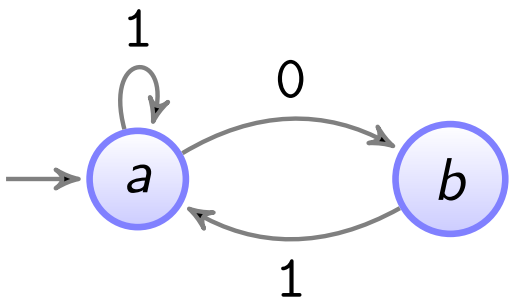
oder



oder . . .

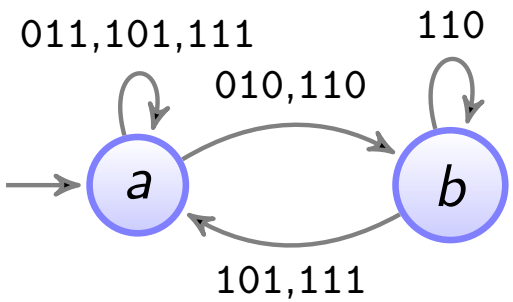
10010 $\Rightarrow$ 10 10 10 11 01 oder 11 01 01 11 01 oder . . .

$(2:3)$-$(0,1)$-RLL-Code: 3 Code-/2 Datenbits, max. ein 0er zwischen 1ern.



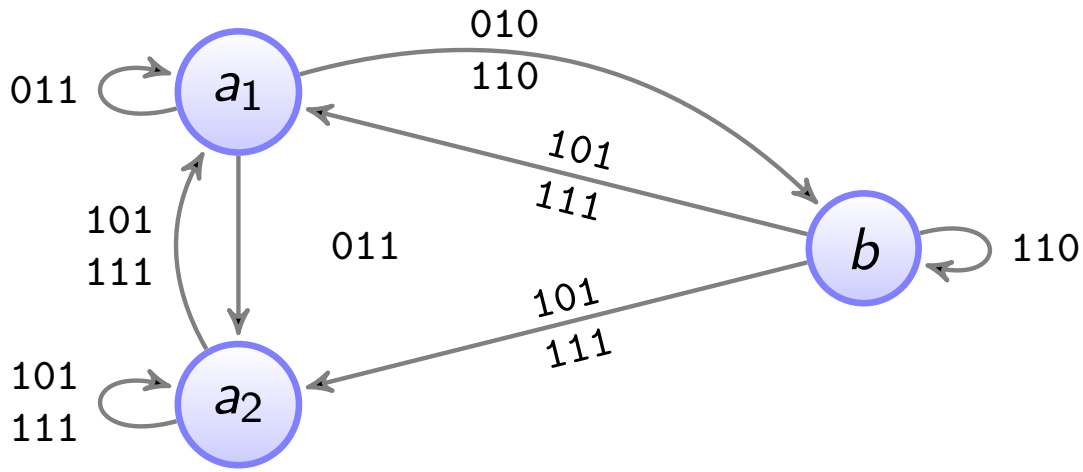
Zustand *a*: zwei Code- für vier Datenworte – reicht nicht.
Zustand *b*: ein Codewort für vier Datenworte – reicht nicht.

Dritte Potenz des Automaten:

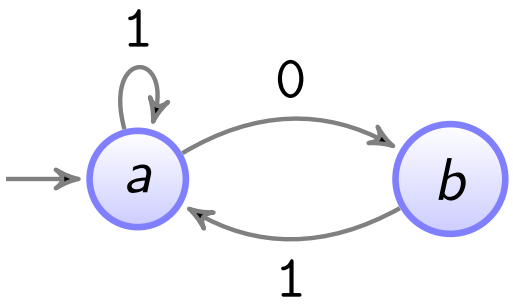


Zustand *a*: fünf Code- für vier Datenworte – reicht.
Zustand *b*: drei Code- für vier Datenworte – reicht nicht.

Teilung von Zustand *a*:

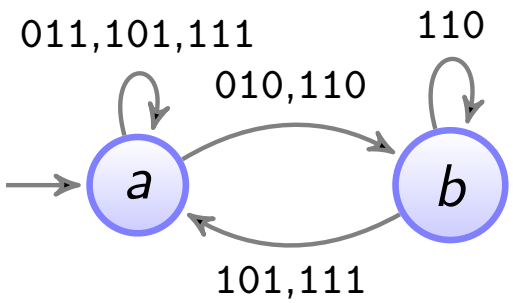


$(2:3)$-$(0,1)$-RLL-Code: 3 Code-/2 Datenbits, max. ein 0er zwischen 1ern.



Zustand a: zwei Code- für vier Datenworte – reicht nicht.
Zustand b: ein Codewort für vier Datenworte – reicht nicht.

Dritte Potenz des Automaten:



Zustand a: fünf Code- für vier Datenworte – reicht.
Zustand b: drei Code- für vier Datenworte – reicht nicht.

Teilung von Zustand a:

Mealy-Automat