

FORMALE SYSTEME

3. Vorlesung: Endliche Automaten

Hannes Straß

Folien: © Markus Krötzsch, <https://iccl.inf.tu-dresden.de/web/FS2020>, CC BY 3.0 DE

TU Dresden, 18. Oktober 2021

Organisatorisches

Übungen

Übungsbeginn: Montag, der 18. Oktober 2021

- Termine: Mo 2., Fr 3., Do 1., Do 5., Fr 4.

Übungsbeginn: Montag, der 1. November 2021

- Termine: Mi 6., Fr 6., Mo 2., Fr 2., Mo 6.

Vorübergehender Übungsbetrieb: 18.–29. Oktober 2021

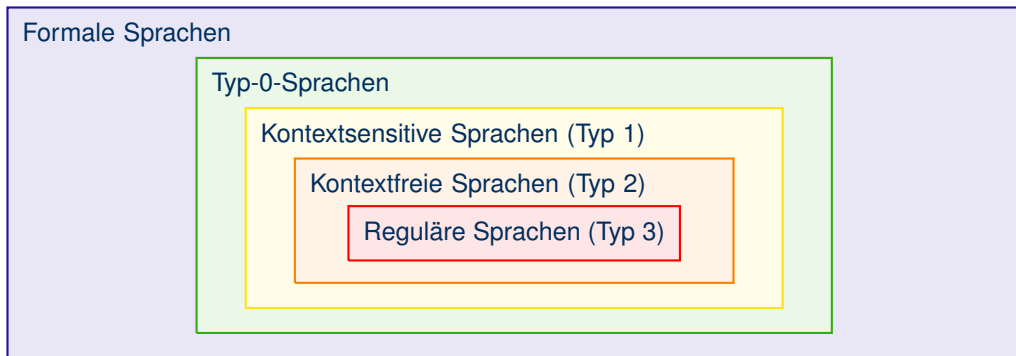
Online-Übungen für alle, deren Übung erst am 1. November beginnt:

- Montags 4. DS (Thomas Feller)
- Mittwochs 6. DS (Dörthe Arndt)
- Donnerstags 5. DS (Hannes Straß)
- Freitags 2. DS (Jonas Karge)

Rückblick

Wiederholung

Mit Grammatiken können wir Sprachen beschreiben und sie grob in Typen unterteilen:



Aufzeichnung startet . . .

Grammatiken in der Praxis

Grammatiken in der Praxis

Eine „ASCII-Syntax“ für Typ-2-Grammatiken ist die sogenannte **Backus-Naur-Form** (BNF):¹

- statt \rightarrow wird $::=$ verwendet
- | für Alternativen (wie bei uns)
- Markierung: "Terminalsymbole" und \langle Nichtterminalsymbole \rangle



oben: John Backus, Peter Naur
unten: Niklaus Wirth, Pāṇini

¹Pāṇini, 5./4. Jhd. v.u.Z.; neu erfunden von John Backus im Jahr 1959.

Grammatiken in der Praxis

Eine „ASCII-Syntax“ für Typ-2-Grammatiken ist die sogenannte **Backus-Naur-Form** (BNF):¹

- statt \rightarrow wird $::=$ verwendet
- | für Alternativen (wie bei uns)
- Markierung: "Terminalsymbole" und \langle Nichtterminalsymbole \rangle

Erweiterte BNF (EBNF, von Niklaus Wirth):

- allgemein wie BNF, aber Konkatenation mit Kommas
- Zusätzliche Abkürzungen $[X]$ für „null oder ein X“ und $\{X\}$ für „null oder mehr X“

In der Praxis sind unterschiedliche Notationen im Einsatz

(EBNF von Wirth; ISO EBNF; W3C EBNF; „Augmented BNF“ der IETF; verschiedene „EBNF“ aus Vorlesungen und Lehrbüchern ...)



oben: John Backus, Peter Naur
unten: Niklaus Wirth, Pāṇini

¹Pāṇini, 5./4. Jhd. v.u.Z.; neu erfunden von John Backus im Jahr 1959.

Sprachen Nutzen und Verstehen

Von Beschreiben zu Erkennen

Grammatiken beschreiben Sprachen als Mengen von Wörtern, die sie erzeugen können.

Praktisch wichtige Aufgabe: Erkennen, ob ein Wort in einer Sprache liegt.

Das **Wortproblem** für eine Sprache L über Alphabet Σ besteht darin, die folgende Funktion zu berechnen:

Eingabe: ein Wort $w \in \Sigma^*$

Ausgabe: „ja“ wenn $w \in L$ und „nein“ wenn $w \notin L$

Von Beschreiben zu Erkennen

Grammatiken beschreiben Sprachen als Mengen von Wörtern, die sie erzeugen können.

Praktisch wichtige Aufgabe: Erkennen, ob ein Wort in einer Sprache liegt.

Das **Wortproblem** für eine Sprache L über Alphabet Σ besteht darin, die folgende Funktion zu berechnen:

Eingabe: ein Wort $w \in \Sigma^*$

Ausgabe: „ja“ wenn $w \in L$ und „nein“ wenn $w \notin L$

Für die Sprache $\{a\} \circ \{b\}^*$ wird das Wortproblem durch einen einfachen Algorithmus in linearer Zeit gelöst (teste, ob der erste Buchstabe a ist und ob alle folgenden Buchstaben b sind).

Andererseits kann das Wortproblem sehr schwer sein, selbst für Sprachen, die mit einer Grammatik formal beschrieben sind.

Ausblick

Wir werden sehen, dass jede Sprachklasse zu einem bestimmten Berechnungsmodell passt:

Sprachklasse	Berechnungsmodell	Wortproblem
Typ 0	Turingmaschine (TM)	semi-entscheidbar ¹
Kontextsensitiv	nichtdeterministische TM mit linearem Speicher	PSpace-vollständig ²
Kontextfrei	nichtdeterministischer Kellerautomat	polynomiell
Regulär	endlicher Automat	polynomiell ³

¹Akzeptanz von Wörtern kann beliebig lange dauern, so dass man nie weiß, ob sie noch passiert; Wörter können nur rekursiv aufgezählt werden.

²Höchstwahrscheinlich schwerer als NP; jeder praktisch bekannte Algorithmus benötigt im schlimmsten Fall exponentiell viel Zeit.

³Tatsächlich sogar noch viel einfacher als Typ 2.

Weitere Fragestellungen

Das Wortproblem ist nicht die einzige praktisch relevante Frage.

Darstellungen von Sprachen

- Welche verschiedenen Darstellungen gibt es (Grammatiken, Automaten, ...)? Wie kann man eine Darstellung in eine andere übersetzen?
- Beschreiben zwei Darstellungen die gleiche Sprache? Beschreibt eine Darstellung die leere Sprache?
- Wie kann eine Darstellung vereinfacht werden?

Eigenschaften von Sprachen

- Was passiert, wenn man Operationen anwendet, um neue Sprachen zu erzeugen?

Beispiel:

Wenn L_1 und L_2 regulär sind, wie ist es dann mit $L_1 \cap L_2$?

→ sogenannte **Abschlusseigenschaften**

Vorschau

Plan der nächsten Vorlesungen:

Reguläre Sprachen

- endliche Automaten
- reguläre Ausdrücke
- Eigenschaften regulärer Sprachen

Kontextfreie Sprachen

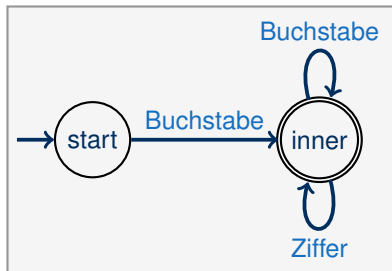
- Normalformen kontextfreier Grammatiken
- Kellerautomaten
- Eigenschaften kontextfreier Sprachen




Endliche Automaten

Beispiel

„Ein Bezeichner ist ein String, der mit einem Buchstaben beginnt und danach nur Buchstaben oder Ziffern enthält.“

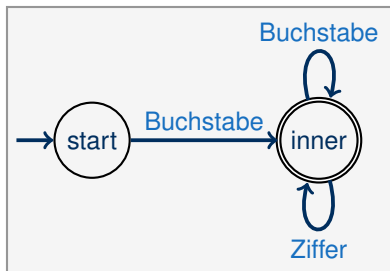
Darstellung als endlicher Automat:



- Automat beginnt im Startzustand → 
- Zustandswechsel gemäß Pfeilen
- Kein passender Pfeil für Symbol?
„**return false**“
- Keine weiteren Symbole?
„**return true**“ in Endzustand 
„**return false**“ falls in Zustand 

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort **Zustandsfolge**

utf8

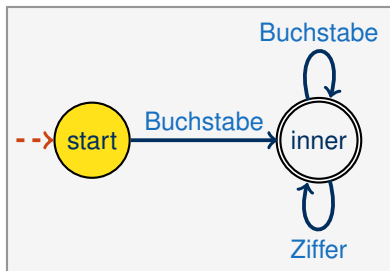
C++

€

Ergebnis

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort **Zustandsfolge**

ut f 8 start

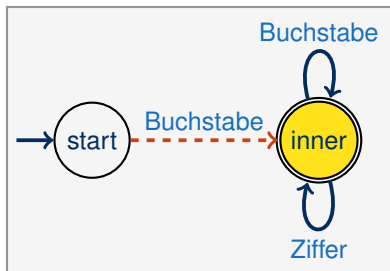
C++

€

Ergebnis

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort **Zustandsfolge**

u t f 8 start inner

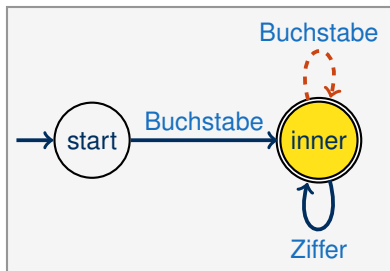
C++

€

Ergebnis

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge
------	---------------

ut.f8	start inner inner
-------	-------------------

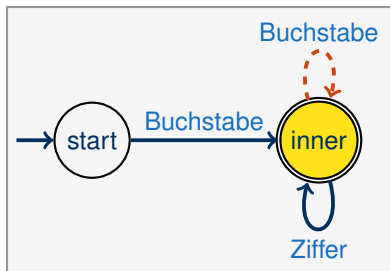
C++	
-----	--

€	
---	--

Ergebnis

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

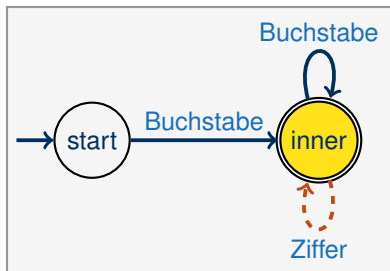
ut <u>f</u> 8	start inner inner inner	
---------------	-------------------------	--

C++		
-----	--	--

€		
---	--	--

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

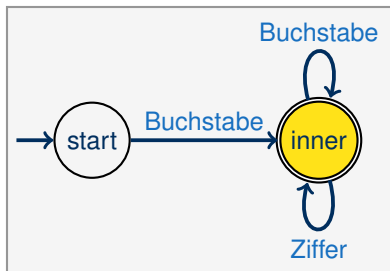
ut.f8	start inner inner inner inner	
-------	-------------------------------	--

C++		
-----	--	--

€		
---	--	--

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

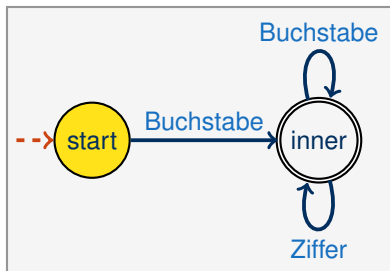
ut.f8	start inner inner inner inner	akzeptiert
-------	-------------------------------	------------

C++

€

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

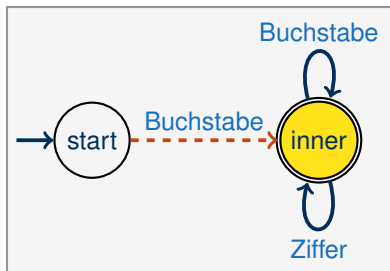
ut.f8	start inner inner inner inner	akzeptiert
-------	-------------------------------	------------

C++	start	
-----	-------	--

€

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

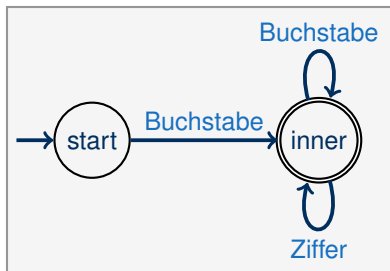
ut.f8	start inner inner inner inner	akzeptiert
-------	-------------------------------	------------

C++	start inner	
-----	-------------	--

€

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

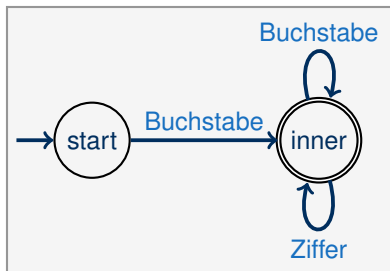
ut.f8	start inner inner inner inner	akzeptiert
-------	-------------------------------	------------

C++	start inner ?	
-----	---------------	--

€

Worterkennung mit Automaten

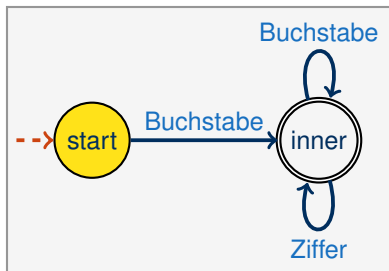
Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
ut.f8	start inner inner inner inner	akzeptiert
C++	start inner ?	abgelehnt (fehlender Übergang)
€		

Worterkennung mit Automaten

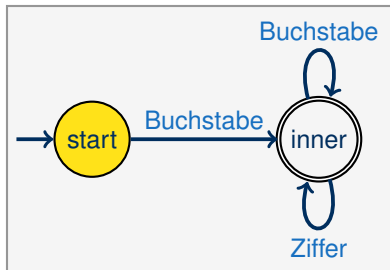
Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
ut.f8	start inner inner inner inner	akzeptiert
C++	start inner ?	abgelehnt (fehlender Übergang)
€	start	

Worterkennung mit Automaten

Ein Automat kann eine Sprache erkennen, indem er Wörter akzeptiert oder ablehnt:



Wort	Zustandsfolge	Ergebnis
ut.f8	start inner inner inner inner	akzeptiert
C++	start inner ?	abgelehnt (fehlender Übergang)
€	start	abgelehnt (kein Endzustand)

Deterministische Endliche Automaten

Die graphische Darstellung von Automaten ist anschaulich, aber nicht immer praktisch. Formal definieren wir Automaten wie folgt:

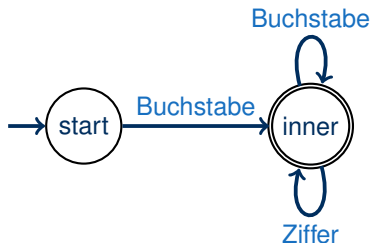
Ein **deterministischer endlicher Automat** (international: „DFA“, deterministic finite automaton) \mathcal{M} ist ein Tupel $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ mit den folgenden Bestandteilen:

- Q : endliche Menge von **Zuständen**
- Σ : Alphabet
- δ : **Übergangsfunktion**, eine **partielle*** Funktion $Q \times \Sigma \rightarrow Q$
- q_0 : **Startzustand** $q_0 \in Q$
- F : Menge von **Endzuständen** $F \subseteq Q$

(* D.h. manche Übergänge sind undefiniert, wie im Beispiel.)

Notation: Wir schreiben statt $\delta(q, \mathbf{a}) = q'$ auch $q \xrightarrow{\mathbf{a}} q'$.

Beispiel



Endlicher Automat $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ mit

- $Q = \{\text{start}, \text{inner}\}$
- $\Sigma = \{\mathbf{a}, \mathbf{b}, \dots, \mathbf{0}, \mathbf{1}, \dots, \mathbf{9}, \dots\}$ (in Zeichnung unterspezifiziert)
- δ definiert wie folgt:
 - $\delta(\text{start}, x) = \text{inner}$ für alle Buchstaben x
 - $\delta(\text{inner}, y) = \text{inner}$ für alle Buchstaben und Ziffern y
 - $\delta(q, x)$ undefiniert für alle anderen Fälle
- $q_0 = \text{start}$
- $F = \{\text{inner}\}$

Ist mein Computer ein DFA?

Beobachtungen:

- Computer haben endlich viel Speicher, können also nur endlich viele Zustände einnehmen
- Programme verändern Speicher nach festen Regeln, die man als Übergangsfunktion auffassen könnte

Sind alle Computer DFAs?

Ist mein Computer ein DFA?

Beobachtungen:

- Computer haben endlich viel Speicher, können also nur endlich viele Zustände einnehmen
- Programme verändern Speicher nach festen Regeln, die man als Übergangsfunktion auffassen könnte

Sind alle Computer DFAs?

Jein.

- Zustandsmenge und Übergangsfunktion extrem groß
~> keine praktisch nützliche Beschreibung für ganze Computer
- Computerprogramme verhalten sich systematisch, unabhängig von der Speichergröße
~> DFA-Übergänge können diese Systematik nicht gut abbilden

Die Sprache eines DFA

Für einen DFA $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ erweitern wir $\delta : Q \times \Sigma \rightarrow Q$ zu einer Übergangsfunktion $\delta^* : Q \times \Sigma^* \rightarrow Q$ auf Wörtern:

Für einen Zustand $q \in Q$ und ein Wort $w \in \Sigma^*$ sei $\delta^*(q, w)$ der eindeutig bestimmte Zustand, den man erreicht, wenn man ausgehend von q das Wort w einliest:

- $\delta^*(q, \epsilon) = q$ (Fall $w = \epsilon$)
- $\delta^*(q, av) = \delta^*(\delta(q, a), v)$ (Fall $w = av$)

Der Wert $\delta^*(q, w)$ ist undefiniert, wenn einer der nötigen Übergänge undefiniert ist.

Die Sprache eines DFA kann nun formal definiert werden:

Die **Sprache eines DFA** $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ist die Menge

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

Hinweis: Wir schreiben zukünftig auch δ für δ^* .

Quiz: Sprache eines DFA

Die **Sprache eines DFA** $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ist $\mathbf{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$.

Dabei ist die erweiterte Übergangsfunktion definiert durch:

- $\delta^*(q, \epsilon) = q$ (Fall $w = \epsilon$)
- $\delta^*(q, \mathbf{a}v) = \delta^*(\delta(q, \mathbf{a}), v)$ (Fall $w = \mathbf{a}v$)

Quiz: Wir betrachten den DFA $\mathcal{M} = \langle Q, \{\mathbf{0}, \mathbf{1}\}, \delta, q_0, F \rangle$ mit den Übergängen:

...

Alternative Definition: Totale Übergänge

Wir können DFAs mit partieller Übergangsfunktion in DFAs mit totaler Übergangsfunktion umwandeln:

Eingabe: DFA $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$

Ausgabe: DFA $\mathcal{M}_{\text{total}} = \langle Q', \Sigma, \delta', q_0, F \rangle$

- füge einen neuen **Fangzustand** q_f ein: $Q' := Q \cup \{q_f\}$
- für alle Zustände $q \in Q'$ und Symbole $a \in \Sigma$:

$$\delta'(q, a) := \begin{cases} \delta(q, a) & \text{falls } \delta(q, a) \text{ definiert} \\ q_f & \text{falls } \delta(q, a) \text{ nicht definiert} \end{cases}$$

Anmerkung: Laut dieser Definition gilt insbesondere $\delta'(q_f, a) = q_f$ für alle $a \in \Sigma$.

↪ Wird beim Lesen eines Wortes q_f erreicht, dann kann es nicht mehr akzeptiert werden (da $q_f \notin F$).

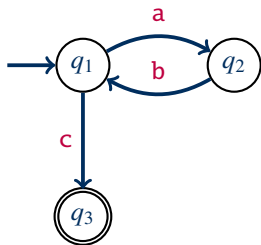
Der Fangzustand q_f wird teilweise auch als „Papierkorbzustand“ bezeichnet.

Totale Übergänge (2)

Satz: $\mathcal{M}_{\text{total}} = \langle Q', \Sigma, \delta', q_0, F \rangle$ hat eine totale Übergangsfunktion und akzeptiert die selbe Sprache wie \mathcal{M} , d.h. $\mathbf{L}(\mathcal{M}_{\text{total}}) = \mathbf{L}(\mathcal{M})$.

Beispiel: wir betrachten das Alphabet $\Sigma = \{a, b, c\}$

DFA \mathcal{M} mit partieller Übergangsfunktion:

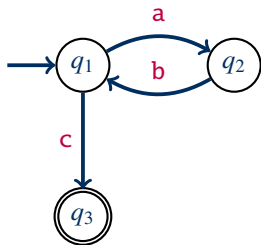


Totale Übergänge (2)

Satz: $\mathcal{M}_{\text{total}} = \langle Q', \Sigma, \delta', q_0, F \rangle$ hat eine totale Übergangsfunktion und akzeptiert die selbe Sprache wie \mathcal{M} , d.h. $\mathbf{L}(\mathcal{M}_{\text{total}}) = \mathbf{L}(\mathcal{M})$.

Beispiel: wir betrachten das Alphabet $\Sigma = \{a, b, c\}$

DFA \mathcal{M} mit partieller
Übergangsfunktion:



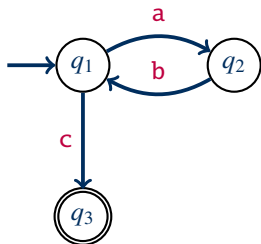
$$\mathbf{L}(\mathcal{M}) = (ab)^*c$$

Totale Übergänge (2)

Satz: $\mathcal{M}_{\text{total}} = \langle Q', \Sigma, \delta', q_0, F \rangle$ hat eine totale Übergangsfunktion und akzeptiert die selbe Sprache wie \mathcal{M} , d.h. $\mathbf{L}(\mathcal{M}_{\text{total}}) = \mathbf{L}(\mathcal{M})$.

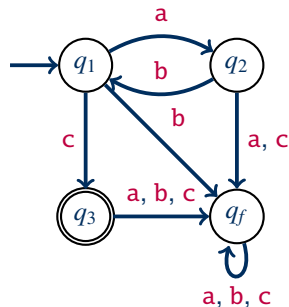
Beispiel: wir betrachten das Alphabet $\Sigma = \{a, b, c\}$

DFA \mathcal{M} mit partieller Übergangsfunktion:



$$\mathbf{L}(\mathcal{M}) = (ab)^*c$$

DFA $\mathcal{M}_{\text{total}}$ mit totaler Übergangsfunktion:



Die Sprachen endlicher Automaten

Idee: Ein Typ von Automaten definiert eine Klasse von Sprachen

Bsp.: „die Klasse aller Sprachen, die irgendein DFA erkennen kann“

Eine Alternative zur Chomsky-Hierarchie?

Die Sprachen endlicher Automaten

Idee: Ein Typ von Automaten definiert eine Klasse von Sprachen

Bsp.: „die Klasse aller Sprachen, die irgendein DFA erkennen kann“

Eine Alternative zur Chomsky-Hierarchie?

Es zeigt sich: Mit DFAs erhält man keine neue Sprachklasse!

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Konsequenzen:

- DFAs können nur sehr einfache Sprachen akzeptieren.
- Typ-3-Sprachen sind eine „natürliche“ Klasse: Sowohl in der „Welt“ der Grammatiken als auch in der „Welt“ der Automaten leicht zu definieren.

Von DFAs zu regulären Grammatiken

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Eine Richtung dieser Behauptung ist leicht zu sehen:

Für jeden DFA \mathcal{M} gibt es eine reguläre Grammatik $G_{\mathcal{M}}$, welche die selbe Sprache erzeugt (d.h., $\mathbf{L}(\mathcal{M}) = \mathbf{L}(G_{\mathcal{M}})$).

Von DFAs zu regulären Grammatiken

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Eine Richtung dieser Behauptung ist leicht zu sehen:

Für jeden DFA \mathcal{M} gibt es eine reguläre Grammatik $G_{\mathcal{M}}$, welche die selbe Sprache erzeugt (d.h., $\mathbf{L}(\mathcal{M}) = \mathbf{L}(G_{\mathcal{M}})$).

Für $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_{\mathcal{M}} = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$

Von DFAs zu regulären Grammatiken

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Eine Richtung dieser Behauptung ist leicht zu sehen:

Für jeden DFA \mathcal{M} gibt es eine reguläre Grammatik $G_{\mathcal{M}}$, welche die selbe Sprache erzeugt (d.h., $\mathbf{L}(\mathcal{M}) = \mathbf{L}(G_{\mathcal{M}})$).

Für $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_{\mathcal{M}} = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$
- $S := q_0$

Von DFAs zu regulären Grammatiken

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Eine Richtung dieser Behauptung ist leicht zu sehen:

Für jeden DFA \mathcal{M} gibt es eine reguläre Grammatik $G_{\mathcal{M}}$, welche die selbe Sprache erzeugt (d.h., $\mathbf{L}(\mathcal{M}) = \mathbf{L}(G_{\mathcal{M}})$).

Für $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_{\mathcal{M}} = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$
- $S := q_0$
- P besteht aus den folgenden Produktionsregeln:

$$q \rightarrow \mathbf{a}q' \quad \text{falls } \delta(q, \mathbf{a}) = q'$$

Von DFAs zu regulären Grammatiken

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Eine Richtung dieser Behauptung ist leicht zu sehen:

Für jeden DFA \mathcal{M} gibt es eine reguläre Grammatik $G_{\mathcal{M}}$, welche die selbe Sprache erzeugt (d.h., $\mathbf{L}(\mathcal{M}) = \mathbf{L}(G_{\mathcal{M}})$).

Für $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_{\mathcal{M}} = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$
- $S := q_0$
- P besteht aus den folgenden Produktionsregeln:

$$q \rightarrow \mathbf{a}q' \quad \text{falls } \delta(q, \mathbf{a}) = q'$$

$$q \rightarrow \mathbf{a} \quad \text{falls } \delta(q, \mathbf{a}) \in F$$

Von DFAs zu regulären Grammatiken

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Eine Richtung dieser Behauptung ist leicht zu sehen:

Für jeden DFA \mathcal{M} gibt es eine reguläre Grammatik $G_{\mathcal{M}}$, welche die selbe Sprache erzeugt (d.h., $\mathbf{L}(\mathcal{M}) = \mathbf{L}(G_{\mathcal{M}})$).

Für $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_{\mathcal{M}} = \langle V, \Sigma, P, S \rangle$ wie folgt:

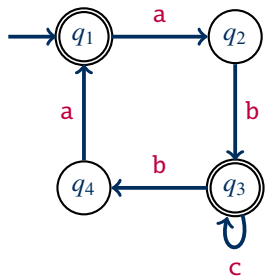
- $V := Q$
- $S := q_0$
- P besteht aus den folgenden Produktionsregeln:

$$q \rightarrow \mathbf{a}q' \quad \text{falls } \delta(q, \mathbf{a}) = q'$$

$$q \rightarrow \mathbf{a} \quad \text{falls } \delta(q, \mathbf{a}) \in F$$

$$q_0 \rightarrow \epsilon \quad \text{falls } q_0 \in F$$

Beispiel

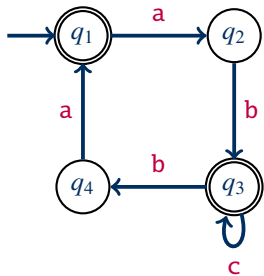


Für $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_M = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$
- $S := q_0$
- P besteht aus den folgenden Produktionsregeln:

Produktionsregeln der entsprechenden Grammatik:

Beispiel



Für $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_M = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$
- $S := q_0$
- P besteht aus den folgenden Produktionsregeln:

$$q \rightarrow \mathbf{a}q' \quad \text{falls } \delta(q, \mathbf{a}) = q'$$

Produktionsregeln der entsprechenden Grammatik:

$$q_1 \rightarrow \mathbf{a}q_2$$

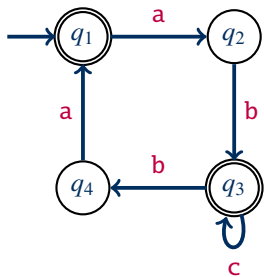
$$q_2 \rightarrow \mathbf{b}q_3$$

$$q_3 \rightarrow \mathbf{c}q_3$$

$$q_3 \rightarrow \mathbf{b}q_4$$

$$q_4 \rightarrow \mathbf{a}q_1$$

Beispiel



Für $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_M = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$
- $S := q_0$
- P besteht aus den folgenden Produktionsregeln:

$$q \rightarrow aq' \quad \text{falls } \delta(q, a) = q'$$

$$q \rightarrow a \quad \text{falls } \delta(q, a) \in F$$

Produktionsregeln der entsprechenden Grammatik:

$$q_1 \rightarrow aq_2$$

$$q_2 \rightarrow bq_3$$

$$q_3 \rightarrow cq_3$$

$$q_3 \rightarrow bq_4$$

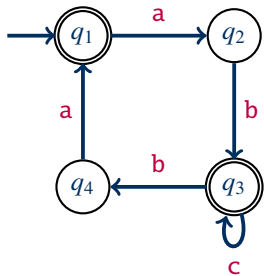
$$q_4 \rightarrow aq_1$$

$$q_2 \rightarrow b$$

$$q_3 \rightarrow c$$

$$q_4 \rightarrow a$$

Beispiel



Für $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ ergibt sich $G_M = \langle V, \Sigma, P, S \rangle$ wie folgt:

- $V := Q$
- $S := q_0$
- P besteht aus den folgenden Produktionsregeln:

$$q \rightarrow aq' \quad \text{falls } \delta(q, a) = q'$$

$$q \rightarrow a \quad \text{falls } \delta(q, a) \in F$$

$$q_0 \rightarrow \epsilon \quad \text{falls } q_0 \in F$$

Produktionsregeln der entsprechenden Grammatik:

$$q_1 \rightarrow aq_2 \quad q_2 \rightarrow bq_3 \quad q_3 \rightarrow cq_3 \quad q_3 \rightarrow bq_4 \quad q_4 \rightarrow aq_1$$

$$q_2 \rightarrow b \quad q_3 \rightarrow c \quad q_4 \rightarrow a$$

$$q_1 \rightarrow \epsilon$$

DFAs akzeptieren reguläre Sprachen: Beweis

Satz: Die Klasse der Sprachen, die durch einen DFA erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis (Fortsetzung): Wir müssen noch die Korrektheit des angegebenen Verfahrens zeigen, also dass $\mathbf{L}(\mathcal{M}) = \mathbf{L}(G_{\mathcal{M}})$.

Dazu beweisen wir beide Richtungen einzeln:

„ \subseteq “: Jedes von \mathcal{M} akzeptierte Wort kann von $G_{\mathcal{M}}$ erzeugt werden.

„ \supseteq “: Jedes von $G_{\mathcal{M}}$ erzeugte Wort kann von \mathcal{M} akzeptiert werden.

$$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

$$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

Falls $w = \epsilon$, dann ist $q_0 \in F$.

$$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

Falls $w = \epsilon$, dann ist $q_0 \in F$.

- Dann hat $G_{\mathcal{M}}$ eine Regel $q_0 \rightarrow \epsilon$.
- Also ist $w \in \mathbf{L}(G_{\mathcal{M}})$.

$$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

Falls $w = \epsilon$, dann ist $q_0 \in F$.

- Dann hat $G_{\mathcal{M}}$ eine Regel $q_0 \rightarrow \epsilon$.
- Also ist $w \in \mathbf{L}(G_{\mathcal{M}})$.

Falls $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ mit $n \geq 1$, dann gilt $\delta(q_0, w) \in F$.

$$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

Falls $w = \epsilon$, dann ist $q_0 \in F$.

- Dann hat $G_{\mathcal{M}}$ eine Regel $q_0 \rightarrow \epsilon$.
- Also ist $w \in \mathbf{L}(G_{\mathcal{M}})$.

Falls $w = \mathbf{a_1} \cdots \mathbf{a_n}$ mit $n \geq 1$, dann gilt $\delta(q_0, w) \in F$.

- Dann gibt es Übergänge $q_0 \xrightarrow{\mathbf{a_1}} q_1 \xrightarrow{\mathbf{a_2}} \dots \xrightarrow{\mathbf{a_n}} q_n$ mit $q_n \in F$

$$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

Falls $w = \epsilon$, dann ist $q_0 \in F$.

- Dann hat $G_{\mathcal{M}}$ eine Regel $q_0 \rightarrow \epsilon$.
- Also ist $w \in \mathbf{L}(G_{\mathcal{M}})$.

Falls $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ mit $n \geq 1$, dann gilt $\delta(q_0, w) \in F$.

- Dann gibt es Übergänge $q_0 \xrightarrow{\mathbf{a}_1} q_1 \xrightarrow{\mathbf{a}_2} \dots \xrightarrow{\mathbf{a}_n} q_n$ mit $q_n \in F$
- Dann hat $G_{\mathcal{M}}$ die Regeln:

$$q_0 \rightarrow \mathbf{a}_1 q_1 \qquad q_1 \rightarrow \mathbf{a}_2 q_2 \qquad \dots \qquad q_{n-1} \rightarrow \mathbf{a}_n$$

$$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

Falls $w = \epsilon$, dann ist $q_0 \in F$.

- Dann hat $G_{\mathcal{M}}$ eine Regel $q_0 \rightarrow \epsilon$.
- Also ist $w \in \mathbf{L}(G_{\mathcal{M}})$.

Falls $w = \mathbf{a_1 \cdots a_n}$ mit $n \geq 1$, dann gilt $\delta(q_0, w) \in F$.

- Dann gibt es Übergänge $q_0 \xrightarrow{\mathbf{a_1}} q_1 \xrightarrow{\mathbf{a_2}} \dots \xrightarrow{\mathbf{a_n}} q_n$ mit $q_n \in F$
- Dann hat $G_{\mathcal{M}}$ die Regeln:

$$q_0 \rightarrow \mathbf{a_1}q_1 \qquad q_1 \rightarrow \mathbf{a_2}q_2 \qquad \dots \qquad q_{n-1} \rightarrow \mathbf{a_n}$$

- Durch Anwendung dieser Regeln kann man ableiten:

$$q_0 \Rightarrow \mathbf{a_1}q_1 \Rightarrow \mathbf{a_1a_2}q_2 \Rightarrow \dots \Rightarrow \mathbf{a_1a_2 \cdots a_{n-1}}q_{n-1} \Rightarrow \mathbf{a_1a_2 \cdots a_{n-1}a_n}$$

$\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$

Wir betrachten ein beliebiges Wort $w \in \mathbf{L}(\mathcal{M})$.

Falls $w = \epsilon$, dann ist $q_0 \in F$.

- Dann hat $G_{\mathcal{M}}$ eine Regel $q_0 \rightarrow \epsilon$.
- Also ist $w \in \mathbf{L}(G_{\mathcal{M}})$.

Falls $w = \mathbf{a_1 \cdots a_n}$ mit $n \geq 1$, dann gilt $\delta(q_0, w) \in F$.

- Dann gibt es Übergänge $q_0 \xrightarrow{\mathbf{a_1}} q_1 \xrightarrow{\mathbf{a_2}} \dots \xrightarrow{\mathbf{a_n}} q_n$ mit $q_n \in F$
- Dann hat $G_{\mathcal{M}}$ die Regeln:

$$q_0 \rightarrow \mathbf{a_1}q_1 \qquad q_1 \rightarrow \mathbf{a_2}q_2 \qquad \dots \qquad q_{n-1} \rightarrow \mathbf{a_n}$$

- Durch Anwendung dieser Regeln kann man ableiten:

$$q_0 \Rightarrow \mathbf{a_1}q_1 \Rightarrow \mathbf{a_1a_2}q_2 \Rightarrow \dots \Rightarrow \mathbf{a_1a_2 \cdots a_{n-1}}q_{n-1} \Rightarrow \mathbf{a_1a_2 \cdots a_{n-1}a_n}$$

- Also ist $w \in \mathbf{L}(G_{\mathcal{M}})$.

Somit gilt $w \in \mathbf{L}(G_{\mathcal{M}})$ für beliebige $w \in \mathbf{L}(\mathcal{M})$, d.h. $\mathbf{L}(\mathcal{M}) \subseteq \mathbf{L}(G_{\mathcal{M}})$.

$$\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$$

$G_{\mathcal{M}}$ hat drei Typen von Regeln: $q \rightarrow \mathbf{a}q'$, $q \rightarrow \mathbf{a}$ und $q_0 \rightarrow \epsilon$.

$$\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$$

$G_{\mathcal{M}}$ hat drei Typen von Regeln: $q \rightarrow \mathbf{a}q'$, $q \rightarrow \mathbf{a}$ und $q_0 \rightarrow \epsilon$.

Für ein Wort $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ sind zwei Arten von Ableitungen denkbar:

$$(1) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n$$

$$(2) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n q_n \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_n$$

$$\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$$

$G_{\mathcal{M}}$ hat drei Typen von Regeln: $q \rightarrow \mathbf{a}q'$, $q \rightarrow \mathbf{a}$ und $q_0 \rightarrow \epsilon$.

Für ein Wort $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ sind zwei Arten von Ableitungen denkbar:

$$(1) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n$$

$$(2) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n q_n \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_n$$

In Fall (1) wurden Regeln der folgenden Form angewendet:

$$q_0 \rightarrow \mathbf{a}_1 q_1 \qquad q_1 \rightarrow \mathbf{a}_2 q_2 \qquad \dots \qquad q_{n-1} \rightarrow \mathbf{a}_n$$

Also hat \mathcal{M} die folgenden Übergänge:

$$q_0 \xrightarrow{\mathbf{a}_1} q_1 \qquad q_1 \xrightarrow{\mathbf{a}_2} q_2 \qquad \dots \qquad q_{n-1} \xrightarrow{\mathbf{a}_n} q_n$$

wobei $q_n \in F$ ein Endzustand ist.

$$\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$$

$G_{\mathcal{M}}$ hat drei Typen von Regeln: $q \rightarrow \mathbf{a}q'$, $q \rightarrow \mathbf{a}$ und $q_0 \rightarrow \epsilon$.

Für ein Wort $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ sind zwei Arten von Ableitungen denkbar:

$$(1) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n$$

$$(2) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n q_n \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_n$$

In Fall (1) wurden Regeln der folgenden Form angewendet:

$$q_0 \rightarrow \mathbf{a}_1 q_1 \qquad q_1 \rightarrow \mathbf{a}_2 q_2 \qquad \dots \qquad q_{n-1} \rightarrow \mathbf{a}_n$$

Also hat \mathcal{M} die folgenden Übergänge:

$$q_0 \xrightarrow{\mathbf{a}_1} q_1 \qquad q_1 \xrightarrow{\mathbf{a}_2} q_2 \qquad \dots \qquad q_{n-1} \xrightarrow{\mathbf{a}_n} q_n$$

wobei $q_n \in F$ ein Endzustand ist.

Also gilt $\delta(q_0, \mathbf{a}_1 \cdots \mathbf{a}_n) = q_n \in F$ und \mathcal{M} akzeptiert das Wort w .

$$\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$$

$G_{\mathcal{M}}$ hat drei Typen von Regeln: $q \rightarrow \mathbf{a}q'$, $q \rightarrow \mathbf{a}$ und $q_0 \rightarrow \epsilon$

Für ein Wort $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ sind zwei Arten von Ableitungen denkbar:

$$(1) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n$$

$$(2) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n q_n \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_n$$

$$\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$$

$G_{\mathcal{M}}$ hat drei Typen von Regeln: $q \rightarrow \mathbf{a}q'$, $q \rightarrow \mathbf{a}$ und $q_0 \rightarrow \epsilon$

Für ein Wort $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ sind zwei Arten von Ableitungen denkbar:

$$(1) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n$$

$$(2) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n q_n \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_n$$

In Fall (2) wurden Regeln der folgenden Form angewendet:

$$q_0 \rightarrow \mathbf{a}_1 q_1 \quad q_1 \rightarrow \mathbf{a}_2 q_2 \quad \dots \quad q_{n-1} \rightarrow \mathbf{a}_n q_n \quad q_n \rightarrow \epsilon$$

Also ist $q_n = q_0$ mit $q_0 \in F$ und \mathcal{M} hat die folgenden Übergänge:

$$q_0 \xrightarrow{\mathbf{a}_1} q_1 \quad q_1 \xrightarrow{\mathbf{a}_2} q_2 \quad \dots \quad q_{n-1} \xrightarrow{\mathbf{a}_n} q_0$$

$$\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$$

$G_{\mathcal{M}}$ hat drei Typen von Regeln: $q \rightarrow \mathbf{a}q'$, $q \rightarrow \mathbf{a}$ und $q_0 \rightarrow \epsilon$

Für ein Wort $w = \mathbf{a}_1 \cdots \mathbf{a}_n$ sind zwei Arten von Ableitungen denkbar:

$$(1) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n$$

$$(2) q_0 \Rightarrow \mathbf{a}_1 q_1 \Rightarrow \dots \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} q_{n-1} \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_{n-1} \mathbf{a}_n q_n \Rightarrow \mathbf{a}_1 \cdots \mathbf{a}_n$$

In Fall (2) wurden Regeln der folgenden Form angewendet:

$$q_0 \rightarrow \mathbf{a}_1 q_1 \quad q_1 \rightarrow \mathbf{a}_2 q_2 \quad \dots \quad q_{n-1} \rightarrow \mathbf{a}_n q_n \quad q_n \rightarrow \epsilon$$

Also ist $q_n = q_0$ mit $q_0 \in F$ und \mathcal{M} hat die folgenden Übergänge:

$$q_0 \xrightarrow{\mathbf{a}_1} q_1 \quad q_1 \xrightarrow{\mathbf{a}_2} q_2 \quad \dots \quad q_{n-1} \xrightarrow{\mathbf{a}_n} q_0$$

Also gilt $\delta(q_0, \mathbf{a}_1 \cdots \mathbf{a}_n) = q_0 \in F$ und \mathcal{M} akzeptiert das Wort w .

Zusammengefasst gilt somit $w \in \mathbf{L}(\mathcal{M})$ für beliebige $w \in \mathbf{L}(G_{\mathcal{M}})$, d.h. $\mathbf{L}(\mathcal{M}) \supseteq \mathbf{L}(G_{\mathcal{M}})$. \square

Reguläre Grammatiken und DFAs

Wir haben bisher gezeigt:

Jede von DFA erkannte Sprache ist regulär.

Für die Umkehrung müsste man reguläre Grammatiken in DFAs übersetzen.

Reguläre Grammatiken und DFAs

Wir haben bisher gezeigt:

Jede von DFA erkannte Sprache ist regulär.

Für die Umkehrung müsste man reguläre Grammatiken in DFAs übersetzen.

Kann man die Übersetzung nicht einfach umdrehen?

- Für jede reguläre Regel $A \rightarrow aB$ definieren wir $\delta(A, a) = B$
- Für jede reguläre Regel $A \rightarrow a$ definieren wir $\delta(A, a) = C$ mit $C \in F$

Warum funktioniert das nicht?

Reguläre Grammatiken und DFAs

Wir haben bisher gezeigt:

Jede von DFA erkannte Sprache ist regulär.

Für die Umkehrung müsste man reguläre Grammatiken in DFAs übersetzen.

Kann man die Übersetzung nicht einfach umdrehen?

- Für jede reguläre Regel $A \rightarrow aB$ definieren wir $\delta(A, a) = B$
- Für jede reguläre Regel $A \rightarrow a$ definieren wir $\delta(A, a) = C$ mit $C \in F$

Warum funktioniert das nicht?

Weil die Übergangsfunktion dann nicht immer eindeutig definiert wäre!

Beispiel: Eine Grammatik kann die Regeln $S \rightarrow aA$, $S \rightarrow aS$ und $A \rightarrow \epsilon$ haben, aber wir können nicht $\delta(S, a) = A$ und $\delta(S, a) = S$ gleichzeitig fordern.

Nichtdeterministische Übergänge

Kann die Übergangsfunktion „mehr als einen Rückgabewert“ haben?

↪ Technisch wäre das darstellbar mittels Mengen, z.B. $\delta(q, a) = \{q_1, q_2\}$.

Was soll das bedeuten?

- Der Automat hat die Wahl zwischen mehreren Übergängen.
- Die Verarbeitung eines Wortes wird **nichtdeterministisch**, weil die Eingabe nicht völlig bestimmt, in welchen Zustand der Automat gelangt.
- Der Automat akzeptiert ein Wort, wenn es eine „richtige“ Wahl von Zustandsübergängen gibt (d.h. eine, die zu einem Endzustand führt).

Nichtdeterministische Automaten

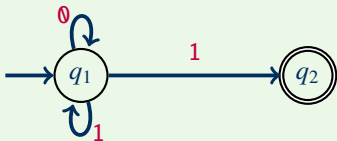
Ein **nichtdeterministischer endlicher Automat** (international: „NFA“) \mathcal{M} ist ein Tupel $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F \rangle$ mit folgenden Bestandteilen:

- Q : endliche Menge von **Zuständen**,
- Σ : Alphabet,
- δ : **Übergangsfunktion**, eine totale Funktion $Q \times \Sigma \rightarrow 2^Q$, wobei 2^Q die Potenzmenge von Q ist;
- Q_0 : Menge möglicher **Startzustände** $Q_0 \subseteq Q$,
- F : Menge von **Endzuständen** $F \subseteq Q$.

Notation: Wir schreiben statt $q' \in \delta(q, \mathbf{a})$ auch $q \xrightarrow{\mathbf{a}} q'$.

Beispiel: NFA

Beispiel:



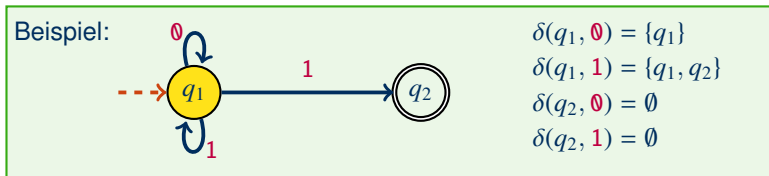
$$\delta(q_1, 0) = \{q_1\}$$

$$\delta(q_1, 1) = \{q_1, q_2\}$$

$$\delta(q_2, 0) = \emptyset$$

$$\delta(q_2, 1) = \emptyset$$

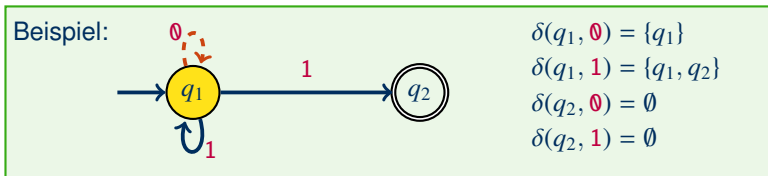
Beispiel: NFA



Wort Zustandsfolge Ergebnis

011 q_1

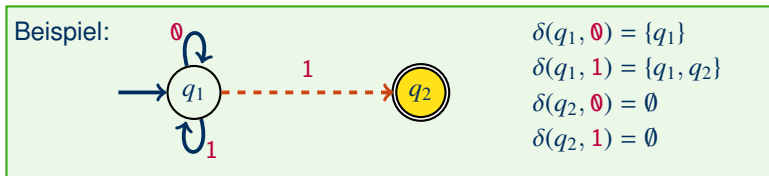
Beispiel: NFA



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

011	$q_1 q_1$	
-----	-----------	--

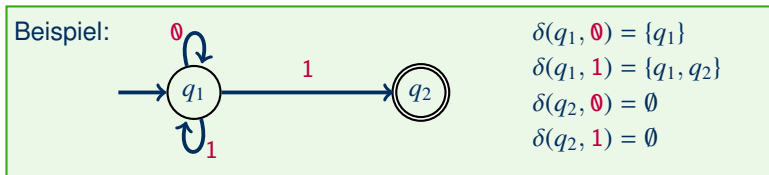
Beispiel: NFA



Wort **Zustandsfolge** **Ergebnis**

011 $q_1 q_1 q_2$

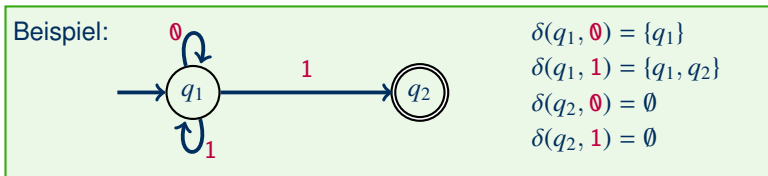
Beispiel: NFA



Wort **Zustandsfolge** **Ergebnis**

011 $q_1 q_1 q_2 ?$

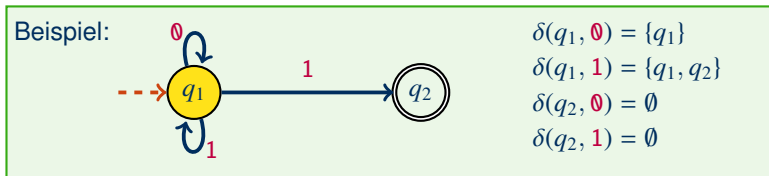
Beispiel: NFA



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

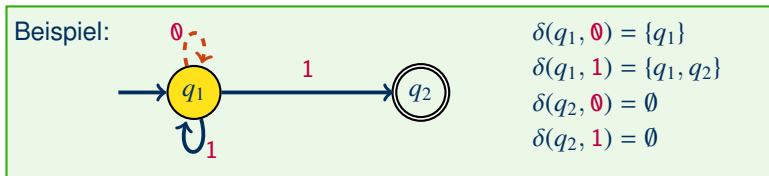
011	$q_1 q_1 q_2 ?$	abgelehnt (fehlender Übergang)
-----	-----------------	--------------------------------

Beispiel: NFA



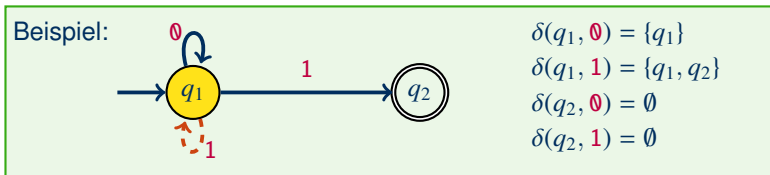
Wort	Zustandsfolge	Ergebnis
011	$q_1 q_1 q_2 ?$	abgelehnt (fehlender Übergang)
011	q_1	

Beispiel: NFA



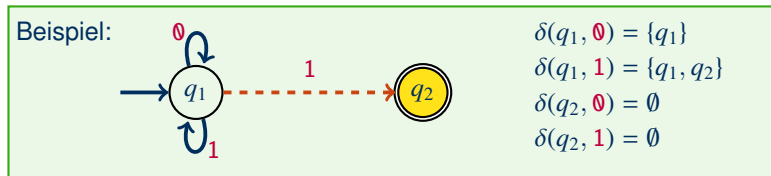
Wort	Zustandsfolge	Ergebnis
011	$q_1 q_1 q_2 ?$	abgelehnt (fehlender Übergang)
011	$q_1 q_1$	

Beispiel: NFA



Wort	Zustandsfolge	Ergebnis
011	$q_1 q_1 q_2 ?$	abgelehnt (fehlender Übergang)
011	$q_1 q_1 q_1$	

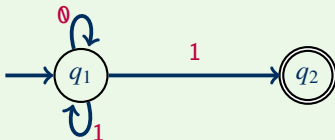
Beispiel: NFA



Wort	Zustandsfolge	Ergebnis
011	$q_1 q_1 q_2 ?$	abgelehnt (fehlender Übergang)
011	$q_1 q_1 q_1 q_2$	

Beispiel: NFA

Beispiel:



$$\delta(q_1, 0) = \{q_1\}$$

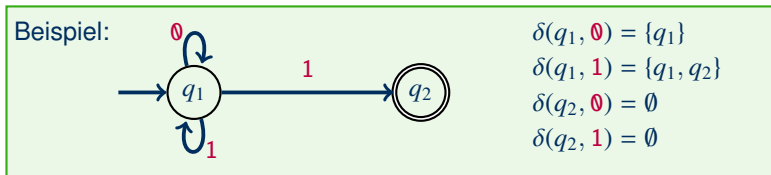
$$\delta(q_1, 1) = \{q_1, q_2\}$$

$$\delta(q_2, 0) = \emptyset$$

$$\delta(q_2, 1) = \emptyset$$

Wort	Zustandsfolge	Ergebnis
011	$q_1 q_1 q_2 ?$	abgelehnt (fehlender Übergang)
011	$q_1 q_1 q_1 q_2$	akzeptiert

Beispiel: NFA



Wort	Zustandsfolge	Ergebnis
------	---------------	----------

011	$q_1 q_1 q_2 ?$	abgelehnt (fehlender Übergang)
-----	-----------------	--------------------------------

011	$q_1 q_1 q_1 q_2$	akzeptiert
-----	-------------------	------------

\leadsto 011 wird nichtdeterministisch akzeptiert

Quiz: Akzeptanz eines NFA

Quiz: Wir betrachten den grafisch gegebenen NFA $\mathcal{M} = \langle Q, \{0, 1\}, \delta, Q_0, F \rangle$:

...

NFAs: Alternative Definitionen

In der Literatur gibt es leicht abgewandelte Definitionen von NFAs:

NFAs: Alternative Definitionen

In der Literatur gibt es leicht abgewandelte Definitionen von NFAs:

- Übergangsrelation statt Übergangsfunktion

Statt einer Funktion $\delta : Q \times \Sigma \rightarrow 2^Q$ kann man auch eine Relation $\Delta \subseteq Q \times \Sigma \times Q$ verwenden, wenn für alle $q, q' \in Q, \sigma \in \Sigma$ gilt:

$$q' \in \delta(q, \sigma) \quad \text{genau dann, wenn} \quad \langle q, \sigma, q' \rangle \in \Delta$$

NFAs: Alternative Definitionen

In der Literatur gibt es leicht abgewandelte Definitionen von NFAs:

- Übergangsrelation statt Übergangsfunktion

Statt einer Funktion $\delta : Q \times \Sigma \rightarrow 2^Q$ kann man auch eine Relation $\Delta \subseteq Q \times \Sigma \times Q$ verwenden, wenn für alle $q, q' \in Q, \sigma \in \Sigma$ gilt:

$$q' \in \delta(q, \sigma) \quad \text{genau dann, wenn} \quad \langle q, \sigma, q' \rangle \in \Delta$$

- Einzelner Startzustand q_0

Manchmal wird statt der Menge Q_0 nur ein Startzustand q_0 verwendet. (Es ist aber leicht, einen NFA unserer Bauart so zu verändern, dass nur ein Startzustand nötig ist.)

NFAs: Alternative Definitionen

In der Literatur gibt es leicht abgewandelte Definitionen von NFAs:

- Übergangsrelation statt Übergangsfunktion

Statt einer Funktion $\delta : Q \times \Sigma \rightarrow 2^Q$ kann man auch eine Relation $\Delta \subseteq Q \times \Sigma \times Q$ verwenden, wenn für alle $q, q' \in Q, \sigma \in \Sigma$ gilt:

$$q' \in \delta(q, \sigma) \quad \text{genau dann, wenn} \quad \langle q, \sigma, q' \rangle \in \Delta$$

- Einzelner Startzustand q_0

Manchmal wird statt der Menge Q_0 nur ein Startzustand q_0 verwendet. (Es ist aber leicht, einen NFA unserer Bauart so zu verändern, dass nur ein Startzustand nötig ist.)

- Einzelner Endzustand q_f

Man kann auch die Menge der Endzustände F leicht auf ein einziges Argument reduzieren.

Ist Nichtdeterminismus sinnvoll?

Nichtdeterministische Automaten müssen jeweils den richtigen Übergang „erraten“.

~> Das entspricht nicht der Funktionsweise echter Computer . . .

Ist Nichtdeterminismus sinnvoll?

Nichtdeterministische Automaten müssen jeweils den richtigen Übergang „erraten“.

↪ Das entspricht nicht der Funktionsweise echter Computer . . .

Dennoch ist Nichtdeterminismus ein wichtiges Prinzip in der Informatik:

- Kann **kompaktere, natürlichere Darstellungen** ermöglichen.
- Beschreibt treffend die Schwierigkeit vieler praktischer Probleme – wichtig für **Untersuchung von Komplexität und Berechenbarkeit**.
- Ist relevant in der **Modellierung parallel arbeitender Systeme**.
- Bildet Ausgangspunkt für die **Entwicklung deterministischer Algorithmen**.

Zusammenfassung und Ausblick

Das **Wortproblem** ist eine von mehreren wichtigen Fragestellungen zu formalen Sprachen.

Deterministische endliche Automaten (DFAs) können mit partieller oder totaler Übergangsfunktion definiert werden.

DFAs lösen das Wortproblem für reguläre Sprachen.

(Bisher gezeigt: Jede durch DFAs erkennbare Sprache ist regulär.)

Nichtdeterminismus ermöglicht endlichen Automaten, bei der Worterkennung zu „raten“.

Nächste Themen:

- Was bringen uns nichtdeterministische Automaten?
- Noch zu zeigen: Alle regulären Sprachen sind durch DFAs erkennbar.
- Nichtdeterministische Automaten
- Gibt es noch mehr Darstellungsformen für reguläre Sprachen?

Bildrechte

Folie 4:

- Portrait John Backus: Wikipedia-Nutzer **Pierre.Lescanne**, CC-By-SA 4.0
- Portrait Peter Naur: Wikipedia-Nutzer **Eriktj**, CC-By-SA 3.0
- Portrait Niklaus Wirth: Wikipedia-Nutzer **Tyomitch**, used with permission of copyright holder
- Foto Büste Pānini: Wikipedia-Nutzer **Jameela P.**, CC-By-SA 4.0