3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

20.3.2019

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

Normalformen

Literal: Variable oder negierte Variable, also A, $\neg A$, B, $\neg B$, ...

Negationsnormalform (NNF)

- Literale sowie $\top$ und $\bot$ sind in NNF.
- $(F \wedge G)$ und $(F \vee G)$ sind in NNF, wenn F und G in NNF sind.
- Keine Formel sonst ist in NNF.

Disjunktive Normalform (DNF)

$\top$, $\bot$ sowie Disjunktionen von Konjunktion von Literalen:

$$(^{(\neg)}A_{1,1} \wedge {}^{(\neg)}A_{1,2} \wedge {}^{(\neg)}A_{1,3} \wedge \cdots) \vee ({}^{(\neg)}A_{2,1} \wedge {}^{(\neg)}A_{2,2} \wedge {}^{(\neg)}A_{2,3} \wedge \cdots) \vee \cdots$$

Konjunktive Normalform (KNF)

$\top$, $\bot$ sowie Konjunktionen von Disjunktion von Literalen:

$$(^{(\neg)}A_{1,1} \vee {}^{(\neg)}A_{1,2} \vee {}^{(\neg)}A_{1,3} \vee \cdots) \wedge ({}^{(\neg)}A_{2,1} \vee {}^{(\neg)}A_{2,2} \vee {}^{(\neg)}A_{2,3} \vee \cdots) \wedge \cdots$$

Konstruktion von DNFs/KNFs – Semantische Methode

Gegeben: Aussagenlogische Formel F
Gesucht: Äquivalente Formel in DNF/KNF

1. Stelle die zu F gehörige Funktion f als Wahrheitstafel dar.
2. Konstruiere DNF_f bzw. KNF_f.

A_1	A_2	A_3	$F := (A_1 \supset (A_2 \equiv A_3)) \wedge (\neg A_1 \supset (A_2 \wedge A_3))$		
1	1	1	1	K_{111}	
1	1	0	0		D_{110}
1	0	1	0		D_{101}
1	0	0	1	K_{100}	
0	1	1	1	K_{011}	
0	1	0	0		D_{010}
0	0	1	0		D_{001}
0	0	0	0		D_{000}

DNF: $F = K_{111} \vee K_{100} \vee K_{011}$
KNF: $F = D_{110} \wedge D_{101} \wedge D_{010} \wedge D_{001} \wedge D_{000}$

Konstruktion von DNFs/KNFs – Algebraische Methode

Gegeben: Aussagenlogische Formel F
Gesucht: Äquivalente Formel in DNF/KNF

1. Ersetze alle Junktoren durch $\wedge$, $\vee$ und $\neg$.
2. Verschiebe Negationen nach innen, eliminiere Doppelnegationen.
3. Wende das Distributivgesetz an.
4. Eliminiere $\top$ und $\bot$.

$(A_1 \supset (A_2 \equiv A_3)) \wedge (\neg A_1 \supset (A_2 \wedge A_3))$

1. $(\neg A_1 \vee (A_2 \wedge A_3) \vee (\neg A_2 \wedge \neg A_3)) \wedge (\neg\neg A_1 \vee (A_2 \wedge A_3))$
2. $(\neg A_1 \vee (A_2 \wedge A_3) \vee (\neg A_2 \wedge \neg A_3)) \wedge (A_1 \vee (A_2 \wedge A_3))$
3. DNF: $(\neg A_1 \wedge A_1) \vee (\neg A_1 \wedge A_2 \wedge A_3) \vee (A_2 \wedge A_3 \wedge A_1) \vee$
   $(A_2 \wedge A_3 \wedge A_2 \wedge A_3) \vee (\neg A_2 \wedge \neg A_3 \wedge A_1) \vee (\neg A_2 \wedge \neg A_3 \wedge A_2 \wedge A_3)$
   KNF: $(\neg A_1 \vee A_2 \vee \neg A_2) \wedge (\neg A_1 \vee A_2 \vee \neg A_3) \wedge (\neg A_1 \vee A_3 \vee \neg A_2) \wedge$
   $(\neg A_1 \vee A_3 \vee \neg A_3) \wedge (A_1 \vee A_2) \wedge (A_1 \vee A_3)$
4. DNF: $(\neg A_1 \wedge A_2 \wedge A_3) \vee (A_2 \wedge A_3) \vee (A_1 \wedge \neg A_2 \wedge \neg A_3)$
   KNF: $(\neg A_1 \vee A_2 \vee \neg A_3) \wedge (\neg A_1 \vee \neg A_2 \vee A_3) \wedge (A_1 \vee A_2) \wedge (A_1 \vee A_3)$

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

Das Erfüllbarkeitsproblem der Aussagenlogik

Erfüllbarkeitsproblem (Satisfiability, SAT)

Gegeben: aussagenlogische Formel F
Frage: Ist F erfüllbar, d.h., gibt es ein $I \in \mathcal{I}$, sodass $\text{val}_I(F) = 1$?

Effiziente Verfahren zur Lösung von SAT sind wichtig in der Praxis:

- Viele praktische Aufgaben sind Probleme der Aussagenlogik.
- Aussagenlogische Fragen lassen sich auf SAT zurückführen.

Unbrauchbare Verfahren:

- Wahrheitstafel: Prüfe, ob es Interpretation mit Ergebnis 1 gibt.
- Umwandlung in DNF: Antwort „nein“, wenn man $\bot$ erhält, sonst „ja“.

SAT-Solver: Programme, die SAT lösen.

- Lösen SAT mit fortgeschrittenen Methoden und Datenstrukturen.
- Können SAT für Formeln mit Millionen von Variablen lösen.
- Aber: immer noch ineffizient für bestimmte Formeltypen.

P = NP?

P: Klasse der Probleme, deren Lösungen sich polynomiell finden lassen.

NP: Klasse jener Probleme, deren Lösungen sich polynomiell verifizieren lassen; die Suche nach der Lösung kann aber aufwändig sein.

P versus NP

Gilt $\mathrm{P} = \mathrm{NP}$ oder $\mathrm{P} \neq \mathrm{NP}$ (gleichbedeutend mit $\mathrm{P} \subsetneq \mathrm{NP}$)?

SAT ist in NP: Gegeben I, lässt sich $\mathrm{val}_I(F) = 1$ leicht überprüfen.
Die Suche nach so einem I scheint aber aufwändig. Exponentiell?

SAT ist NP-vollständig: SAT gehört zu den schwierigsten Problemen in NP.
Kann man ein NP-vollständiges Problem effizient lösen, dann kann man alle Probleme in NP effizient lösen.

- SAT polynomiell lösbar $\Longrightarrow \mathrm{P} = \mathrm{NP}$
- SAT nicht polynomiell lösbar $\Longrightarrow \mathrm{P} \neq \mathrm{NP}$

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

Dualität von Funktionen, Operatoren und Formeln

Duale Funktionen

Zwei n-stellige Funktionen f und g heißen dual zueinander, wenn gilt: $\text{not}\, f(x_1, \ldots, x_n) = g(\text{not}\, x_1, \ldots, \text{not}\, x_n)$.

Duale Operatoren

Zwei Operatoren heißen dual, wenn die zugehörigen Funktionen dual sind.

Duale Formeln

Zwei Formeln $F[A_1, \ldots, A_n]$ und $G[A_1, \ldots, A_n]$ heißen dual zueinander, wenn gilt: $\neg F[A_1, \ldots, A_n] = G[\neg A_1, \ldots, \neg A_n]$

$\neg F[\neg A_1, \ldots, \neg A_n]$ ist dual zu $F[A_1, \ldots, A_n]$.

Sei G die Formel, die aus F durch Ersetzen aller Operatoren durch ihre dualen hervorgeht. Dann ist G dual zu F.

$F = G$ gilt genau dann, wenn $F^* = G^*$ gilt.

Was Sie heute erwartet

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. Beispiel: House
   - 3.8. Dualität von Funktionen, Operatoren und Formeln
   - 3.9. Gone Maggie gone
4. Endliche Automaten

Gone Maggie gone

„The Simpsons“, Staffel 20, Folge 13

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:

$n = 7$,

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

Gone Maggie gone (Fortsetzung)

„The Simpsons“, Staffel 20, Folge 13

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 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

The Simpsons – Modellierung als Automat

Systemkomponenten: Maggie (M), Knecht Ruprecht (K), Gift (G), Homer+Boot (H), linkes/rechtes Flussufer

Situationsbeschreibung (Systemzustand): $\frac{\text{Lebewesen/Dinge links}}{\text{Lebewesen/Dinge rechts}}$

(Wer/Was befindet sich momentan auf welcher Seite des Flusses?)

Anfangszustand: $\frac{MKGH}{}$

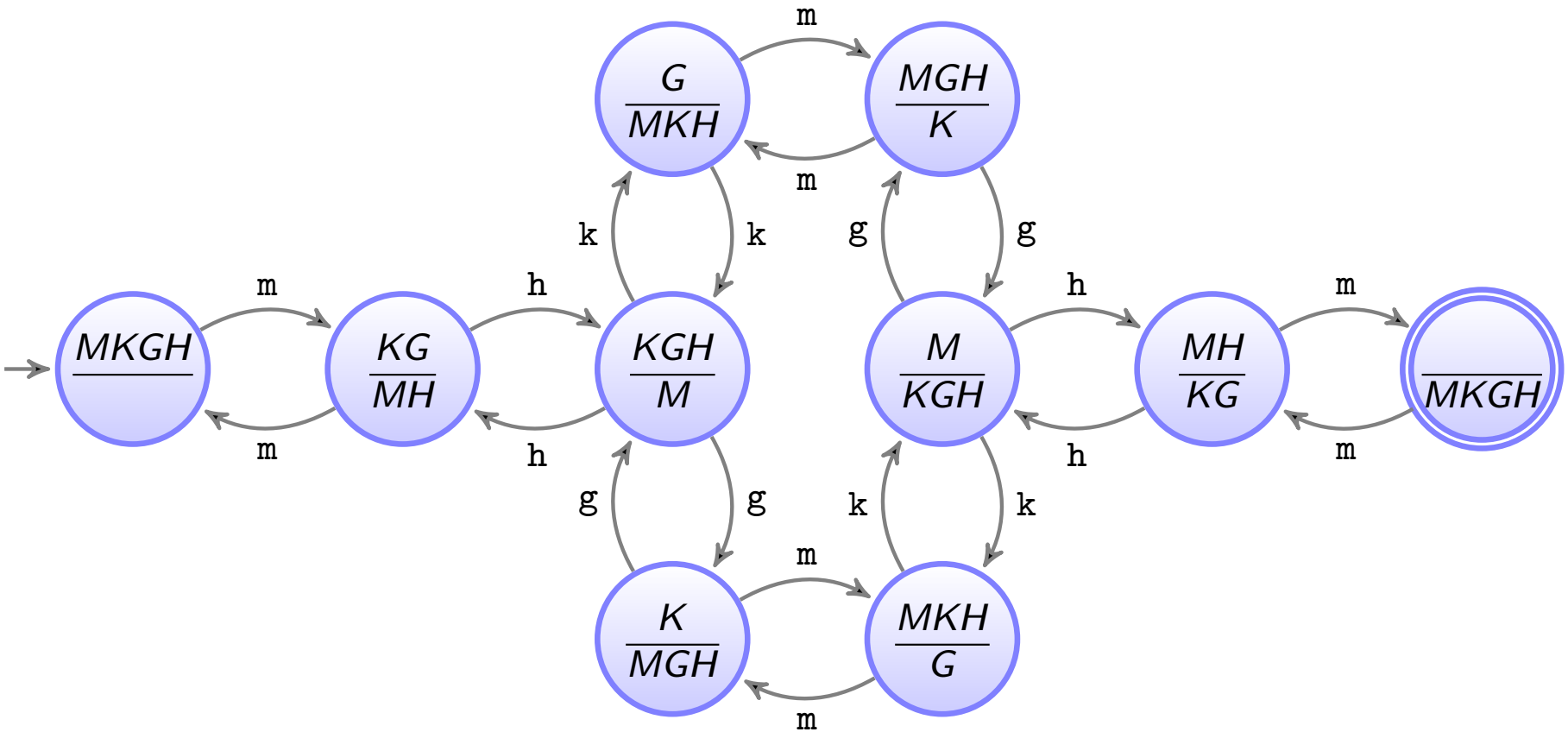
Endzustand (Ziel): $\frac{}{MKGH}$

Verbotene Zustände: $\frac{GH}{MK}$, $\frac{MK}{GH}$, $\frac{KH}{MG}$, $\frac{MG}{KH}$, $\frac{H}{MKG}$, $\frac{MKG}{H}$

Zustandsübergänge:

`h`, `m`, `k`, `g` ... Homer fährt alleine/mit Maggie/KR/Gift über den Fluss.

Automat (ohne verbotene Zustände und Übergänge):



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

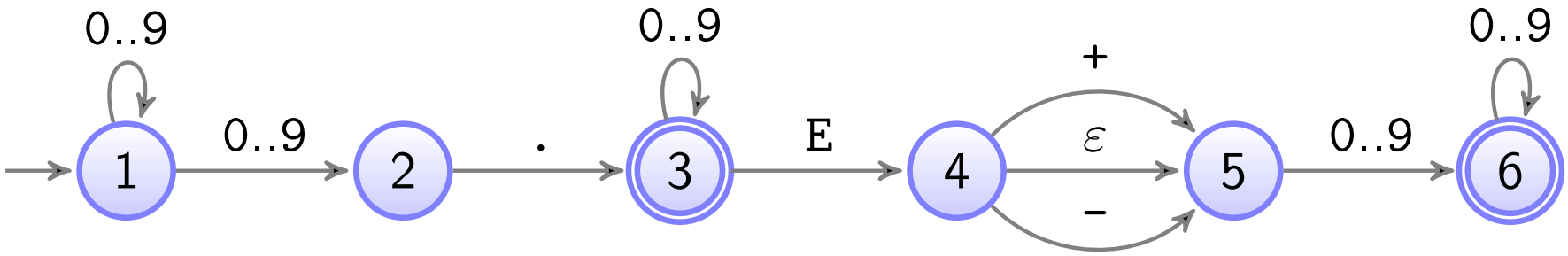
„Sprache des Automaten“

Beispiel: Reelle Numerale mit Exponentialteil

Z.B. `3.14`, `0.314E1` ($= 0.314 \cdot 10^1$), `314.E-2` ($= 314 \cdot 10^{-2}$)

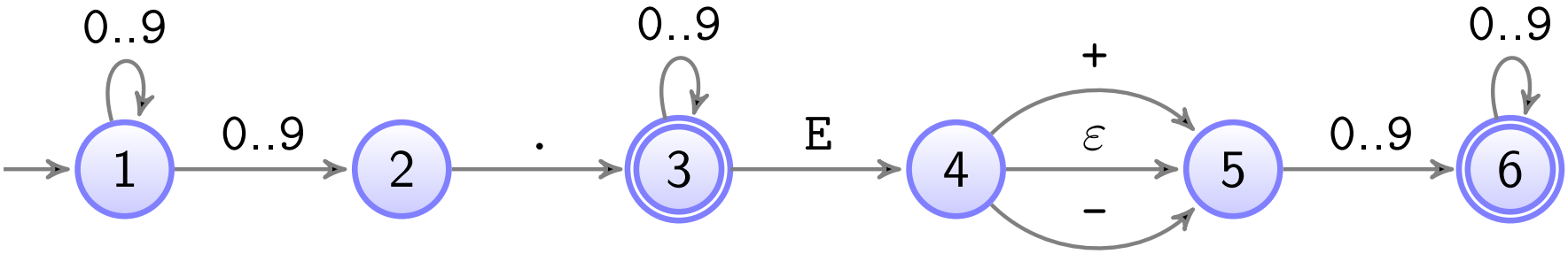
- Mindestens eine Ziffer vor Dezimalpunkt
- Dezimalpunkt
- Nachkommastellen optional
- Exponentialteil optional:
  - eingeleitet durch `E`
  - Vorzeichen optional
  - mindestens eine Ziffer

Endlicher Automat für die reellen Numerale



`0..9` ... Abkürzung für 10 parallele Übergänge beschriftet mit `0` bis `9`.

ε ... Leerwort, „Nichts“



- 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 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

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

Kennzeichen:

- endliche Menge von Zuständen
- Übergänge zwischen Zuständen
- Eingaben, die die Übergänge steuern.
- Ausgaben oder Aktionen, die in den Zuständen oder während der Übergänge getätigt werden.
- Anfangszustand
- Endzustände (optional)
- deterministisch: Der momentane Zustand und die nächste Eingabe bestimmen eindeutig den Folgezustand.
  nichtdeterministisch: Es gibt Zustände, die bei manchen Eingaben mehrere mögliche Folgezustände besitzen.

Arten endlicher Automaten

(Klassischer) Endlicher Automat:

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

Transducer: wie endlicher Automat, aber mit Ein- und Ausgaben.

Mealy-Automat: deterministischer Transducer; in der Regel keine Endzustände, verarbeitet daher unendliche Symbolfolgen.

Moore-Automat: wie Mealy-Automat, die Ausgaben sind aber mit den Zuständen verknüpft.

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

Weitere Typen: Verallgemeinerter endlicher Automat, Muller-Automat, Rabin-Automat, Baumautomaten, ...

Englische Begriffe: automaton/automata, finite state machine, DFA (Deterministic Finite Automaton), NFA (Non-Determinstic FA)

Weitere (nicht-endliche) Automatenarten: Kellerautomaten (Push-down automata), Turing-Maschinen, Registermaschinen etc. können Ausgaben wieder lesen $\Rightarrow$ zusätzlicher Speicher, mächtiger als endliche Automaten.

Spezifikation von Automaten:

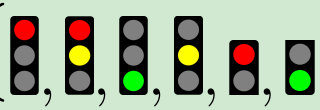
- Graphisch: Zustände sind Knoten, Übergänge sind Kanten, Ein- und Ausgaben sind Beschriftungen von Knoten und Kanten.
- Tabellarisch: Zu jedem Zustand und Eingabesymbol gibt es einen Eintrag mit zugehöriger Ausgabe und den Folgezuständen.

Was Sie 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

Formale Sprachen

Alphabet (Σ): endliche, nicht-leere Menge atomarer Symbole

- Menge aller lateinischen Buchstaben, Ziffern und Sonderzeichen
- Menge aller ägyptischen Hieroglyphen
- $\{$  $\}$
- $\{0, \ldots, 9, ., \mathtt{E}, +, -\}$
- $\{0, 1\}$
- $\{00, 01, 10, 11\}$

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

ε ... Leerwort

$\Sigma^+ = \{ s_1 \cdots s_n \mid s_i \in \Sigma, 1 \leq i \leq n \}$ ... Menge aller nicht-leeren Wörter über Σ

$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$ ... Menge aller Wörter über Σ (inklusive Leerwort)

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

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

D.h.: Für alle Wörter $u, v, w \in \Sigma^*$ gelten folgende Gleichungen:

$(u \cdot v) \cdot w = u \cdot (v \cdot w)$ Assoziativität

$w \cdot \varepsilon = \varepsilon \cdot w = w$ Neutralität

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

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

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

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

Formale Sprache über Σ: beliebige Teilmenge von Σ^*

- die Menge aller deutschen Sätze (Alphabet: Buchstaben+Satzzeichen)
- die Menge aller Java-Programme (Alphabet: ASCII-Zeichen)
- $\{\}$, $\{\varepsilon\}$, Σ^*

2^{Σ^*} ... Menge aller Sprachen über Σ = Menge aller Teilmengen von Σ^*

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

Deterministische endliche Automaten

Deterministischer endlicher Automat (DEA)

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

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet (*input alphabet*)
- $\delta\colon Q \times \Sigma \mapsto Q$ ... Übergangsfunktion (total) (*transition function*)
- $q_0 \in Q$ ... Anfangszustand (*initial state*)
- $F \subseteq Q$ ... Menge der Endzustände (*final states*)

δ ist eine totale Funktion: Folgezustand $\delta(q, s)$ ist für jeden Zustand $q \in Q$ und jede Eingabe $s \in \Sigma$ eindeutig definiert. $\Longrightarrow$ „deterministisch“

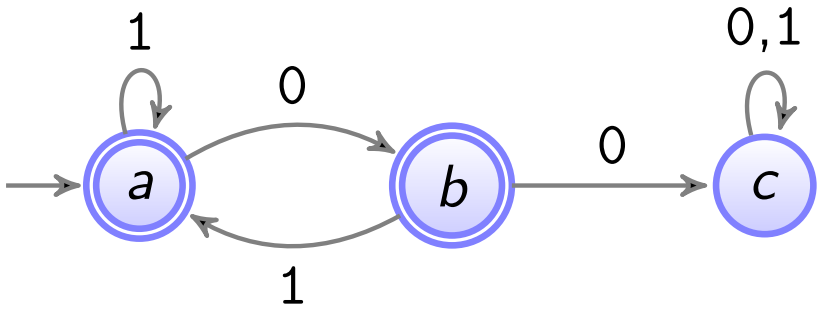
Erweiterte Übergangsfunktion $\delta^*\colon Q \times \Sigma^* \mapsto Q$

$\delta^*(q, \varepsilon) = q, \quad \delta^*(q, sw) = \delta^*(\delta(q, s), w)$ für alle $q \in Q$, $s \in \Sigma$, $w \in \Sigma^*$.

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{\, w \in \Sigma^* \mid \delta^*(q_0, w) \in F \,\}$

Beispiel: 00-freie Binärstrings



c ... „Falle“, Fehlerzustand
wird oft auch weggelassen

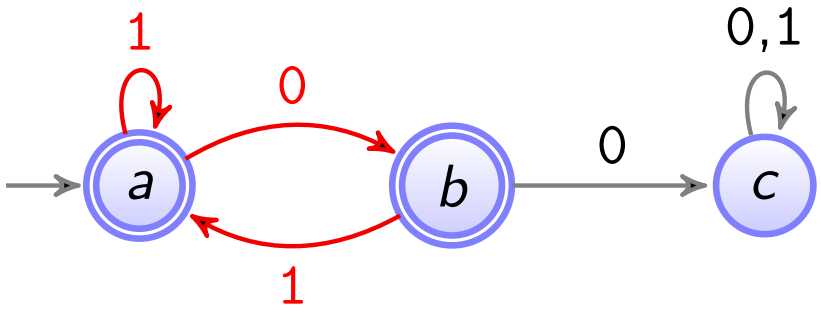
$\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, wobei

- $Q = \{a, b, c\}$ ... Zustandsmenge
- $\Sigma = \{0, 1\}$ ... Eingabealphabet
- $\delta\colon Q \times \Sigma \mapsto Q$ ... Übergangsfunktion definiert durch:

δ	0	1
a	b	a
b	c	a
c	c	c

- $q_0 = a$ ... Anfangszustand
- $F = \{a, b\}$ ... Endzustände

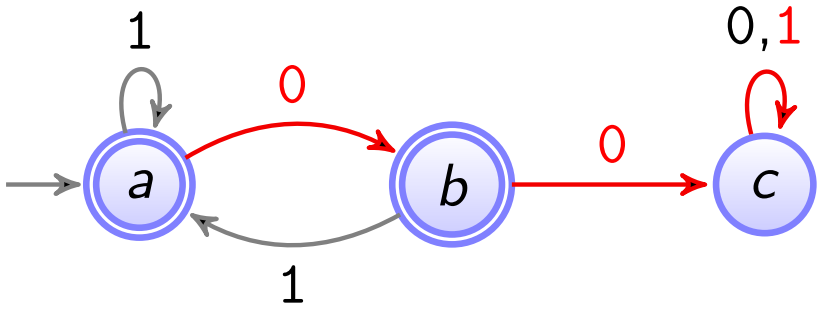
Beispiel: 00-freie Binärstrings



$$
\begin{aligned}
\delta^*(a, 101) &= \delta^*(\delta(a, 1), 01) \qquad & \delta^*(q, sw) &= \delta^*(\delta(q, s), w) \\
&= \delta^*(a, 01) \\
&= \delta^*(\delta(a, 0), 1) \\
&= \delta^*(b, 1) \\
&= \delta^*(\delta(b, 1), \varepsilon) \\
&= \delta^*(a, \varepsilon) \qquad & \delta^*(q, \varepsilon) &= q \\
&= a
\end{aligned}
$$

Das Wort 101 wird von $\mathcal{A}$ akzeptiert/generiert, d.h., $101 \in \mathcal{L}(\mathcal{A})$, weil $\delta^*(a, 101) = a$ ein Endzustand ist.

Beispiel: 00-freie Binärstrings



$$\begin{aligned}\delta^*(a, 001) &= \delta^*(\delta(a, 0), 01) \\ &= \delta^*(b, 01) \\ &= \delta^*(\delta(b, 0), 1) \\ &= \delta^*(c, 1) \\ &= \delta^*(\delta(c, 1), \varepsilon) \\ &= \delta^*(c, \varepsilon) \\ &= c\end{aligned}$$

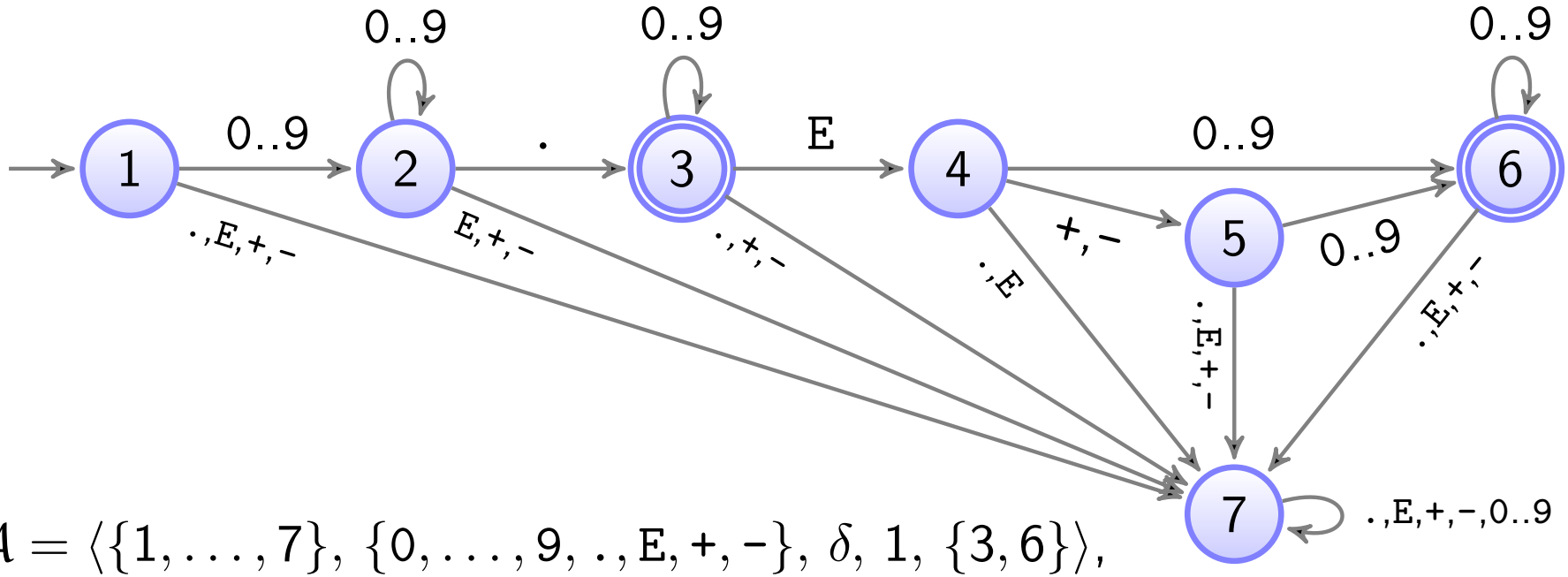
$$\delta^*(q, sw) = \delta^*(\delta(q, s), w)$$

$$\delta^*(q, \varepsilon) = q$$

Das Wort 001 wird von $\mathcal{A}$ nicht akzeptiert/generiert, $001 \notin \mathcal{L}(\mathcal{A})$, weil $\delta^*(a, 001) = c$ kein Endzustand ist.

$\mathcal{L}(\mathcal{A}) = \{ w \in \{0, 1\}^* \mid 00 \text{ kommt nicht in } w \text{ vor} \}$

Beispiel: reelle Numerale

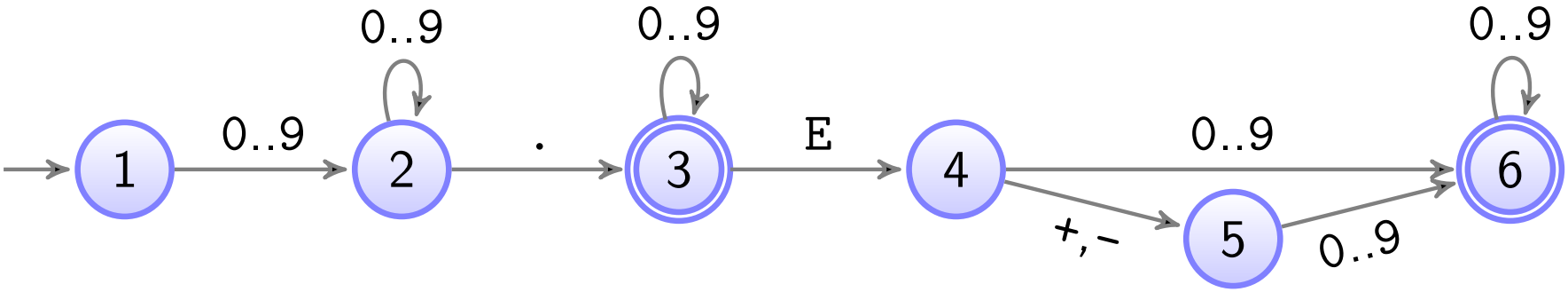


$\mathcal{A} = \langle \{1, \ldots, 7\},\ \{0, \ldots, 9, .\, , \mathtt{E}, +, -\},\ \delta,\ 1,\ \{3, 6\} \rangle,$

wobei

δ	$0 \cdots 9$	.	E	+	−
1	$2 \cdots 2$	7	7	7	7
2	$2 \cdots 2$	3	7	7	7
3	$3 \cdots 3$	7	4	7	7
4	$6 \cdots 6$	7	7	5	5
5	$6 \cdots 6$	7	7	7	7
6	$6 \cdots 6$	7	7	7	7
7	$7 \cdots 7$	7	7	7	7

Beispiel: reelle Numerale



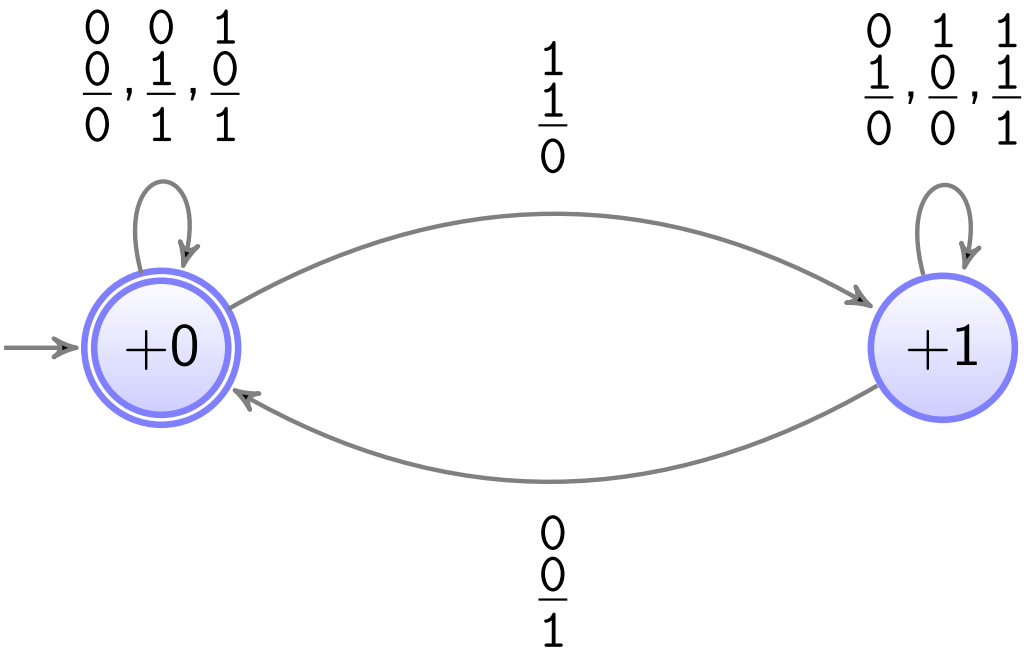
	δ	0 ··· 9	.	E	+	−
AZ	1	2 ··· 2	7	7	7	7
	2	2 ··· 2	3	7	7	7
EZ	3	3 ··· 3	7	4	7	7
	4	6 ··· 6	7	7	5	5
	5	6 ··· 6	7	7	7	7
EZ	6	6 ··· 6	7	7	7	7
	7	7 ··· 7	7	7	7	7

Beispiel: Binäraddition von rechts nach links (Kontrolle)

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

$\longleftarrow$

$Q = \{+0, +1\}$ $\quad$ $\Sigma = \{\frac{0}{0}, \frac{0}{1}, \frac{1}{0}, \frac{1}{1}, \frac{0}{0}, \frac{0}{1}, \frac{1}{0}, \frac{1}{1}\}$



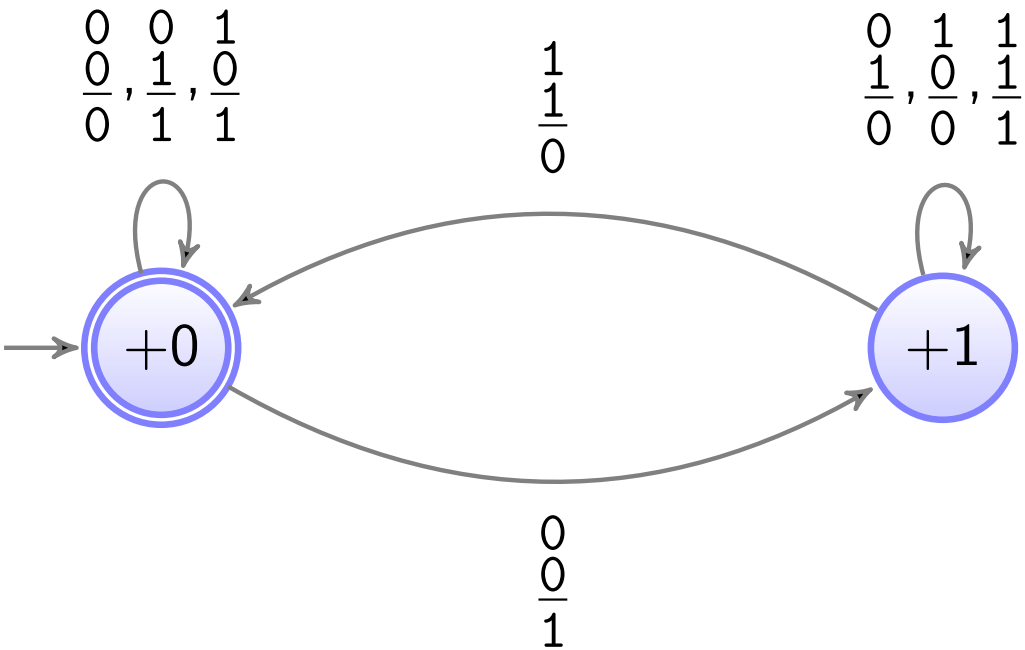
+0 . . . kein Übertrag
+1 . . . Übertrag

Beispiel: Binäraddition von links nach rechts (Kontrolle)

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

$\longrightarrow$

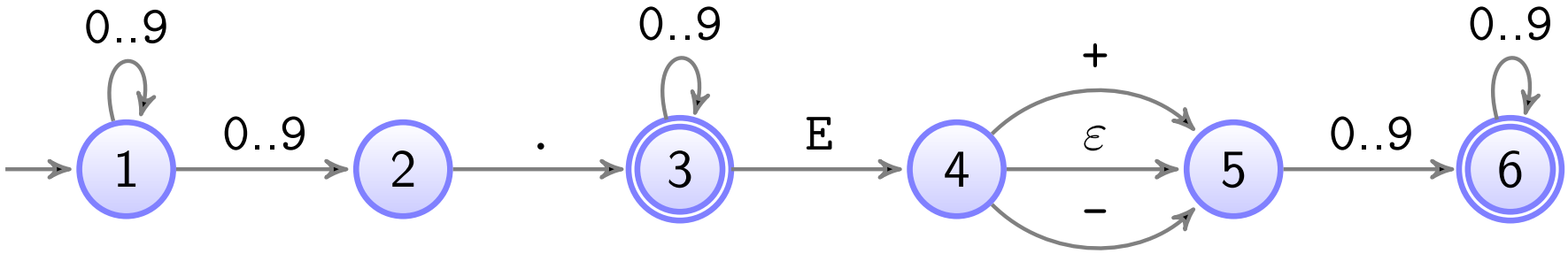
$Q = \{+0, +1\}$ $\quad$ $\Sigma = \{\frac{0}{0}, \frac{0}{1}, \frac{1}{0}, \frac{1}{1}, \frac{0}{0}, \frac{0}{1}, \frac{1}{0}, \frac{1}{1}\}$



+0 . . . kein Übertrag
+1 . . . Übertrag

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



Kein deterministischer Automat!

- $\delta(1, 0) = 1$?
  $\delta(1, 0) = 2$?
  Die Übergangsfunktion muss ein eindeutiges Ergebnis besitzen.
- $\delta(4, \varepsilon) = 5$?
  Die Übergangsfunktion ist vom Typ $Q \times \Sigma \mapsto Q$, aber $\varepsilon \notin \Sigma$!

Indeterminismus: Der momentane Zustand und das Eingabesymbol legen den nächsten Zustand bzw. die nächste Aktion nicht eindeutig fest.

- In Zustand 1 sind bei Eingabe `0` die Folgezustände 1 und 2 möglich.
- In Zustand 3 sind bei Eingabe `E` die Folgezustände 4 und 5 möglich.
- In Zustand 4 ist Zustand 5 mit und ohne Eingabe erreichbar.

Ob die richtige Entscheidung getroffen wurde, wird erst später klar.

Nichtdeterministische endliche Automaten

Nichtdeterministischer endlicher Automat (NEA)

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

- Q, Σ, q_0, F ... siehe DEAs
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ ... Übergangsrelation

DEA: $\delta\colon Q \times \Sigma \mapsto Q$ ... totale Übergangsfunktion

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

δ^* ist die kleinste Menge mit folgenden Eigenschaften:

- $(q, \varepsilon, q) \in \delta^*$ für alle $q \in Q$
- Wenn $(q_1, w, q_2) \in \delta^*$ und $(q_2, s, q_3) \in \delta$, dann $(q_1, ws, q_3) \in \delta^*$.

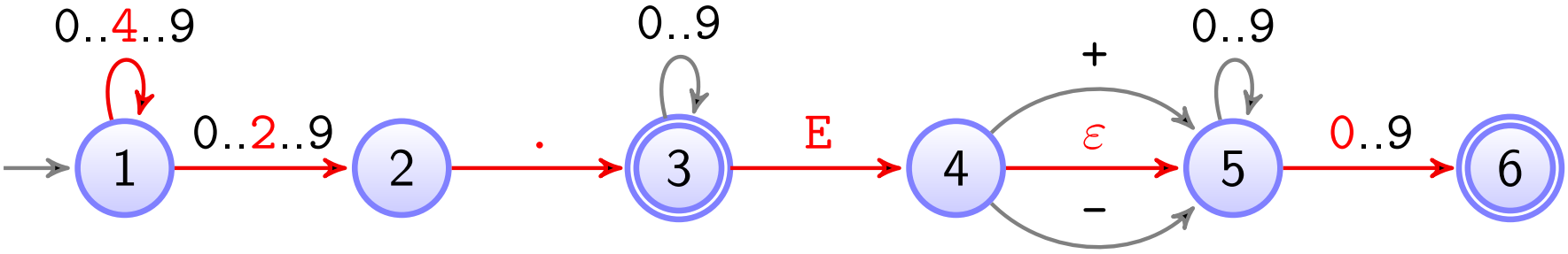
DEA: $\delta^*(q, \varepsilon) = q, \quad \delta^*(q, sw) = \delta^*(\delta(q, s), w)$

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{\, w \in \Sigma^* \mid (q_0, w, q_f) \in \delta^* \text{ für ein } q_f \in F \,\}$

DEA: $\mathcal{L}(\mathcal{A}) = \{\, w \in \Sigma^* \mid \delta^*(q_0, w) \in F \,\}$

Beispiel: 42.E0 ist ein reelles Numeral



$\mathcal{L}(\mathcal{A}) = \{\, w \in \Sigma^* \mid (1, w, 3) \in \delta^* \text{ oder } (1, w, 6) \in \delta^* \,\}$

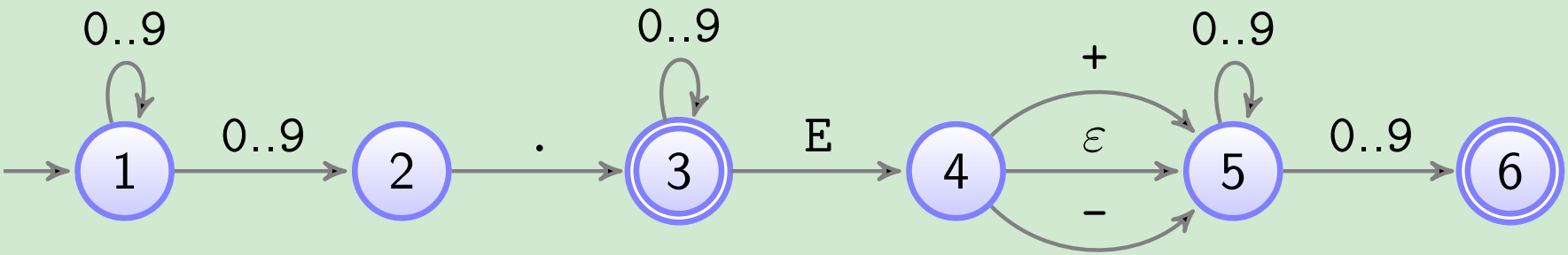
Zu zeigen: $\texttt{42.E0} \in \mathcal{L}(\mathcal{A})$

Wenn $(q_1, w, q_2) \in \delta^*$ und $(q_2, s, q_3) \in \delta$, dann $(q_1, ws, q_3) \in \delta^*$.		
$(1, \varepsilon, 1)$	$(1, 4, 1)$	$(1, 4, 1)$
$(1, 4, 1)$	$(1, 2, 2)$	$(1, 42, 2)$
$(1, 42, 2)$	$(2, ., 3)$	$(1, 42., 3)$
$(1, 42., 3)$	$(3, \texttt{E}, 4)$	$(1, 42.\texttt{E}, 4)$
$(1, 42.\texttt{E}, 4)$	$(4, \varepsilon, 5)$	$(1, 42.\texttt{E}, 5)$
$(1, 42.\texttt{E}, 5)$	$(5, 0, 6)$	$(1, 42.\texttt{E0}, 6)$

$(1, 42.\texttt{E0}, 6) \in \delta^*$, 1 ist Anfangs- und 6 Endzustand $\Longrightarrow \texttt{42.E0} \in \mathcal{L}(\mathcal{A})$

Tabellarische Darstellung der Übergangsrelation $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

Für jeden Zustand $q \in Q$ und jede Eingabe $s \in \Sigma \cup \{\varepsilon\}$:
Tabelleneintrag mit der Menge $\{\, q' \in Q \mid (q, s, q') \in \delta \,\}$ der Folgezustände



	δ	0	$\cdots$	9	.	E	+	–	ε
AZ	1	$\{1,2\}$	$\cdots$	$\{1,2\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
	2	$\{\}$	$\cdots$	$\{\}$	$\{3\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
EZ	3	$\{3\}$	$\cdots$	$\{3\}$	$\{\}$	$\{4\}$	$\{\}$	$\{\}$	$\{\}$
	4	$\{\}$	$\cdots$	$\{\}$	$\{\}$	$\{\}$	$\{5\}$	$\{5\}$	$\{5\}$
	5	$\{5,6\}$	$\cdots$	$\{5,6\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
EZ	6	$\{\}$	$\cdots$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$

Alternative Definition von NEAs:
Übergangsfunktion $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$ (ist total!) an Stelle von
Übergangsrelation $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

Vergleich DEA – NEA

NEAs sind flexibler:

- Mehrere Folgezustände pro Zustand und Eingabe möglich;
- Kein Folgezustand zu einem Zustand und einer Eingabe erlaubt;
- Zustandswechsel ohne Eingabe möglich (ε-Übergang).

DEAs und NEAs besitzen dieselbe Ausdrucksstärke.

- Jeder DEA ist per Definition auch ein NEA.
- Zu jedem NEA gibt es einen DEA, der dieselbe Sprache akzeptiert.
  (Lässt sich automatisch finden, siehe später.)

Vorteile von NEAs:

- Benötigen teilweise erheblich weniger Zustände und Übergänge als äquivalente DEAs.
  Die Zustandszahl im DEA kann exponentiell größer sein als im NEA.
- Bei Modellierungsaufgaben leichter zu konstruieren.

Vorteile von DEAs:

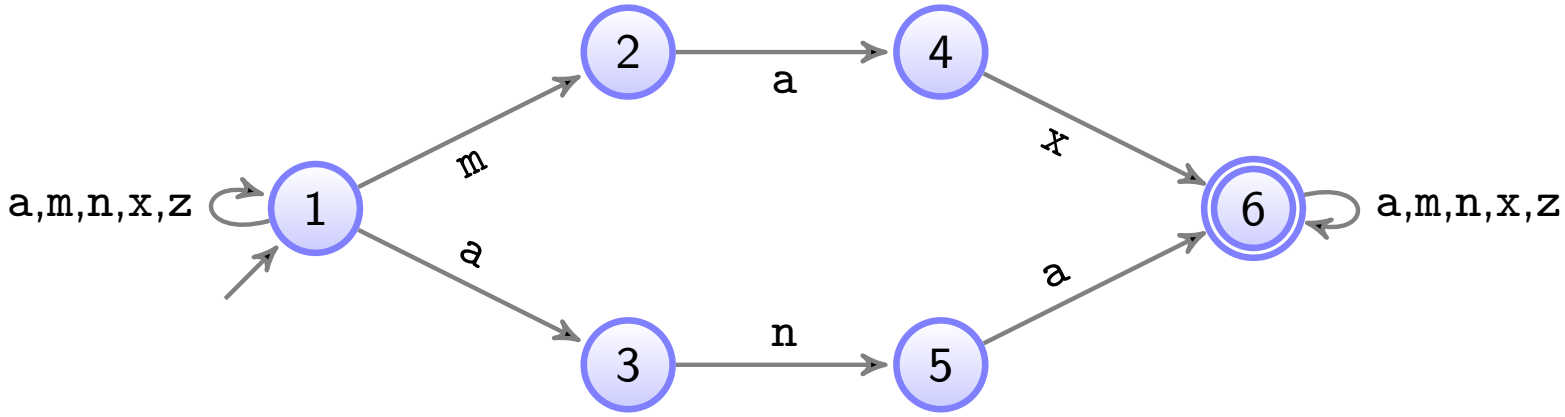
- Effiziente Abarbeitung, kein Backtracking.

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: