TECHNISCHE UNIVERSITÄT DRESDEN

FORMALE SYSTEME

4. Vorlesung: Nichtdeterministische Endliche Automaten

Markus Krötzsch

Lehrstuhl Wissensbasierte Systeme

TU Dresden, 20. Oktober 2016

Wiederholung

- Grammatiken können Sprachen beschreiben und sie grob in Typen unterteilen
- Typ-3-Grammatiken generieren reguläre Sprachen
- Deterministische endliche Automaten erkennen reguläre Sprachen
- Nichtdeterministische endliche Automaten verallgemeinern die Definition der Übergangsfunktion: der Automat „rät", welcher Übergang der richtige ist

Wiederholung: NFA

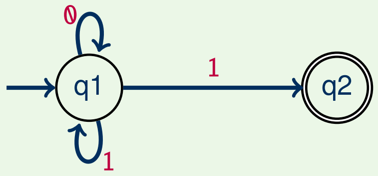
Ein nichtdeterministischer endlicher Automat (international: „NFA") $\mathcal{M}$ ist ein Tupel $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ mit den folgenden Bestandteilen:

- Q: endliche Menge von Zuständen
- Σ: Alphabet
- δ: Übergangsfunktion, eine totale Funktion $Q \times \Sigma \to 2^Q$, 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 $q' \in \delta(q, \mathtt{a})$ auch $q \xrightarrow{\mathtt{a}} q'$.

Beispiel: NFA

Beispiel:



$\delta(q_1, \mathtt{0}) = \{q_1\}$
$\delta(q_1, \mathtt{1}) = \{q_1, q_2\}$
$\delta(q_2, \mathtt{0}) = \emptyset$
$\delta(q_2, \mathtt{1}) = \emptyset$

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

$\rightsquigarrow$ `011` wird nichtdeterministisch akzeptiert

Die Sprache eines NFA

Läufe eines NFA

Ein Lauf eines NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F \rangle$ für ein Wort $w = \sigma_1 \cdots \sigma_n$ ist eine Folge von Zuständen $q_0 \ldots q_m$, so dass gilt:

- $q_0 \in Q_0$
- $q_{i+1} \in \delta(q_i, \sigma_i)$ für alle $0 \leq i < m$
- (1) $m = |w| = n$ oder (2) $m < n$ und $\delta(q_m, \sigma_m) = \emptyset$

Ein Lauf heißt akzeptierend, falls $m = n$ und $q_n \in F$. Andernfalls heißt der Lauf verwerfend.

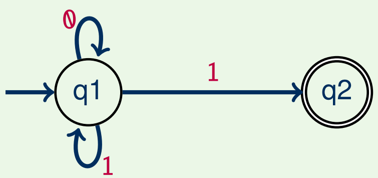
$\leadsto$ Ein DFA hat genau einen Lauf für jedes Wort.
Er akzeptiert wenn dieser Lauf akzeptierend ist.

$\leadsto$ Ein NFA kann für ein Wort mehrere Läufe haben.
Er akzeptiert wenn einer dieser Läufe akzeptierend ist.

Sprache eines NFA

Die Sprache eines NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F \rangle$ ist die Menge aller Wörter w für die $\mathcal{M}$ einen akzeptierenden Lauf hat.

Beispiel:

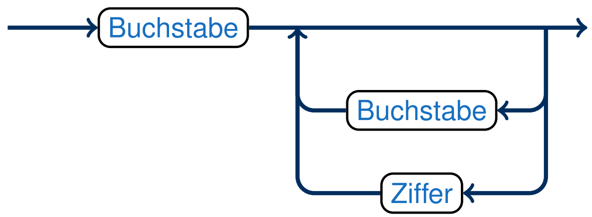


$\delta(q_1, \mathtt{0}) = \{q_1\}$
$\delta(q_1, \mathtt{1}) = \{q_1, q_2\}$
$\delta(q_2, \mathtt{0}) = \emptyset$
$\delta(q_2, \mathtt{1}) = \emptyset$

Wort	Lauf	Ergebnis
011	$q_1\, q_1\, q_2$	verwerfend (zu kurz)
011	$q_1\, q_1\, q_1\, q_2$	akzeptierend
011	$q_1\, q_1\, q_1\, q_1$	verwerfend (kein Endzustand)

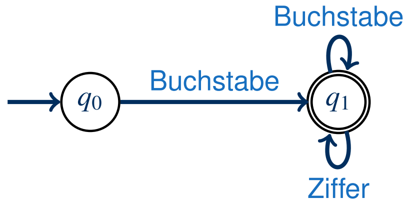
$\mathbf{L}(\mathcal{M}) = \{\mathtt{0}, \mathtt{1}\}^* \circ \{\mathtt{1}\}$

NFA zur Darstellung von Syntaxdiagrammen



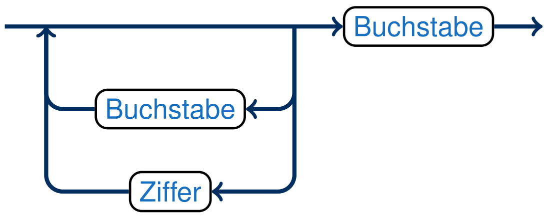
Übersetzung in NFA:

- zusammenhängende Linienbereiche werden Zustände
- Knoten werden Übergänge

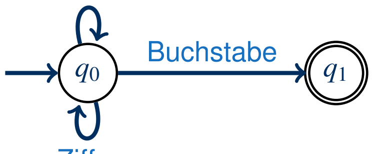


Syntaxdiagramme und Nichtdeterminismus

Das folgende Beispiel führt zu einem NFA, der kein DFA ist:



Entsprechender NFA:



Verallgemeinerte NFA-Übergangsfunktion

Wie beim DFA können wir auch bei einem NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ eine erweiterte Übergangsfunktion definieren, die ganze Wörter einliest

Zuerst erweitern wir δ auf Mengen von Zuständen:

Für eine Zustandsmenge $R \subseteq Q$ und ein Terminalsymbol $\mathtt{a}$ sei

$$\delta(R, \mathtt{a}) = \bigcup_{q \in R} \delta(q, \mathtt{a}).$$

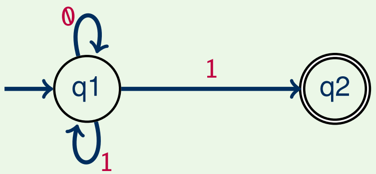
Dann erweitern wir δ von einzelnen Symbolen zu beliebigen Wörtern:

Für eine Zustandsmenge $R \subseteq Q$ und ein Wort $w \in \Sigma^*$ sei $\delta(R, w)$ die Menge aller Zustände, die man erreichen kann, wenn man in einem Zustand aus R beginnt und das Wort w einliest, formal:

- $\delta(R, \epsilon) = R$
- $\delta(R, \mathtt{a}v) = \delta(\delta(R, \mathtt{a}), v)$

Beispiel

Beispiel:



$\delta(q_1, \mathtt{0}) = \{q_1\}$
$\delta(q_1, \mathtt{1}) = \{q_1, q_2\}$
$\delta(q_2, \mathtt{0}) = \emptyset$
$\delta(q_2, \mathtt{1}) = \emptyset$

Die Menge der Startzustände ist $Q_0 = \{q_1\}$.

Dann gilt:

$$\delta(Q_0, \mathtt{0}) = \delta(q_1, \mathtt{0}) = \{q_1\}$$

$$\delta(Q_0, \mathtt{1}) = \delta(q_1, \mathtt{1}) = \{q_1, q_2\}$$

$$\begin{aligned}\delta(Q_0, \mathtt{10}) &= \delta(\delta(Q_0, \mathtt{1}), \mathtt{0}) = \delta(\{q_1, q_2\}, \mathtt{0}) \\ &= \delta(q_1, \mathtt{0}) \cup \delta(q_2, \mathtt{0}) = \{q_1\} \cup \emptyset = \{q_1\}\end{aligned}$$

$$\delta(Q_0, \mathtt{01}) = \delta(\delta(Q_0, \mathtt{0}), \mathtt{1}) = \delta(\{q_1\}, \mathtt{1}) = \{q_1, q_2\}$$

Sprache eines NFA (2. Version)

Die erweiterte Übergangsfunktion hilft bei der Definition der Sprache, die ein NFA akzeptiert:

Die Sprache eines NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ ist die Menge

$$\mathbf{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \delta(Q_0, w) \cap F \neq \emptyset\}$$

Die Bedingung „$\delta(Q_0, w) \cap F \neq \emptyset$“ bedeutet:

„mindestens einer der Zustände, die man durch Einlesen von w von einem Startzustand aus erreichen kann, ist ein Endzustand.“

Behauptung: Diese Variante stimmt mit der vorherigen (mit akzeptierenden Läufen) überein.

Äquivalenz der Sprachdefinitionen für NFAs

Sei $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ NFA und $w = \sigma_1 \cdots \sigma_n \in \Sigma^*$ ein Wort.

Behauptung: Es gibt einen akzeptierenden Lauf für w genau dann wenn $\delta(Q_0, w) \cap F \neq \emptyset$.

Beweis „⇒“: Angenommen es gibt einen akzeptierenden Lauf $q_0 \ldots q_n$ für w.

- Dann ist $q_n \in F$.
- Wir behaupten $q_n \in \delta(Q_0, w)$ (damit folgt $\delta(Q_0, w) \cap F \neq \emptyset$)
- Wir zeigen die stärkere Behauptung $q_i \in \delta(Q_0, \sigma_1 \cdots \sigma_i)$ für alle $0 \leq i \leq n$ mittels Induktion über $|w|$:
  - Induktionsanfang: Für $i = 0$ gilt $q_0 \in Q_0 = \delta(Q_0, \epsilon)$
  - Induktionshypothese: die Behauptung gelte für i
  - Induktionsschritt: für $i + 1$ gilt:
    $q_i \in \delta(Q_0, \sigma_1 \cdots \sigma_i)$ (Induktionshypothese)
    $q_{i+1} \in \delta(q_i, \sigma_{i+1})$ (laut Definition eines Laufs)
    $q_{i+1} \in \delta(\delta(Q_0, \sigma_1 \cdots \sigma_i), \sigma_{i+1}) = \delta(Q_0, \sigma_1 \cdots \sigma_i\sigma_{i+1})$

Äquivalenz der Sprachdefinitionen für NFAs

Sei $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ NFA und $w = \sigma_1 \cdots \sigma_n \in \Sigma^*$ ein Wort.

Behauptung: Es gibt einen akzeptierenden Lauf für w genau dann wenn $\delta(Q_0, w) \cap F \neq \emptyset$.

Beweis „⇐“: Angenommen $\delta(Q_0, w) \cap F \neq \emptyset$.

- Wir ermitteln einen akzeptierenden Lauf $q_0 \ldots q_n$ für w
- Dazu gehen wir rückwärts vor:
  - Wähle $q_n \in F \cap \delta(Q_0, w)$
  - Für alle $i = n, \ldots, 1$:
    Wähle $q_{i-1} \in \delta(Q_0, \sigma_0 \cdots \sigma_{i-1})$, so dass $q_i \in \delta(q_{i-1}, \sigma_i)$
- Dies ist ein Lauf, da $q_0 \in \delta(Q_0, \epsilon) = Q_0$ und alle Übergänge erlaubt sind.
- Es ist ein akzeptierender Lauf, da $q_n \in F$. □

NFA vs. DFA

Vergleich DFA – NFA

Offensichtlich sind NFAs allgemeiner als DFAs:

> Satz: Jeder DFA kann als NFA aufgefasst werden. Daher wird jede von einem DFA akzeptierbare Sprache auch von einen NFA akzeptiert.

Beweis: Für jeden DFA $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F\rangle$ gibt es einen entsprechenden NFA $\mathcal{M}' = \langle Q, \Sigma, \delta_{\mathsf{NFA}}, \{q_0\}, F\rangle$ mit $\delta_{\mathsf{NFA}}(q, \mathtt{a}) = \{\delta(q, \mathtt{a})\}$. □

Die Umkehrung dieses Satzes gilt allerdings auch:

> Satz: Jede von einem NFA akzeptierbare Sprache wird auch von einen DFA akzeptiert.

In diesem Sinne sind NFA nicht ausdrucksstärker als DFA – wie kann das sein?

NFAs als DFAs – Idee

Die verallgemeinerte NFA-Übergangsfunktion bildet Mengen von Zuständen auf Mengen von Zuständen ab:

$$\delta(R, \mathtt{a}) = \bigcup_{q \in R} \delta(q, \mathtt{a}).$$

„Wenn der Automat in einem der Zustände R ist und $\mathtt{a}$ liest, so ist er anschließend in einem der Zustände der Menge $\delta(R, \mathtt{a})$."

Dieser Übergang zwischen Mengen möglicher Zustände ist an sich deterministisch.

$\rightsquigarrow$ wir können einen NFA deterministisch simulieren, indem wir die Menge der möglichen Zustände berechnen

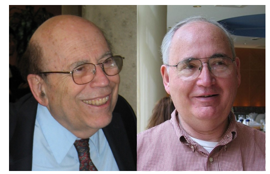
Die Potenzmengenkonstruktion

Für einen NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ definieren wir den Potenzmengen-DFA $\mathcal{M}_{\text{DFA}} = \langle Q_{\text{DFA}}, \Sigma, \delta_{\text{DFA}}, q_0, F_{\text{DFA}}\rangle$ wie folgt:

- $Q_{\text{DFA}} = 2^Q$ (Potenzmenge von Q)
- $\delta_{\text{DFA}}(R, \mathtt{a}) = \bigcup_{q \in R} \delta(q, \mathtt{a})$
- $q_0 = Q_0$
- $F_{\text{DFA}} = \{R \in 2^Q \mid R \cap F \neq \emptyset\}$

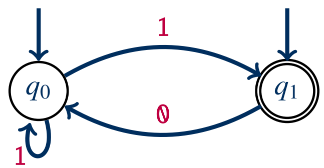
Satz (Rabin/Scott): $\mathbf{L}(\mathcal{M}) = \mathbf{L}(\mathcal{M}_{\text{DFA}})$

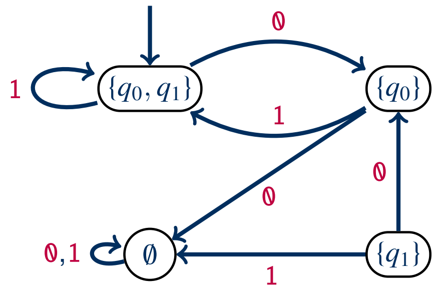
(Beweis später)



Michael Oser Rabin Dana Scott

Beispiel Potenzmengenkonstruktion



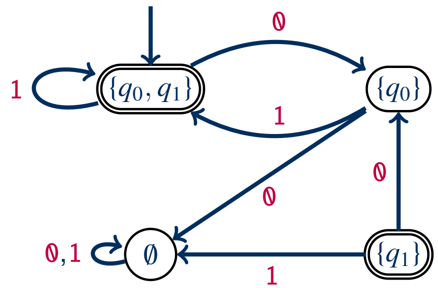


$\delta_{\text{DFA}}(\{q_0\}, \mathtt{0}) = \emptyset$
$\delta_{\text{DFA}}(\{q_0\}, \mathtt{1}) = \{q_0, q_1\}$
$\delta_{\text{DFA}}(\{q_1\}, \mathtt{0}) = \{q_0\}$
$\delta_{\text{DFA}}(\{q_1\}, \mathtt{1}) = \emptyset$
$\delta_{\text{DFA}}(\{q_0, q_1\}, \mathtt{0}) = \{q_0\}$
$\delta_{\text{DFA}}(\{q_0, q_1\}, \mathtt{1}) = \{q_0, q_1\}$
$\delta_{\text{DFA}}(\emptyset, \mathtt{0}) = \emptyset$
$\delta_{\text{DFA}}(\emptyset, \mathtt{1}) = \emptyset$

Erkannte Sprache:
$\{\mathtt{1}\}^* \circ (\{\mathtt{0}\} \circ \{\mathtt{1}\}^*)^*$

Vereinfachung Potenzmengenkonstruktion

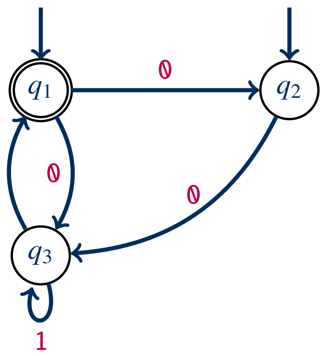
Der Automat aus dem vorherigen Beispiel kann vereinfacht werden:

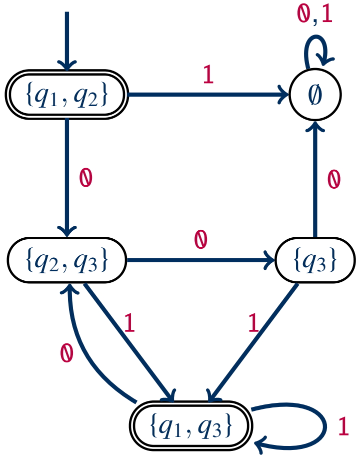


- Zustand $\{q_1\}$ ist unerreichbar
- Zustand $\emptyset$ kann nicht verlassen werden (irrelevant für akzeptierende Läufe)

Potenzmengenkonstruktion „on the fly“

Vermeidung unnötiger Zustände durch schrittweise Konstruktion vom Startzustand:





- Erreichbarer Teil spart drei Zustände ein
- Zustand $\emptyset$ wie zuvor unnötig

Potenzmengenkonstruktion: Korrektheit

Satz (Rabin/Scott): $\mathbf{L}(\mathcal{M}) = \mathbf{L}(\mathcal{M}_{\mathsf{DFA}})$

Beweis: Wir nutzen die Korrespondenz der verallgemeinerten Übergangsfunktionen aus. Zuerst zeigen wir, dass für jedes Wort $w \in \Sigma^*$ und jede Zustandsmenge R gilt: $\delta_{\mathsf{DFA}}(R, w) = \delta(R, w)$.

Induktionsanfang:

(1) $\delta_{\mathsf{DFA}}(R, \epsilon) = R = \delta(R, \epsilon)$

(2) $\delta_{\mathsf{DFA}}(R, \mathtt{a}) = \bigcup_{q \in R} \delta(q, \mathtt{a}) = \delta(R, \mathtt{a})$

Induktionshypothese: $\delta_{\mathsf{DFA}}(R, v) = \delta(R, v)$ für Wörter v mit $|v| < |w|$

Induktionsschritt:

(3) $$\begin{aligned} \delta_{\mathsf{DFA}}(R, \mathtt{a}v) &= \delta_{\mathsf{DFA}}(\delta_{\mathsf{DFA}}(R, \mathtt{a}), v) \\ &= \delta_{\mathsf{DFA}}(\delta(R, \mathtt{a}), v) && \text{(wegen (2))} \\ &= \delta(\delta(R, \mathtt{a}), v) && \text{(Induktionshypothese)} \\ &= \delta(R, \mathtt{a}v) \end{aligned}$$

Potenzmengenkonstruktion: Korrektheit (2)

Satz (Rabin/Scott): $\mathbf{L}(\mathcal{M}) = \mathbf{L}(\mathcal{M}_{\mathsf{DFA}})$

Beweis (Fortsetzung): Wir haben gezeigt: $\delta_{\mathsf{DFA}}(R, w) = \delta(R, w)$.

Damit ergibt sich, für beliebige Wörter $w \in \Sigma^*$:

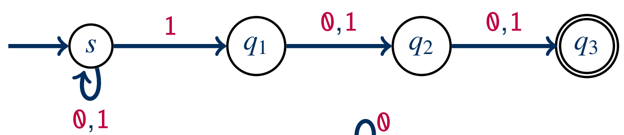
$$\begin{aligned} w \in \mathbf{L}(\mathcal{M}) \quad &\text{gdw.} \quad \delta(Q_0, w) \cap F \neq \emptyset \\ &\text{gdw.} \quad \delta_{\mathsf{DFA}}(Q_0, w) \cap F \neq \emptyset \\ &\text{gdw.} \quad \delta_{\mathsf{DFA}}(Q_0, w) \in F_{\mathsf{DFA}} \\ &\text{gdw.} \quad w \in \mathbf{L}(\mathcal{M}_{\mathsf{DFA}}) \end{aligned}$$

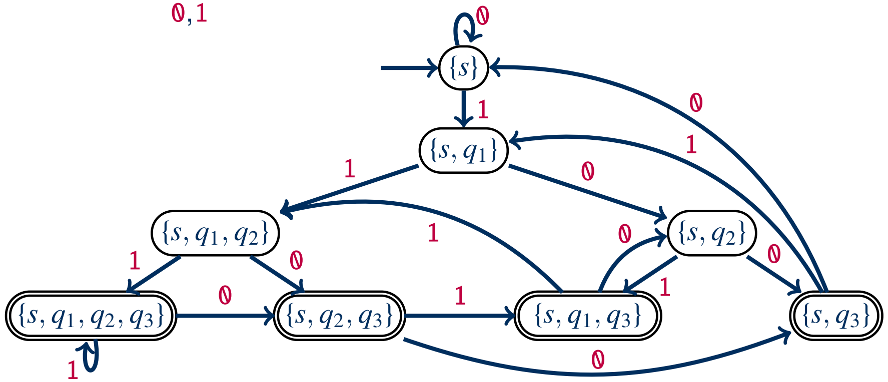
□

Größenvergleich

Der DFA eines NFA hat $2^{|Q|}$ – also exponentiell viele – Zustände. Auch „on the fly“ lässt sich das im Allgemeinen nicht vermeiden.

Beispiel: „Wörter mit 1 an drittletzter Stelle“





Größenvergleich (2)

Allgemein kann man für jede Zahl $n \geq 1$ die Sprache
$\mathbf{L}_n = \{0, 1\}^* 1 \{0, 1\}^{n-1}$ betrachten („Wörter mit 1 an n-letzter Stelle“)

Es gilt:

- Es gibt einen NFA mit $n + 1$ Zuständen, der $\mathbf{L}_n$ erkennt.
- Jeder DFA, der $\mathbf{L}_n$ erkennt, hat mindestens 2^n Zustände.

Schlussfolgerung:
NFAs können exponentiell kompakter sein als äquivalente DFAs.

Zusammenfassung und Ausblick

Nichtdeterministische endliche Automaten (NFA) vereinfachen die Modellierung, z.B. die direkte Darstellung von Syntaxdiagrammen

Rabin/Scott: DFAs und NFAs erkennen die selben Sprachen (Potenzmengenkonstruktion)

Offene Fragen:

- Wir wollten noch zeigen: alle regulären Sprachen sind durch DFAs erkennbar – wie?
- Gibt es noch mehr Darstellungsformen für reguläre Sprachen?
- Wir haben gesehen, dass man Automaten manchmal vereinfachen kann – geht das noch besser?