



FORMALE SYSTEME

4. Vorlesung: Nichtdeterministische Endliche Automaten

Markus Krötzsch

Professur für Wissensbasierte Systeme

TU Dresden, 19. Oktober 2023

Wiederholung

- Grammatiken können Sprachen beschreiben und sie grob in Typen unterteilen
- Typ-3-Grammatiken generieren reguläre Sprachen
- Deterministische endliche Automaten erkennen reguläre Sprachen

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 mehr als einen Wert hätte!

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 Wert“ haben?

↪ darstellbar als Menge, 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, die zu einem Endzustand führt

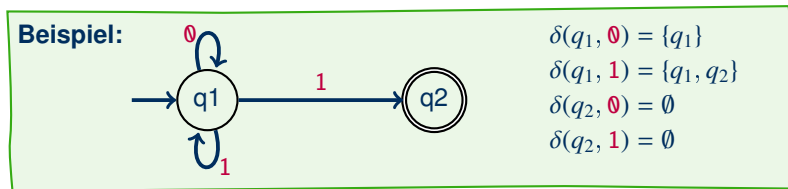
Nichtdeterministische Automaten

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 $q' \in \delta(q, a)$ auch $q \xrightarrow{a} q'$.

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

\rightsquigarrow 011 wird nichtdeterministisch akzeptiert

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 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“

↪ 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 möglichen Ausgangspunkt für die **Entwicklung deterministischer Algorithmen**

Die Sprache eines NFA

Läufe eines NFA

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

- $q_0 \in Q_0$
- $q_{i+1} \in \delta(q_i, \sigma_{i+1})$ 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**.

↪ Ein DFA hat genau einen Lauf für jedes Wort.

Er akzeptiert wenn dieser Lauf akzeptierend ist.

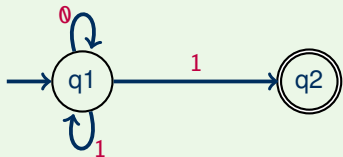
↪ Ein NFA kann für ein Wort mehrere Läufe haben.

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:



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

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

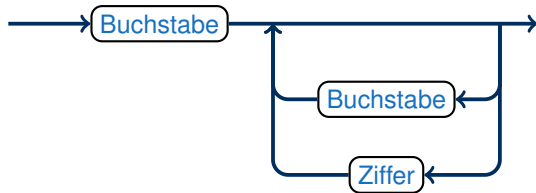
$$\delta(q_2, 0) = \emptyset$$

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

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)

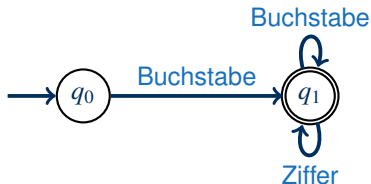
$$L(\mathcal{M}) = \{0, 1\}^* \circ \{1\}$$

NFA zur Darstellung von Syntaxdiagrammen



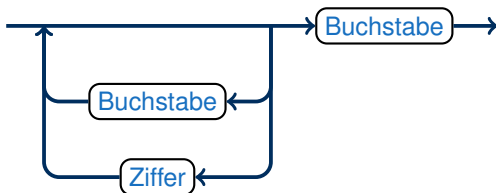
Übersetzung in NFA:

- zusammenhängende Linienbereiche werden Zustände
- Knoten werden Übergänge

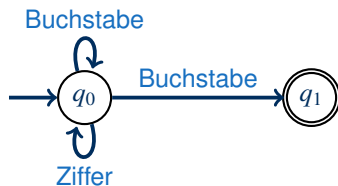


Syntaxdiagramme und Nichtdeterminismus

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 \mathbf{a} sei

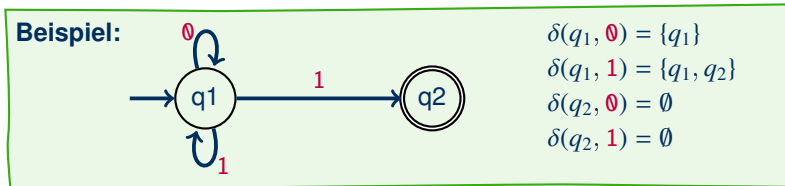
$$\delta(R, \mathbf{a}) = \bigcup_{q \in R} \delta(q, \mathbf{a}).$$

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:

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

Beispiel



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

Dann gilt:

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

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

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

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

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 „ \Rightarrow “: Angenommen es gibt einen akzeptierenden Lauf $q_0 \dots q_n$ 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 „ \Leftarrow “: Angenommen $\delta(Q_0, w) \cap F \neq \emptyset$.

- Wir ermitteln einen akzeptierenden Lauf $q_0 \dots q_n$ für w
- Dazu gehen wir rückwärts vor:
 - Wähle $q_n \in F \cap \delta(Q_0, w)$
 - Für alle $i = n, \dots, 1$:
 - Wähle $q_{i-1} \in \delta(Q_0, \sigma_1 \cdots \sigma_{i-1})$, 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 einem NFA akzeptiert.

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

Die Umkehrung dieses Satzes gilt allerdings auch:

Satz: Jede von einem NFA akzeptierbare Sprache wird auch von einem 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:

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

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

Dieser Übergang zwischen Mengen möglicher Zustände ist an sich deterministisch.

↪ wir können einen NFA deterministisch simulieren, indem wir die Menge der möglichen Zustände berechnen

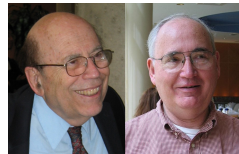
Die Potenzmengenkonstruktion

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

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

(Beweis später)

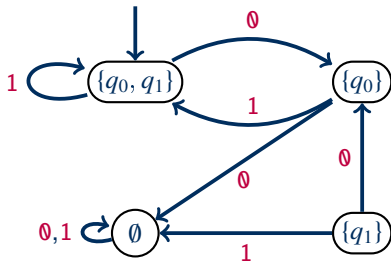
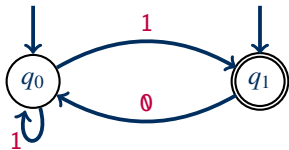


Michael Oser Rabin

Dana Scott

Andrej Bauer, CC-BY-SA 2.5

Beispiel Potenzmengenkonstruktion



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

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

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

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

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

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

$$\delta_{\text{DFA}}(\emptyset, 0) = \emptyset$$

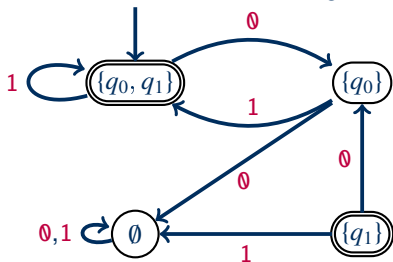
$$\delta_{\text{DFA}}(\emptyset, 1) = \emptyset$$

Erkannte Sprache:

$$\{1\}^* \circ (\{0\} \circ \{1\}^+)^*$$

Vereinfachung Potenzmengenkonstruktion

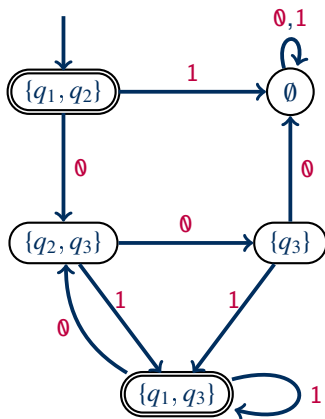
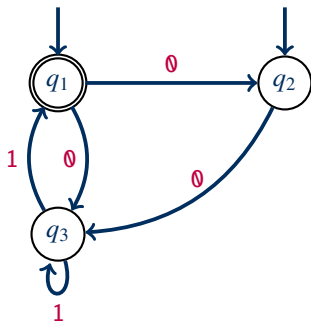
Der Automat aus dem vorherigen Beispiel kann vereinfacht werden:



- 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): $L(\mathcal{M}) = L(\mathcal{M}_{\text{DFA}})$

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

Induktionsanfang:

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

$$(2) \quad \delta_{\text{DFA}}(R, \mathbf{a}) = \bigcup_{q \in R} \delta(q, \mathbf{a}) = \delta(R, \mathbf{a})$$

Induktionshypothese: $\delta_{\text{DFA}}(R, v) = \delta(R, v)$ für Wörter v der Länge ℓ

Induktionsschritt: wir zeigen $\delta_{\text{DFA}}(R, \mathbf{a}v) = \delta(R, \mathbf{a}v)$ ein beliebiges Wort $\mathbf{a}v$ der Länge $\ell + 1$

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

Potenzmengenkonstruktion: Korrektheit (2)

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

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

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

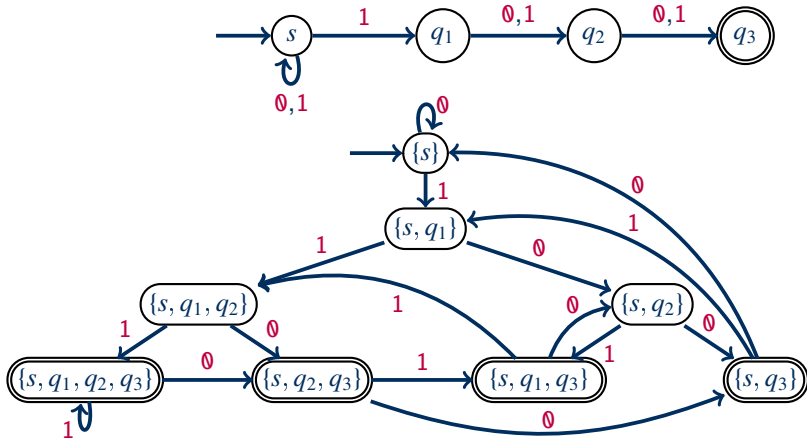
- $w \in \mathbf{L}(\mathcal{M})$ gdw. $\delta(Q_0, w) \cap F \neq \emptyset$
- gdw. $\delta_{\text{DFA}}(Q_0, w) \cap F \neq \emptyset$
- gdw. $\delta_{\text{DFA}}(Q_0, w) \in F_{\text{DFA}}$
- gdw. $w \in \mathbf{L}(\mathcal{M}_{\text{DFA}})$

□

Größenvergleich

Der DFA eines NFA hat $2^{|\mathcal{Q}|}$ – 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 $L_n = \{0, 1\}^* 1 \{0, 1\}^{n-1}$ betrachten („Wörter mit 1 an n -letzter Stelle“)

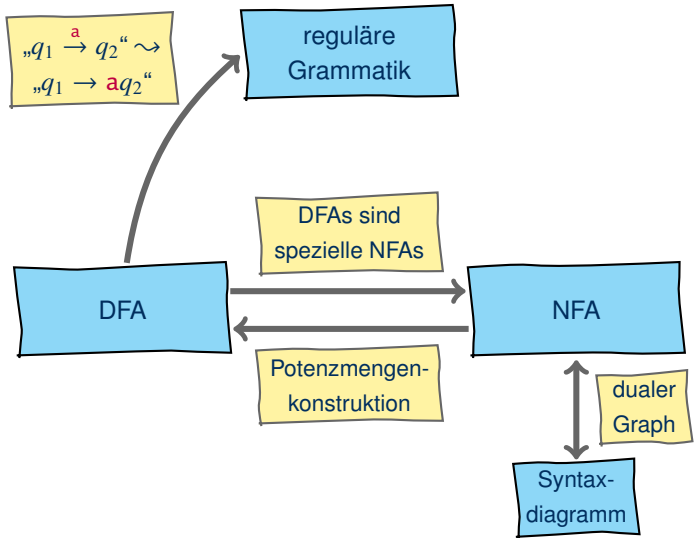
Es gilt:

- Es gibt einen NFA mit $n + 1$ Zuständen, der L_n erkennt.
- Jeder DFA, der L_n erkennt, hat mindestens 2^n Zustände.

Schlussfolgerung:

NFAs können exponentiell kompakter sein als äquivalente DFAs.

Darstellungen von Typ-3-Sprachen



Von regulären Grammatiken zu NFAs

Satz: Die Klasse der Sprachen, die durch DFAs oder NFAs erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Wir können nun die noch fehlende Richtung dieser Behauptung zeigen:
Für jede reguläre Grammatik G gibt es einen NFA \mathcal{M}_G , welcher die selbe Sprache akzeptiert (d.h., $\mathbf{L}(G) = \mathbf{L}(\mathcal{M}_G)$).

Für $G = \langle V, \Sigma, P, S \rangle$ ergibt sich $\mathcal{M}_G = \langle Q, \Sigma, \delta, Q_0, F \rangle$ wie folgt:

- $Q := V \cup \{q_f\}$
- $Q_0 := \{S\}$
- $F := \{q_f\} \cup \{A \in V \mid A \rightarrow \epsilon \in P\}$
- $\delta(A, c) := \{B \mid A \rightarrow cB \in P\} \cup \{q_f \mid A \rightarrow c \in P\}$

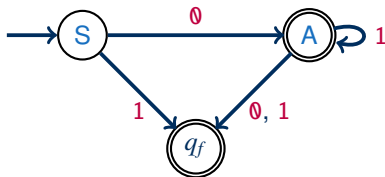
Beispiel

Wir betrachten eine reguläre Grammatik mit den folgenden sechs Regeln:

$$S \rightarrow 1 \mid 0A$$

$$A \rightarrow 0 \mid 1 \mid 1A \mid \epsilon$$

Entsprechender NFA:



Dargestellte Sprache: $\{1\} \cup (\{0\} \circ \{1\}^* \circ \{\epsilon, 0\})$

Korrektheit

Satz: Die Klasse der Sprachen, die durch DFAs oder NFAs erkannt werden können, ist genau die Klasse der regulären Sprachen.

Beweis: Wir behaupten, dass $\mathbf{L}(G) = \mathbf{L}(\mathcal{M}_G)$, d.h. für jedes Wort $w \in \Sigma^*$ soll gelten:
 $w \in \mathbf{L}(G)$ gdw. $w \in \mathbf{L}(\mathcal{M}_G)$.

Der Sonderfall $w = \epsilon$ ist ziemlich einfach:

$\epsilon \in \mathbf{L}(G)$ gdw. $S \rightarrow \epsilon \in P$

gdw. $S \in F$

gdw. $\epsilon \in \mathbf{L}(\mathcal{M}_G)$

$$\mathbf{L}(G) \subseteq \mathbf{L}(\mathcal{M}_G)$$

Wir zeigen noch $w \in \mathbf{L}(G)$ gdw. $w \in \mathbf{L}(\mathcal{M}_G)$ für den Fall $|w| \geq 1$.

„ \Rightarrow “ Angenommen $w \in \mathbf{L}(G)$ mit $w = a_1 \cdots a_n$ und $n \geq 1$.

Es gibt zwei mögliche Herleitungen für w :

$$(1) S \Rightarrow a_1 B_1 \Rightarrow \dots \Rightarrow a_1 \cdots a_{n-1} B_{n-1} \Rightarrow a_1 \cdots a_{n-1} a_n$$

$$(2) S \Rightarrow a_1 B_1 \Rightarrow \dots \Rightarrow a_1 \cdots a_{n-1} B_{n-1} \Rightarrow a_1 \cdots a_{n-1} a_n B_n \Rightarrow a_1 \cdots a_n$$

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

$$S \rightarrow a_1 B_1 \qquad B_1 \rightarrow a_2 B_2 \qquad \dots \qquad B_{n-1} \rightarrow a_n$$

Also hat \mathcal{M}_G die folgenden Übergänge:

$$S \xrightarrow{a_1} B_1 \qquad B_1 \xrightarrow{a_2} B_2 \qquad \dots \qquad B_{n-1} \xrightarrow{a_n} q_f$$

Also ist $S B_1 B_2 \dots B_{n-1} q_f$ ein akzeptierender Lauf von \mathcal{M}_G und \mathcal{M}_G akzeptiert das Wort w .

Fall (2) ist ähnlich, wobei der Lauf auf B_n endet und $B_n \in F$.

$$\mathbf{L}(G) \supseteq \mathbf{L}(\mathcal{M}_G)$$

Wir zeigen noch $w \in \mathbf{L}(G)$ gdw. $w \in \mathbf{L}(\mathcal{M}_G)$ für den Fall $|w| \geq 1$.

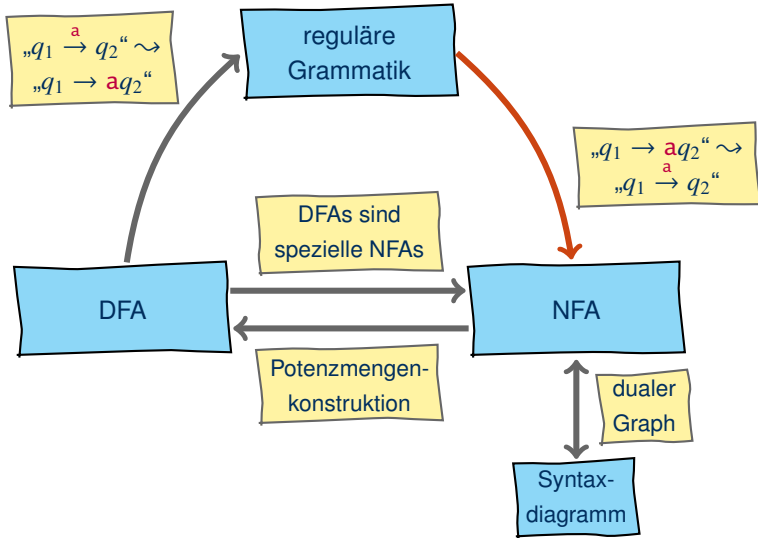
„ \Leftarrow “ Angenommen $w \in \mathbf{L}(\mathcal{M}_G)$ mit $w = a_1 \cdots a_n$ und $n \geq 1$.

Beweis analog zur vorangegangenen Richtung; grob skizziert:

- w hat einen akzeptierenden Lauf in \mathcal{M}_G
- wir betrachten die möglichen Formen solcher Läufe
- in jedem Fall finden wir entsprechende NFA-Übergänge
- daraus ergeben sich geeignete Grammatikregeln, um w abzuleiten

□

Darstellungen von Typ-3-Sprachen



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)

Und das sind zudem genau die **regulären Sprachen**
(Grammatik \leftrightarrow NFA)

Offene Fragen:

- Gibt es noch mehr Darstellungsformen für reguläre Sprachen?
- Was kommt heraus, wenn man Operationen auf reguläre Sprache anwendet?
- Wir haben gesehen, dass man Automaten manchmal vereinfachen kann – geht das noch besser?