Theorie der Informatik

7. Reguläre Sprachen I

Malte Helmert Gabriele Röger

Universität Basel

19. März 2014

Reguläre Grammatiken

Wiederholung: Reguläre Grammatiken

Definition (Reguläre Grammatik)

Eine reguläre Grammatik ist ein 4-Tupel (Σ, V, P, S) mit

1. Σ endliches Terminalalphabet,
2. V endliche Menge von Variablen (mit $V \cap \Sigma = \emptyset$),
3. $P \subseteq (V \times (\Sigma \cup \Sigma V)) \cup \{(S, \varepsilon)\}$ endliche Menge von Regeln,
4. falls $S \to \varepsilon \in P$, gibt es kein $X \in V$ mit $X \to S \in P$, und
5. $S \in V$ Startvariable.

Wiederholung: Reguläre Grammatiken

Definition (Reguläre Grammatik)

Eine reguläre Grammatik ist ein 4-Tupel (Σ, V, P, S) mit

1. Σ endliches Terminalalphabet,
2. V endliche Menge von Variablen (mit $V \cap \Sigma = \emptyset$),
3. $P \subseteq (V \times (\Sigma \cup \Sigma V)) \cup \{(S, \varepsilon)\}$ endliche Menge von Regeln,
4. falls $S \to \varepsilon \in P$, gibt es kein $X \in V$ mit $X \to S \in P$, und
5. $S \in V$ Startvariable.

Regel $X \to \varepsilon$ ist nur erlaubt, wenn $X = S$ und
S in keiner rechten Regelseite vorkommt.

Wiederholung: Reguläre Grammatiken

Definition (Reguläre Grammatik)

Eine reguläre Grammatik ist ein 4-Tupel (Σ, V, P, S) mit

1. Σ endliches Terminalalphabet,
2. V endliche Menge von Variablen (mit $V \cap \Sigma = \emptyset$),
3. $P \subseteq (V \times (\Sigma \cup \Sigma V)) \cup \{(S, \varepsilon)\}$ endliche Menge von Regeln,
4. falls $S \to \varepsilon \in P$, gibt es kein $X \in V$ mit $X \to S \in P$, und
5. $S \in V$ Startvariable.

Regel $X \to \varepsilon$ ist nur erlaubt, wenn $X = S$ und S in keiner rechten Regelseite vorkommt.

Wie restriktiv ist das?

Epsilon-Regeln

Satz

Für jede Grammatik G mit Regeln $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ gibt es eine reguläre Grammatik G' mit $\mathcal{L}(G) = \mathcal{L}(G')$.

Epsilon-Regeln

Satz

Für jede Grammatik G mit Regeln $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ gibt es eine reguläre Grammatik G' mit $\mathcal{L}(G) = \mathcal{L}(G')$.

Beweis.

Sei $G = (\Sigma, V, P, S)$ Grammatik mit $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$.
Sei $V_\varepsilon = \{A \in V \mid A \Rightarrow^* \varepsilon\}$. Diese Menge V_ε besteht genau aus den Variablen A mit $A \to \varepsilon \in P$.

...

Epsilon-Regeln

Satz

Für jede Grammatik G mit Regeln $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ gibt es eine reguläre Grammatik G' mit $\mathcal{L}(G) = \mathcal{L}(G')$.

Beweis.

Sei $G = (\Sigma, V, P, S)$ Grammatik mit $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$. Sei $V_\varepsilon = \{A \in V \mid A \Rightarrow^* \varepsilon\}$. Diese Menge V_ε besteht genau aus den Variablen A mit $A \to \varepsilon \in P$.

Sei P' die Regelmenge, die aus P hervorgeht, indem man alle Regeln der Form $A \to \varepsilon$ $(A \neq S)$ entfernt. Zudem fügt man für jede Regel der Form $B \to xA$ mit $A \in V_\varepsilon, B \in V, x \in \Sigma$ eine Regel $B \to x$ zu P' hinzu. ...

Epsilon-Regeln

Satz

Für jede Grammatik G mit Regeln $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ gibt es eine reguläre Grammatik G' mit $\mathcal{L}(G) = \mathcal{L}(G')$.

Beweis (Fortsetzung).

Damit ist $\mathcal{L}(G) = \mathcal{L}((\Sigma, V, P', S))$ und P' enthält keine Regel $A \to \varepsilon$ mit $A \neq S$. Ist $S \to \varepsilon \notin P$, sind wir fertig.

Epsilon-Regeln

Satz

Für jede Grammatik G mit Regeln $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ gibt es eine reguläre Grammatik G' mit $\mathcal{L}(G) = \mathcal{L}(G')$.

Beweis (Fortsetzung).

Damit ist $\mathcal{L}(G) = \mathcal{L}((\Sigma, V, P', S))$ und P' enthält keine Regel $A \to \varepsilon$ mit $A \neq S$. Ist $S \to \varepsilon \notin P$, sind wir fertig.

Epsilon-Regeln

Satz

Für jede Grammatik G mit Regeln $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ gibt es eine reguläre Grammatik G' mit $\mathcal{L}(G) = \mathcal{L}(G')$.

Beweis (Fortsetzung).

Damit ist $\mathcal{L}(G) = \mathcal{L}((\Sigma, V, P', S))$ und P' enthält keine Regel $A \to \varepsilon$ mit $A \neq S$. Ist $S \to \varepsilon \notin P$, sind wir fertig.

Sonst sei S' neue Variable und P'' gehe aus P' hervor, indem man

1. Regeln $X \to aS$ mit $X \in V, a \in \Sigma$ durch $X \to aS'$ ersetzt,
2. für jede Regel $S \to aX$ mit $X \in V, a \in \Sigma$ die Regel $S' \to aX$ hinzufügt, und
3. für jede Regel $S \to a$ mit $a \in \Sigma$ die Regel $S' \to a$ hinzufügt.

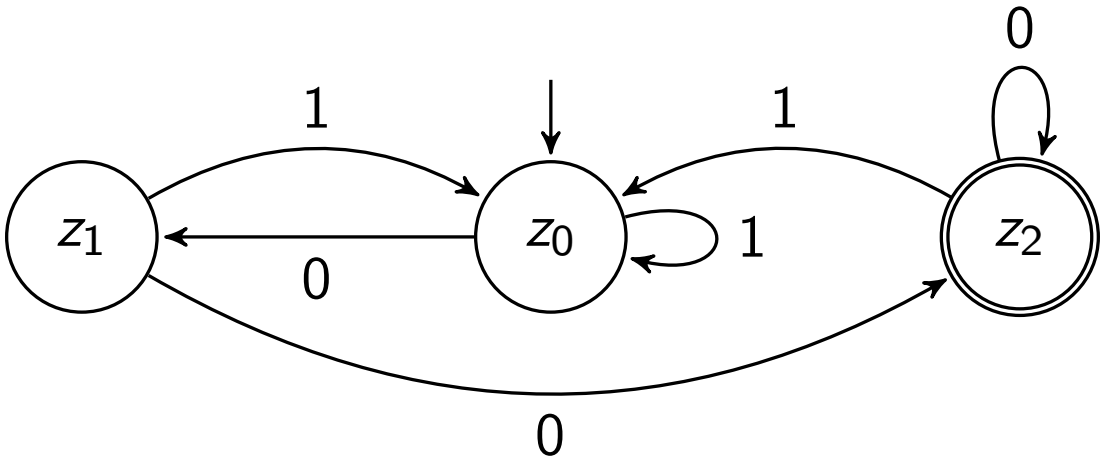
Dann ist $\mathcal{L}(G) = \mathcal{L}((\Sigma, V \cup \{S'\}, P'', S))$. □

Fragen

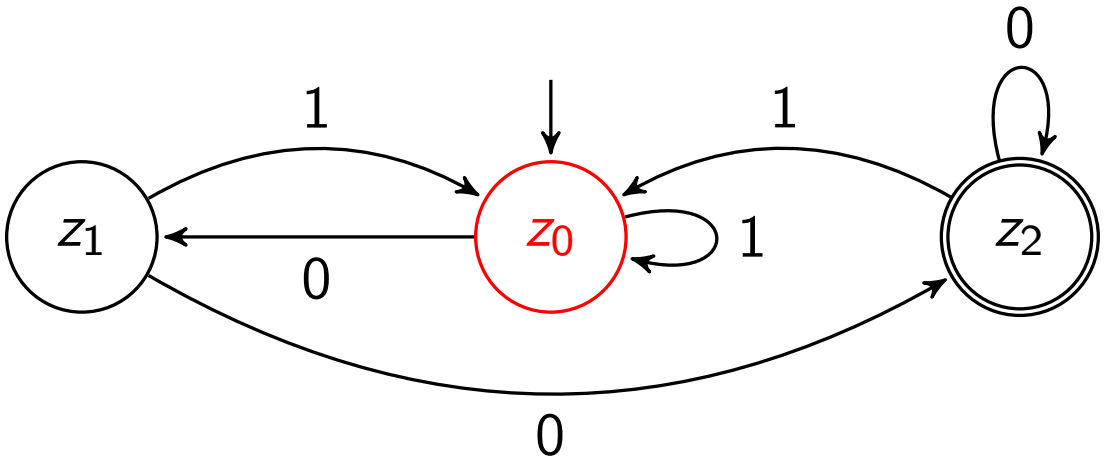
Fragen?

DFAs

Endliche Automaten: Beispiel

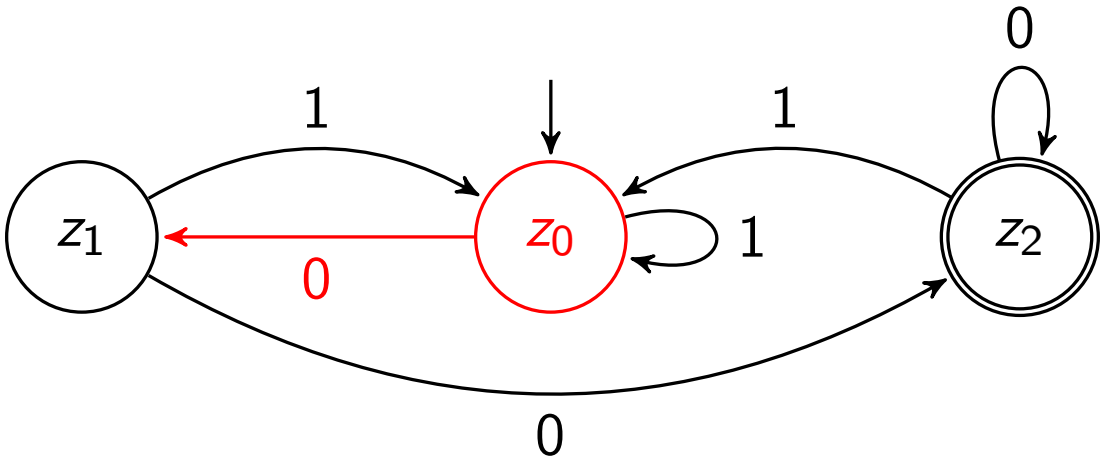


Endliche Automaten: Beispiel



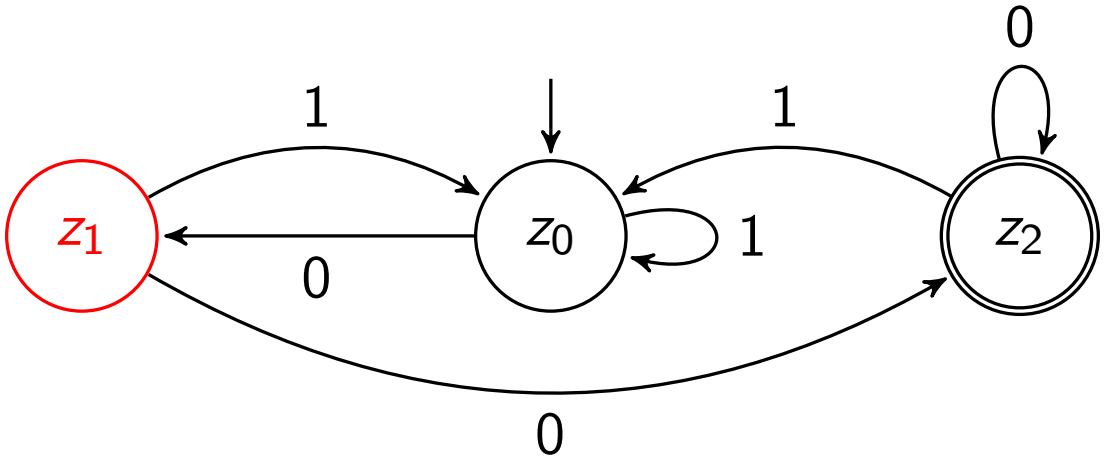
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0,

Endliche Automaten: Beispiel



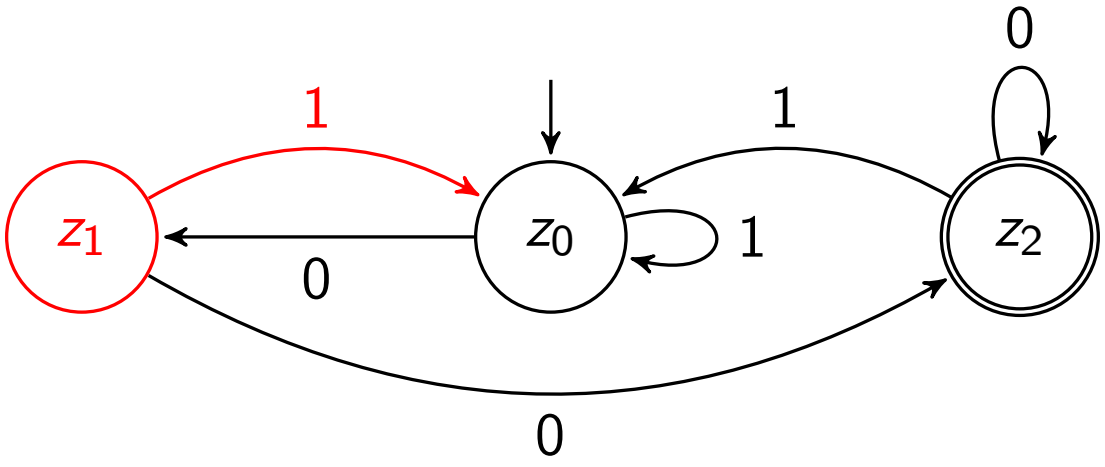
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0,

Endliche Automaten: Beispiel



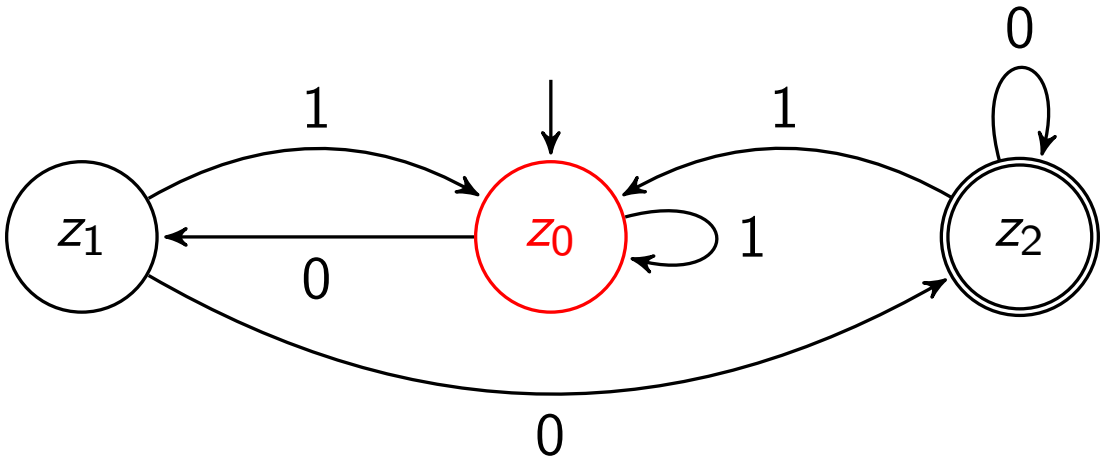
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1,

Endliche Automaten: Beispiel



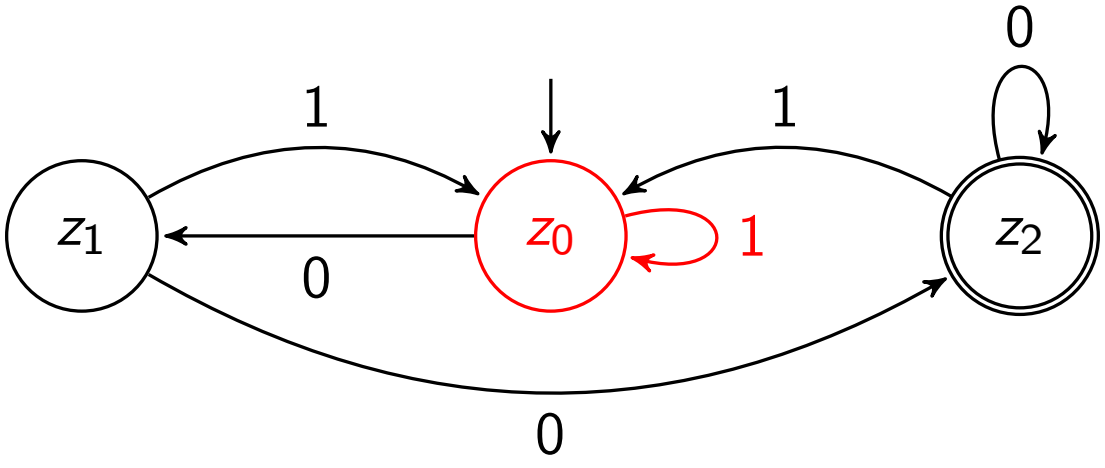
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1,

Endliche Automaten: Beispiel



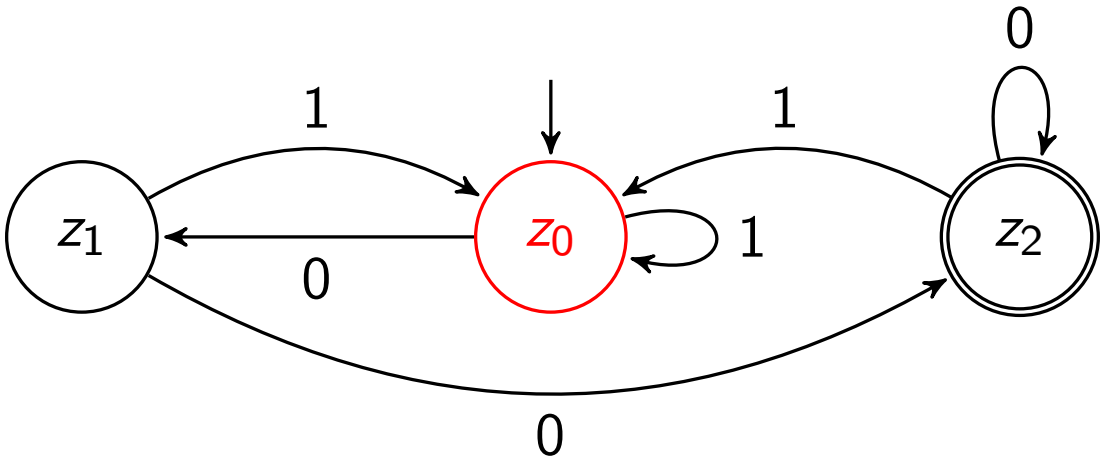
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1, z_0,

Endliche Automaten: Beispiel



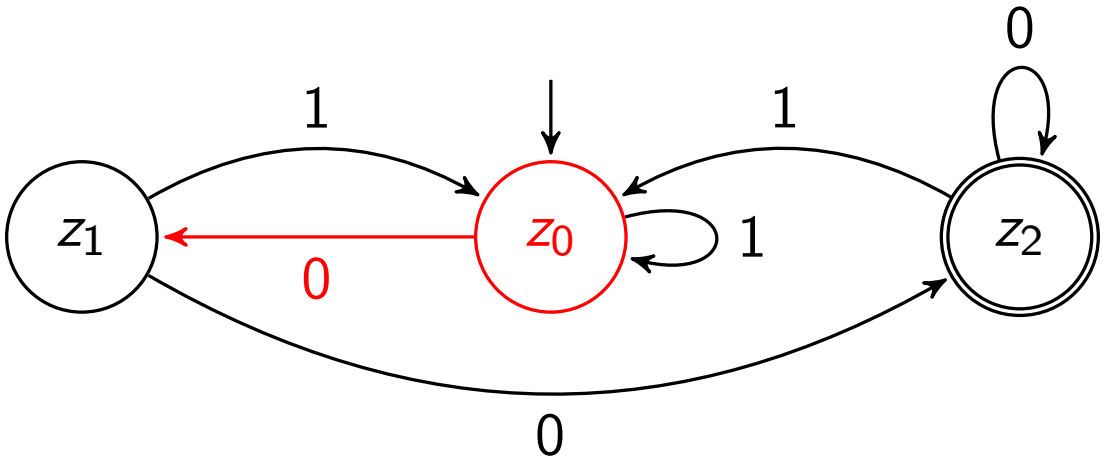
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1, z_0,

Endliche Automaten: Beispiel



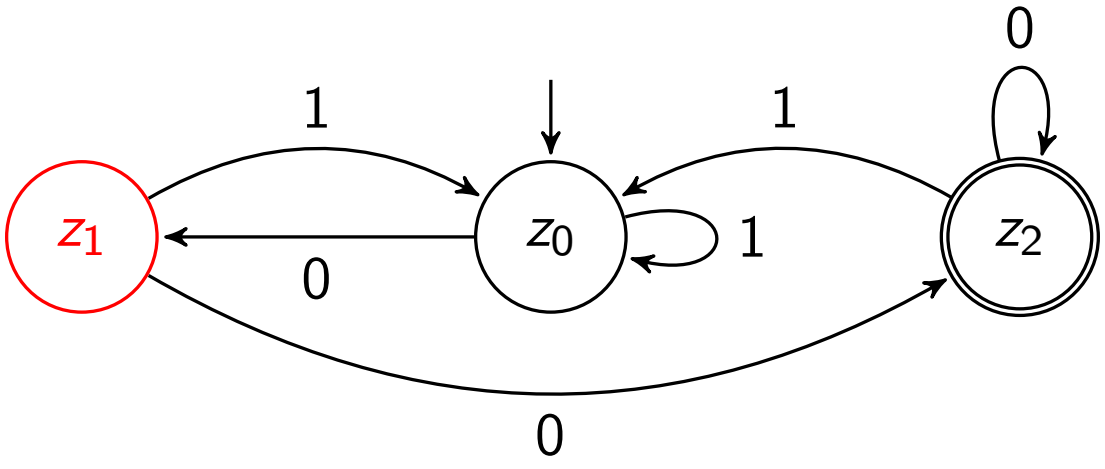
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1, z_0, z_0,

Endliche Automaten: Beispiel



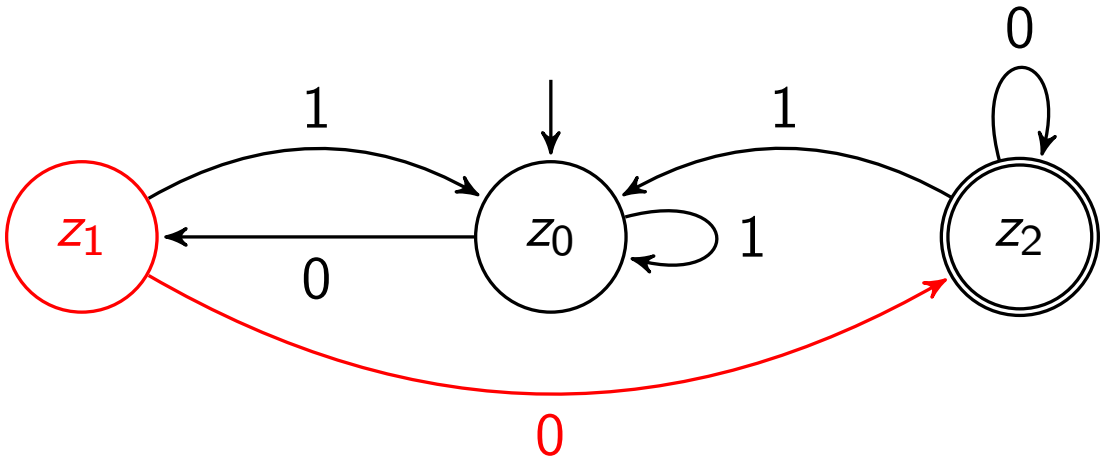
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1, z_0, z_0,

Endliche Automaten: Beispiel



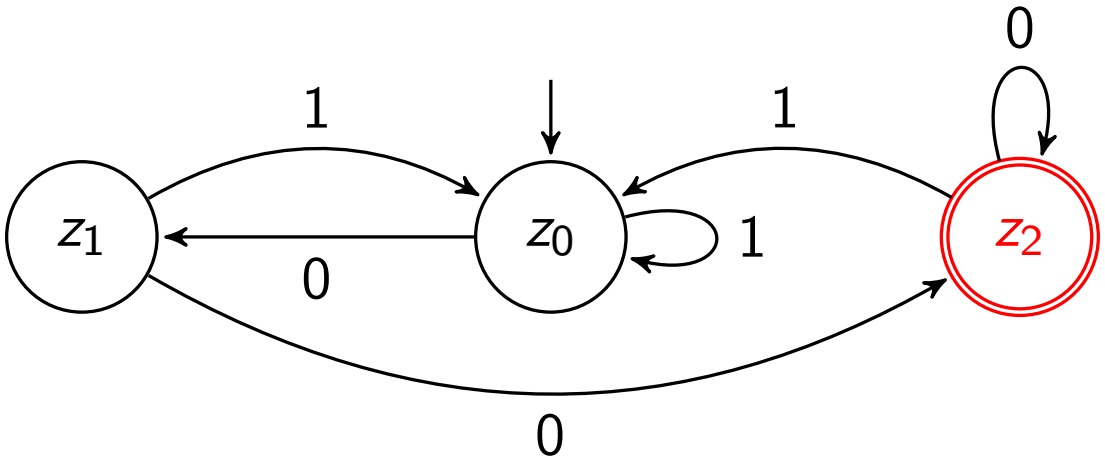
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1, z_0, z_0, z_1,

Endliche Automaten: Beispiel



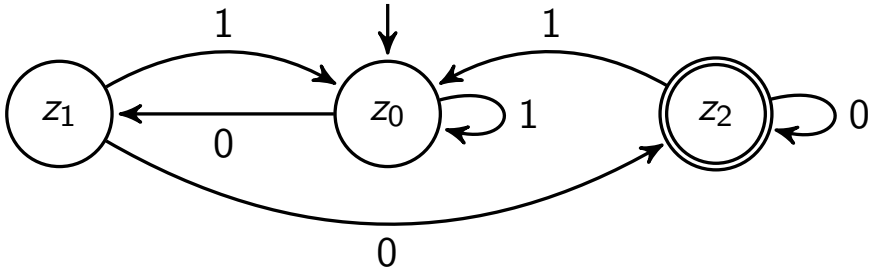
Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1, z_0, z_0, z_1,

Endliche Automaten: Beispiel

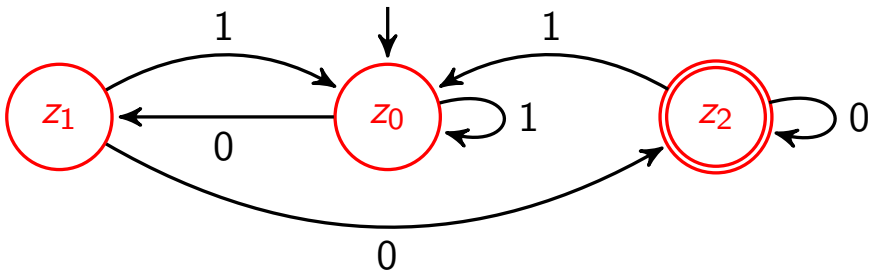


Beim Lesen der Eingabe 01100 durchläuft der Automat die Zustände z_0, z_1, z_0, z_0, z_1, z_2.

Endliche Automaten: Terminologie und Notation

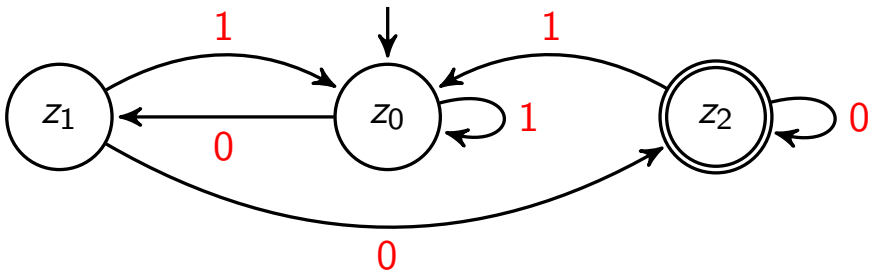


Endliche Automaten: Terminologie und Notation



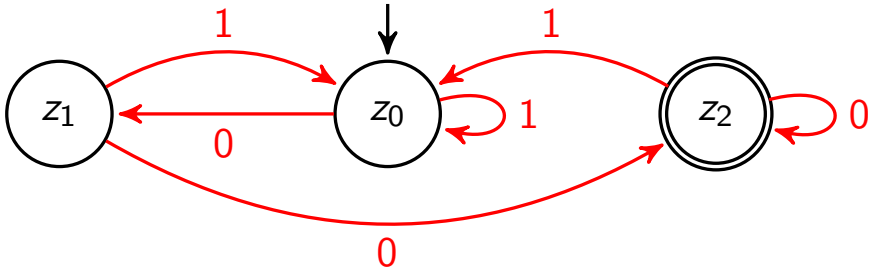
- Zustände
  $Z = \{z_0, z_1, z_2\}$

Endliche Automaten: Terminologie und Notation



- Zustände
  $Z = \{z_0, z_1, z_2\}$
- Eingabealphabet
  $\Sigma = \{0, 1\}$

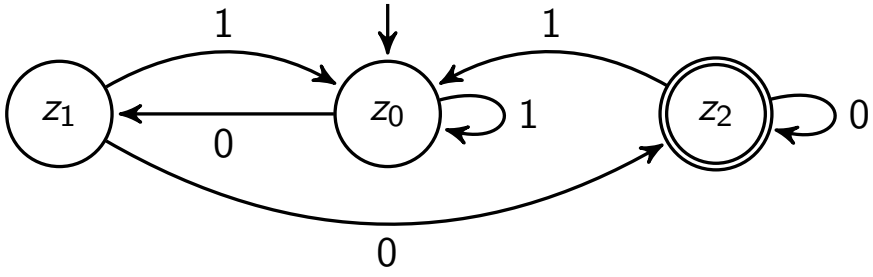
Endliche Automaten: Terminologie und Notation



- Zustände
  $Z = \{z_0, z_1, z_2\}$
- Eingabealphabet
  $\Sigma = \{0, 1\}$
- Überführungsfunktion δ

$$\delta(z_0, 0) = z_1$$
$$\delta(z_0, 1) = z_0$$
$$\delta(z_1, 0) = z_2$$
$$\delta(z_1, 1) = z_0$$
$$\delta(z_2, 0) = z_2$$
$$\delta(z_2, 1) = z_0$$

Endliche Automaten: Terminologie und Notation



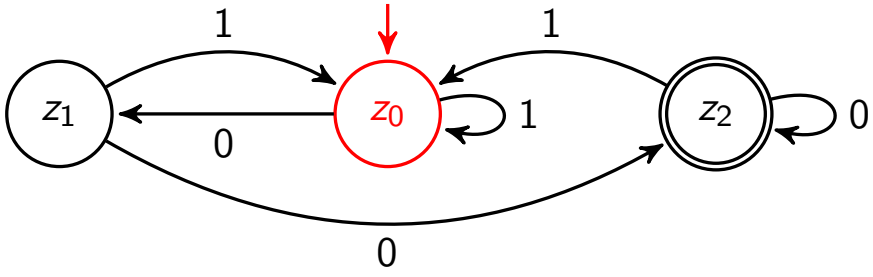
- Zustände
  $Z = \{z_0, z_1, z_2\}$
- Eingabealphabet
  $\Sigma = \{0, 1\}$
- Überführungsfunktion δ

$$\delta(z_0, 0) = z_1$$
$$\delta(z_0, 1) = z_0$$
$$\delta(z_1, 0) = z_2$$
$$\delta(z_1, 1) = z_0$$
$$\delta(z_2, 0) = z_2$$
$$\delta(z_2, 1) = z_0$$

δ	0	1
z_0	z_1	z_0
z_1	z_2	z_0
z_2	z_2	z_0

Alternative Schreibweise von δ

Endliche Automaten: Terminologie und Notation



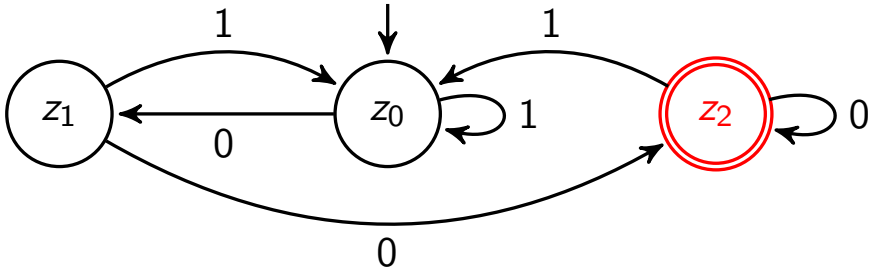
- Zustände
  $Z = \{z_0, z_1, z_2\}$
- Eingabealphabet
  $\Sigma = \{0, 1\}$
- Überführungsfunktion δ
- Startzustand z_0

$$\delta(z_0, 0) = z_1$$
$$\delta(z_0, 1) = z_0$$
$$\delta(z_1, 0) = z_2$$
$$\delta(z_1, 1) = z_0$$
$$\delta(z_2, 0) = z_2$$
$$\delta(z_2, 1) = z_0$$

δ	0	1
z_0	z_1	z_0
z_1	z_2	z_0
z_2	z_2	z_0

Alternative Schreibweise von δ

Endliche Automaten: Terminologie und Notation



- Zustände
  $Z = \{z_0, z_1, z_2\}$
- Eingabealphabet
  $\Sigma = \{0, 1\}$
- Überführungsfunktion δ
- Startzustand z_0
- Endzustände $\{z_2\}$

$$\delta(z_0, 0) = z_1$$
$$\delta(z_0, 1) = z_0$$
$$\delta(z_1, 0) = z_2$$
$$\delta(z_1, 1) = z_0$$
$$\delta(z_2, 0) = z_2$$
$$\delta(z_2, 1) = z_0$$

δ	0	1
z_0	z_1	z_0
z_1	z_2	z_0
z_2	z_2	z_0

Alternative Schreibweise von δ

Deterministischer endlicher Automat: Definition

Definition (Deterministischer endlicher Automat (DFA))

Ein (deterministischer) endlicher Automat (deterministic finite automaton, DFA) ist ein 5-Tupel $M = (Z, \Sigma, \delta, z_0, E)$ mit

- Z die endlichen Menge von Zustände,
- Σ das endliche Eingabealphabet (wobei $Z \cap \Sigma = \emptyset$),
- $\delta : Z \times \Sigma \to Z$ die Überführungsfunktion,
- $z_0 \in Z$ der Startzustand,
- $E \subseteq Z$ die Menge der Endzustände.

DFA: Erkannte Wörter

Definition (erkanntes Wort bei DFAs)

DFA $M = (Z, \Sigma, \delta, z_0, E)$ erkennt das Wort $w = a_1 \dots a_n$ genau dann, wenn eine Folge von Zuständen $z'_0, \dots, z'_n \in Z$ existiert mit

1. $z'_0 = z_0$,
2. $\delta(z'_{i-1}, a_i) = z'_i$ für alle $i \in \{1, \dots, n\}$ und
3. $z'_n \in E$.

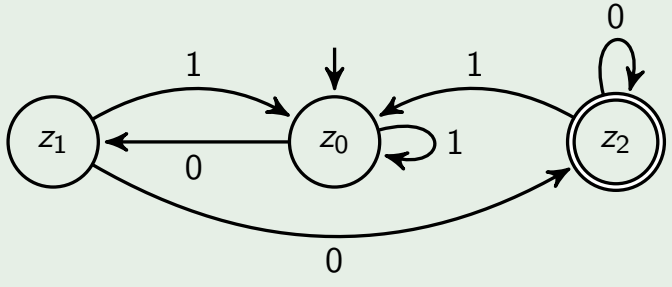
DFA: Erkannte Wörter

Definition (erkanntes Wort bei DFAs)

DFA $M = (Z, \Sigma, \delta, z_0, E)$ erkennt das Wort $w = a_1 \dots a_n$ genau dann, wenn eine Folge von Zuständen $z'_0, \dots, z'_n \in Z$ existiert mit

1. $z'_0 = z_0$,
2. $\delta(z'_{i-1}, a_i) = z'_i$ für alle $i \in \{1, \dots, n\}$ und
3. $z'_n \in E$.

Beispiel



Erkennt z.B.	Erkennt z.B. nicht
00	ε
10010100	1001010
01000	010001

DFA: Akzeptierte Sprache

Definition (akzeptierte Sprache eines DFA)

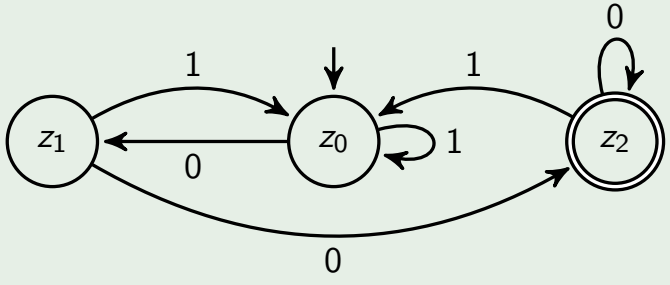
Sei M ein endlicher Automat. Die von M akzeptierte Sprache ist definiert durch $\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ wird von } M \text{ erkannt}\}$.

DFA: Akzeptierte Sprache

Definition (akzeptierte Sprache eines DFA)

Sei M ein endlicher Automat. Die von M akzeptierte Sprache ist definiert durch $\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ wird von } M \text{ erkannt}\}$.

Beispiel

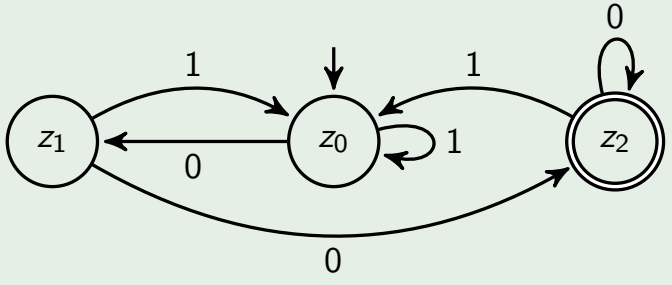


DFA: Akzeptierte Sprache

Definition (akzeptierte Sprache eines DFA)

Sei M ein endlicher Automat. Die von M akzeptierte Sprache ist definiert durch $\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ wird von } M \text{ erkannt}\}$.

Beispiel



Der DFA akzeptiert die Sprache $\{w \in \{0,1\}^* \mid w \text{ endet mit } 00\}$.

Von DFAs akzeptierte Sprachen sind regulär

Satz

Jede durch deterministische endliche Automaten akzeptierte Sprache ist regulär (Typ 3).

Von DFAs akzeptierte Sprachen sind regulär

Satz

Jede durch deterministische endliche Automaten akzeptierte Sprache ist regulär (Typ 3).

Beweis.

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA. Wir definieren eine reguläre Grammatik G, so dass $\mathcal{L}(G) = \mathcal{L}(M)$.

Definiere $G = (\Sigma, Z, P, z_0)$, wobei P für jedes $\delta(z, a) = z'$ eine Regel $z \to az'$ enthält. Zudem enthält P für jedes $z \in E$ die Regel $z \to \varepsilon$ (evtl. Umstrukturierung nötig wie für Epsilon-Regeln beschrieben). ...

Von DFAs akzeptierte Sprachen sind regulär

Satz

Jede durch deterministische endliche Automaten akzeptierte Sprache ist regulär (Typ 3).

Beweis (Fortsetzung).

Dann gilt für $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$

gdw. es eine Folge von Zuständen $z'_0, z'_1, \dots, z'_n$ gibt mit
$z'_0 = z_0$, $z'_n \in E$ und $\delta(z'_{i-1}, a_i) = z'_i$ für alle $i \in \{1, \dots, n\}$

gdw. es eine Folge von Variablen $z'_0, z'_1, \dots, z'_{n-1}$ gibt mit
z'_0 ist Startvariable und es gilt $z'_0 \Rightarrow a_1 z'_1 \Rightarrow a_1 a_2 z'_2 \Rightarrow$
$\dots \Rightarrow a_1 a_2 \dots a_{n-1} z'_{n-1} \Rightarrow a_1 a_2 \dots a_n$.

gdw. $w \in \mathcal{L}(G)$ □

Von DFAs akzeptierte Sprachen sind regulär

Satz

Jede durch deterministische endliche Automaten akzeptierte Sprache ist regulär (Typ 3).

Beweis (Fortsetzung).

Dann gilt für $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$

gdw. es eine Folge von Zuständen $z'_0, z'_1, \dots, z'_n$ gibt mit
$z'_0 = z_0$, $z'_n \in E$ und $\delta(z'_{i-1}, a_i) = z'_i$ für alle $i \in \{1, \dots, n\}$

gdw. es eine Folge von Variablen $z'_0, z'_1, \dots, z'_{n-1}$ gibt mit
z'_0 ist Startvariable und es gilt $z'_0 \Rightarrow a_1 z'_1 \Rightarrow a_1 a_2 z'_2 \Rightarrow$
$\dots \Rightarrow a_1 a_2 \dots a_{n-1} z'_{n-1} \Rightarrow a_1 a_2 \dots a_n$.

gdw. $w \in \mathcal{L}(G)$ □

Beispiel: Tafel

Frage

Gibt es umgekehrt für alle regulären Sprachen einen DFA, der sie erkennt? Sind also die von DFAs erkannten Sprachen genau die regulären Sprachen?

Frage

Gibt es umgekehrt für alle regulären Sprachen einen DFA, der sie erkennt? Sind also die von DFAs erkannten Sprachen genau die regulären Sprachen?

Ja!

Wir werden das später auch noch (indirekt) beweisen.

Fragen

Fragen?

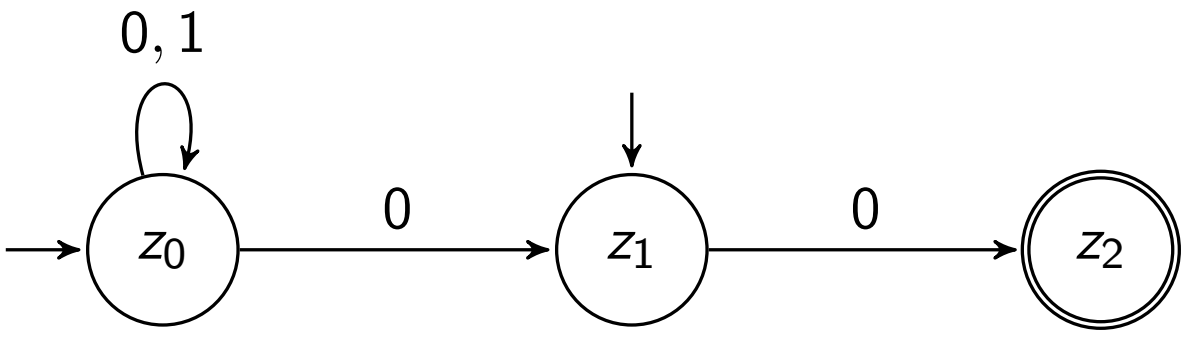
Quiz

NFAs

Nichtdeterministische Automaten

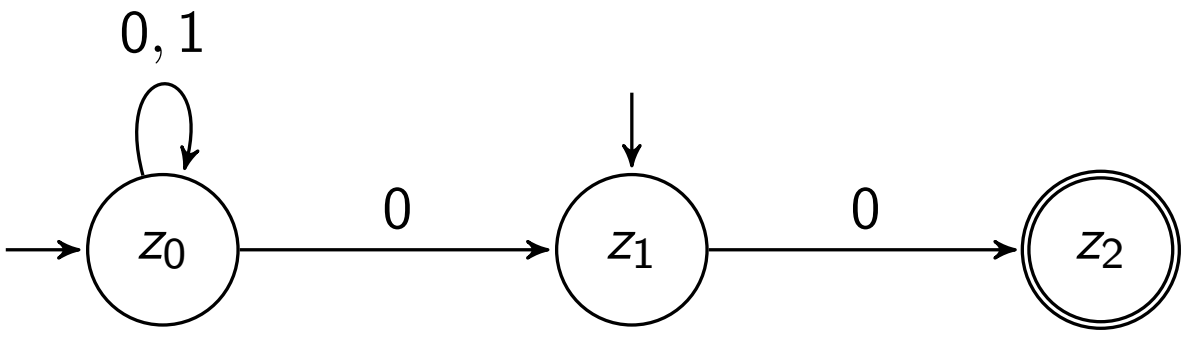
Wieso heissen DFAs deterministische Automaten? Was sind dann nichtdeterministische Automaten?

Nichtdeterministische Automaten: Beispiel



Unterschiede zu DFAs:

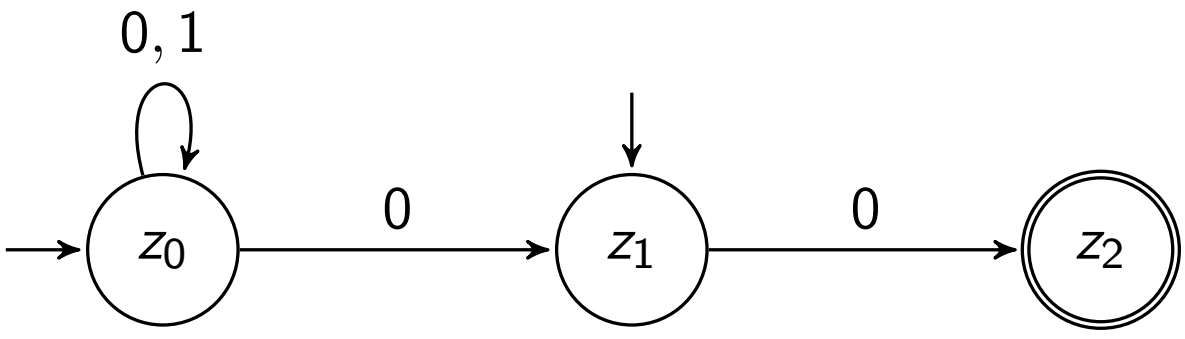
Nichtdeterministische Automaten: Beispiel



Unterschiede zu DFAs:

- mehrere Startzustände möglich

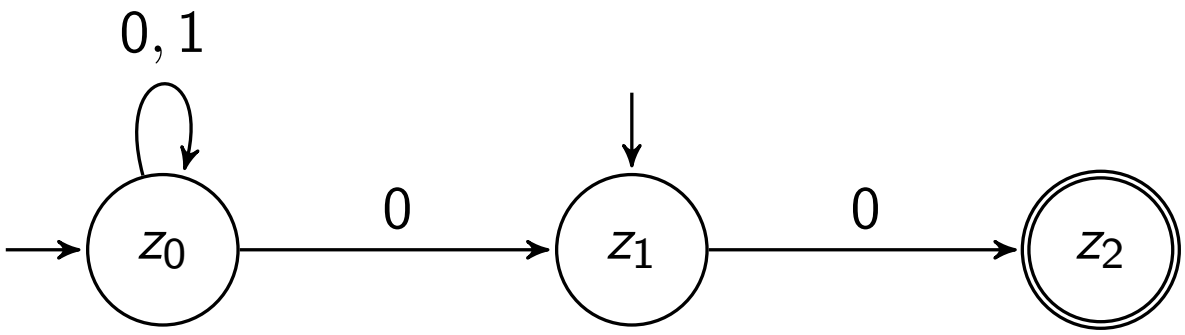
Nichtdeterministische Automaten: Beispiel



Unterschiede zu DFAs:

- mehrere Startzustände möglich
- Überführungsfunktion δ kann für ein $a \in \Sigma$ in keine oder mehrere Nachfolgezustände führen.

Nichtdeterministische Automaten: Beispiel



Unterschiede zu DFAs:

- mehrere Startzustände möglich
- Überführungsfunktion δ kann für ein $a \in \Sigma$ in keine oder mehrere Nachfolgezustände führen.
- Automat erkennt ein Wort, wenn es mindestens eine akzeptierende Zustandsfolge gibt.

Nichtdeterministischer endlicher Automat: Definition

Definition (nichtdeterministischer endlicher Automat = NFA)

Ein nichtdeterministischer endlicher Automat (nondeterministic finite automaton, NFA) ist ein 5-Tupel $M = (Z, \Sigma, \delta, S, E)$ mit

- Z die Menge der Zustände,
- Σ das Eingabealphabet (mit $Z \cap \Sigma = \emptyset$),
- $\delta : Z \times \Sigma \to \mathcal{P}(Z)$ die Überführungsfunktion in die Potenzmenge von Z,
- $S \subseteq Z$ die Menge der Startzustände,
- $E \subseteq Z$ die Menge der Endzustände.

Nichtdeterministischer endlicher Automat: Definition

Definition (nichtdeterministischer endlicher Automat = NFA)

Ein nichtdeterministischer endlicher Automat (nondeterministic finite automaton, NFA) ist ein 5-Tupel $M = (Z, \Sigma, \delta, S, E)$ mit

- Z die Menge der Zustände,
- Σ das Eingabealphabet (mit $Z \cap \Sigma = \emptyset$),
- $\delta : Z \times \Sigma \to \mathcal{P}(Z)$ die Überführungsfunktion in die Potenzmenge von Z,
- $S \subseteq Z$ die Menge der Startzustände,
- $E \subseteq Z$ die Menge der Endzustände.

DFAs sind Spezialfall von NFAs.

NFA: Erkannte Wörter

Definition (erkanntes Wort bei NFAs)

NFA $M = (Z, \Sigma, \delta, S, E)$ erkennt das Wort $w = a_1 \dots a_n$ genau dann, wenn eine Folge von Zuständen $z'_0, \dots, z'_n \in Z$ existiert mit

1. $z'_0 \in S$,
2. $z'_i \in \delta(z'_{i-1}, a_i)$ für alle $i \in \{1, \dots, n\}$ und
3. $z'_n \in E$.

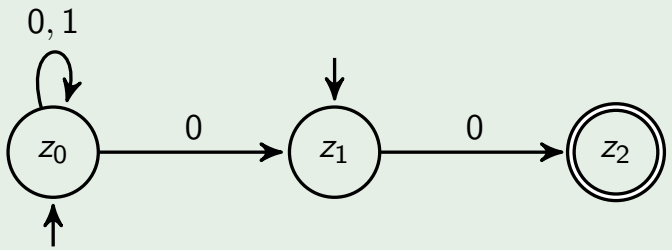
NFA: Erkannte Wörter

Definition (erkanntes Wort bei NFAs)

NFA $M = (Z, \Sigma, \delta, S, E)$ erkennt das Wort $w = a_1 \dots a_n$ genau dann, wenn eine Folge von Zuständen $z'_0, \dots, z'_n \in Z$ existiert mit

1. $z'_0 \in S$,
2. $z'_i \in \delta(z'_{i-1}, a_i)$ für alle $i \in \{1, \dots, n\}$ und
3. $z'_n \in E$.

Beispiel



Erkennt z.B.	Erkennt z.B. nicht
0	ε
10010100	1001010
01000	010001

NFA: Akzeptierte Sprache

Definition (akzeptierte Sprache eines NFA)

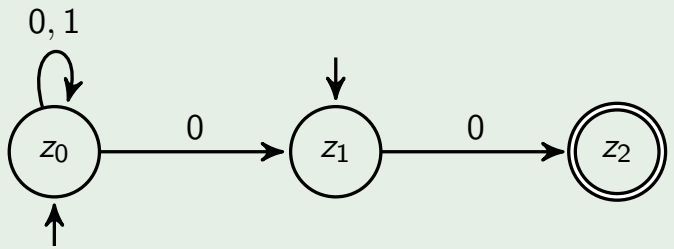
Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA. Die von M akzeptierte Sprache ist definiert durch $\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ wird von } M \text{ erkannt}\}$.

NFA: Akzeptierte Sprache

Definition (akzeptierte Sprache eines NFA)

Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA. Die von M akzeptierte Sprache ist definiert durch $\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ wird von } M \text{ erkannt}\}$.

Beispiel

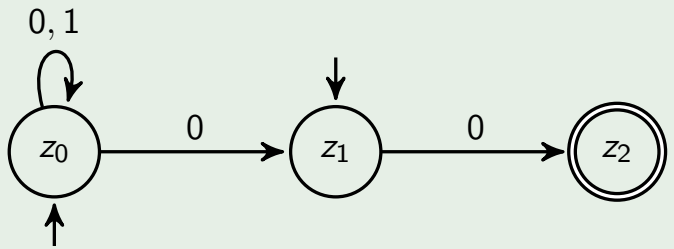


NFA: Akzeptierte Sprache

Definition (akzeptierte Sprache eines NFA)

Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA. Die von M akzeptierte Sprache ist definiert durch $\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ wird von } M \text{ erkannt}\}$.

Beispiel



Der NFA akzeptiert die Sprache
$\{w \in \{0,1\}^* \mid w = 0 \text{ oder } w \text{ endet mit } 00\}$.

Fragen

Fragen?

Quiz

NFAs nicht mächtiger als DFAs

Satz (Rabin, Scott)

Jede von einem NFA akzeptierbare Sprache ist auch durch einen DFA akzeptierbar.

NFAs nicht mächtiger als DFAs

Satz (Rabin, Scott)

Jede von einem NFA akzeptierbare Sprache ist auch durch einen DFA akzeptierbar.

Beweis.

Zu jedem NFA $M = (Z, \Sigma, \delta, S, E)$ kann man einen DFA $M' = (Z', \Sigma, \delta', z_0', E')$ bauen mit $\mathcal{L}(M) = \mathcal{L}(M')$.
Hierbei ist M' wie folgt definiert:

- $Z' := \mathcal{P}(Z)$ (die Potenzmenge von Z)
- $z_0' := S$
- $E' := \{\mathcal{Z} \subseteq Z \mid \mathcal{Z} \cap E \neq \emptyset\}$
- Für alle $\mathcal{Z} \in Z'$: $\delta'(\mathcal{Z}, a) := \bigcup_{z \in \mathcal{Z}} \delta(z, a)$

...

NFAs nicht mächtiger als DFAs

Satz (Rabin, Scott)

Jede von einem NFA akzeptierbare Sprache ist auch durch einen DFA akzeptierbar.

Beweis (Fortsetzung).

Dann gilt für $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$
gdw. es eine Folge von Zuständen $z_0, z_1, \dots, z_n$ gibt mit
$z_0 \in S$, $z_n \in E$ und $z_i \in \delta(z_{i-1}, a_i)$ für alle $i \in \{1, \dots, n\}$
gdw. es eine Folge von Teilmengen $\mathcal{Z}_0, \mathcal{Z}_1, \dots, \mathcal{Z}_n$ gibt mit
$\mathcal{Z}_0 = z_0'$, $\mathcal{Z}_n \in E'$ und $\delta'(\mathcal{Z}_{i-1}, a_i) = \mathcal{Z}_i$ f.a. $i \in \{1, \dots, n\}$
gdw. $w \in \mathcal{L}(M')$ □

NFAs nicht mächtiger als DFAs

Satz (Rabin, Scott)

Jede von einem NFA akzeptierbare Sprache ist auch durch einen DFA akzeptierbar.

Beweis (Fortsetzung).

Dann gilt für $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$
gdw. es eine Folge von Zuständen $z_0, z_1, \dots, z_n$ gibt mit
$z_0 \in S$, $z_n \in E$ und $z_i \in \delta(z_{i-1}, a_i)$ für alle $i \in \{1, \dots, n\}$
gdw. es eine Folge von Teilmengen $\mathcal{Z}_0, \mathcal{Z}_1, \dots, \mathcal{Z}_n$ gibt mit
$\mathcal{Z}_0 = z_0'$, $\mathcal{Z}_n \in E'$ und $\delta'(\mathcal{Z}_{i-1}, a_i) = \mathcal{Z}_i$ f.a. $i \in \{1, \dots, n\}$
gdw. $w \in \mathcal{L}(M')$ □

Beispiel: Tafel

NFAs kompakter als DFAs

Beispiel

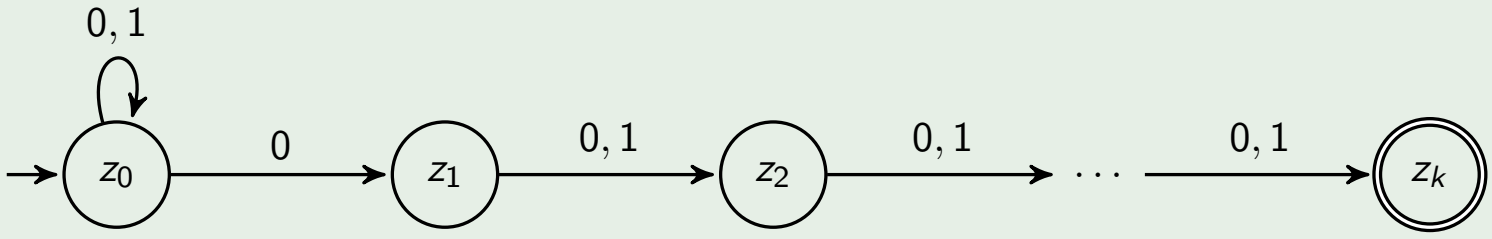
Betrachte für $k \geq 1$ die Sprache
$L_k = \{w \in \{0,1\} \mid |w| \geq k \text{ und das } k\text{-letzte Zeichen von } w \text{ ist } 0\}$.

NFAs kompakter als DFAs

Beispiel

Betrachte für $k \geq 1$ die Sprache
$L_k = \{w \in \{0,1\} \mid |w| \geq k \text{ und das } k\text{-letzte Zeichen von } w \text{ ist } 0\}$.

Die Sprache L_k kann von einem NFA mit $k+1$ Zuständen akzeptiert werden:

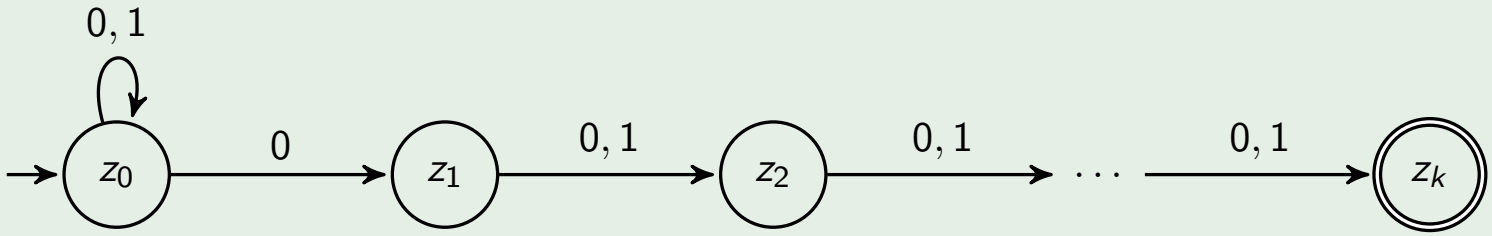


NFAs kompakter als DFAs

Beispiel

Betrachte für $k \geq 1$ die Sprache
$L_k = \{w \in \{0,1\} \mid |w| \geq k \text{ und das } k\text{-letzte Zeichen von } w \text{ ist } 0\}$.

Die Sprache L_k kann von einem NFA mit $k+1$ Zuständen akzeptiert werden:



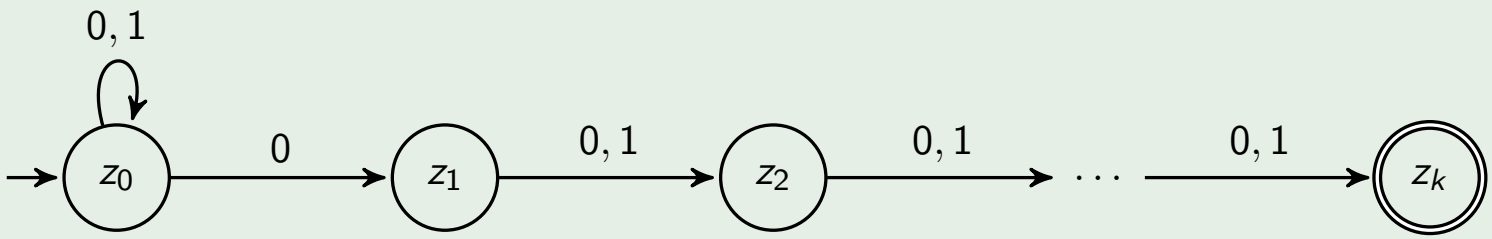
Es gibt keinen DFA mit weniger als 2^k Zuständen, der L_k akzeptiert (ohne Beweis).

NFAs kompakter als DFAs

Beispiel

Betrachte für $k \geq 1$ die Sprache
$L_k = \{w \in \{0,1\} \mid |w| \geq k \text{ und das } k\text{-letzte Zeichen von } w \text{ ist } 0\}$.

Die Sprache L_k kann von einem NFA mit $k+1$ Zuständen akzeptiert werden:



Es gibt keinen DFA mit weniger als 2^k Zuständen, der L_k akzeptiert (ohne Beweis).

NFAs können Sprachen oft viel kompakter repräsentieren als DFAs.

Reguläre Grammatiken nicht mächtiger als NFAs

Satz

Für jede reguläre Grammatik G gibt es einen NFA M mit $\mathcal{L}(G) = \mathcal{L}(M)$.

Reguläre Grammatiken nicht mächtiger als NFAs

Satz

Für jede reguläre Grammatik G gibt es einen NFA M mit $\mathcal{L}(G) = \mathcal{L}(M)$.

Beweis.

Sei $G = (\Sigma, V, P, S)$ eine reguläre Grammatik.
Definiere NFA $M = (Z, \Sigma, \delta, S', E)$ mit

$$Z = V \cup \{X\}, \quad X \notin V$$

$$S' = \{S\}$$

$$E = \begin{cases} \{S, X\} & \text{falls } S \to \varepsilon \in P \\ \{X\} & \text{falls } S \to \varepsilon \notin P \end{cases}$$

$$B \in \delta(A, a) \text{ falls } A \to aB \in P$$

$$X \in \delta(A, a) \text{ falls } A \to a \in P$$

Reguläre Grammatiken nicht mächtiger als NFAs

Satz

Für jede reguläre Grammatik G gibt es einen NFA M mit $\mathcal{L}(G) = \mathcal{L}(M)$.

Beweis (Fortsetzung).

Dann gilt für $w = a_1 a_2 \dots a_n \in \Sigma^*$ mit $n \geq 1$:

$w \in \mathcal{L}(G)$

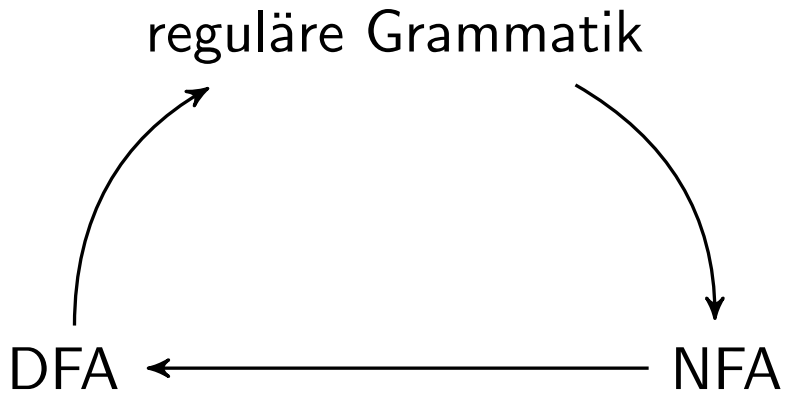
gdw. es eine Folge von Variablen $A_1, A_2, \dots, A_{n-1}$ gibt mit
$a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \dots a_{n-1} A_{n-1} \Rightarrow a_1 a_2 \dots a_n$.

gdw. es eine Folge von Variablen $A_1, A_2, \dots, A_{n-1}$ gibt mit
$A_1 \in \delta(S, a_1), A_2 \in \delta(A_1, a_2), \dots, X \in \delta(A_{n-1}, a_n)$.

gdw. $w \in \mathcal{L}(M)$.

Fall $w = \varepsilon$ gilt, da $S \in E$ gdw. $S \to \varepsilon \in P$. □

Endliche Automaten und reguläre Sprachen



Insgesamt folgt damit:

Folgerung

$\mathcal{L}$ regulär $\iff$ $\mathcal{L}$ wird von einem endlichen Automaten erkannt.

Fragen

Fragen?

Reguläre Ausdrücke

Formalismen für reguläre Sprachen

- DFAs, NFAs und reguläre Grammatiken können alle genau die regulären Sprachen beschreiben.

Formalismen für reguläre Sprachen

- DFAs, NFAs und reguläre Grammatiken können alle genau die regulären Sprachen beschreiben.
- Gibt es noch andere Konzepte mit der gleichen Mächtigkeit?

Formalismen für reguläre Sprachen

- DFAs, NFAs und reguläre Grammatiken können alle genau die regulären Sprachen beschreiben.
- Gibt es noch andere Konzepte mit der gleichen Mächtigkeit?
- Ja! $\rightsquigarrow$ reguläre Ausdrücke

Formalismen für reguläre Sprachen

- DFAs, NFAs und reguläre Grammatiken können alle genau die regulären Sprachen beschreiben.
- Gibt es noch andere Konzepte mit der gleichen Mächtigkeit?
- Ja! $\rightsquigarrow$ reguläre Ausdrücke

Formalismen für reguläre Sprachen

- DFAs, NFAs und reguläre Grammatiken können alle genau die regulären Sprachen beschreiben.
- Gibt es noch andere Konzepte mit der gleichen Mächtigkeit?
- Ja! $\rightsquigarrow$ reguläre Ausdrücke

Live-Demo

Reguläre Ausdrücke: Definition

Definition (regulärer Ausdruck)

Für ein Alphabet Σ sind reguläre Ausdrücke induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck
- ε ist ein regulärer Ausdruck
- Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- Für reguläre Ausdrücke α und β sind auch $(\alpha\beta)$, $(\alpha|\beta)$ und (α^*) reguläre Ausdrücke.

Reguläre Ausdrücke: Definition

Definition (regulärer Ausdruck)

Für ein Alphabet Σ sind reguläre Ausdrücke induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck
- ε ist ein regulärer Ausdruck
- Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- Für reguläre Ausdrücke α und β sind auch $(\alpha\beta)$, $(\alpha|\beta)$ und (α^*) reguläre Ausdrücke.

Klammerung per Konvention:

- Kleenesche Hülle α^* bindet stärker als Verkettung $\alpha\beta$
- Verkettung bindet stärker als Alternative $\alpha|\beta$
- z.B. $ab^*c|\varepsilon|abab^*$ entspricht $((((a(b^*))c)|\varepsilon)|(((ab)a)(b^*)))$

Reguläre Ausdrücke: Beispiele

Einige reguläre Ausdrücke für $\Sigma = \{0, 1\}$

- 0^*10^*
- $(0|1)^*1(0|1)^*$
- $((0|1)(0|1))^*$
- $01|10$
- $0(0|1)^*0|1(0|1)^*1|0|1$

Reguläre Ausdrücke: Sprache

Definition (Sprache eines regulären Ausdrucks)

Die Sprache $\mathcal{L}(\gamma)$ eines regulären Ausdrucks γ ist induktiv definiert wie folgt:

- Falls $\gamma = \emptyset$, ist $\mathcal{L}(\gamma) = \emptyset$.
- Falls $\gamma = \varepsilon$, ist $\mathcal{L}(\gamma) = \{\varepsilon\}$.
- Falls $\gamma = a$ mit $a \in \Sigma$, ist $\mathcal{L}(\gamma) = \{a\}$.
- Falls $\gamma = (\alpha\beta)$, wobei α und β reguläre Ausdrücke sind, ist $\mathcal{L}(\gamma) = \mathcal{L}(\alpha)\mathcal{L}(\beta)$.
- Falls $\gamma = (\alpha|\beta)$, wobei α und β reguläre Ausdrücke sind, ist $\mathcal{L}(\gamma) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$.
- Falls $\gamma = (\alpha^*)$ wobei α ein regulärer Ausdruck ist, so ist $\mathcal{L}(\gamma) = \mathcal{L}(\alpha)^*$.

Beispiele: Tafel

Endliche Sprachen durch reguläre Ausdrücke beschreibbar

Satz

Jede endliche Sprache ist durch einen regulären Ausdruck beschreibbar.

Beweis.

Für eine endliche Sprache $L = \{w_1, w_2, \ldots, w_n\}$ ist $\gamma = (\ldots((w_1|w_2)|w_3)\ldots w_n)$ offensichtlich ein regulärer Ausdruck mit $\mathcal{L}(\gamma) = L$. □

Quiz

Reguläre Ausdrücke nicht mächtiger als NFAs

Satz

Für jede durch einen regulären Ausdruck beschreibbare Sprache gibt es einen NFA, der sie akzeptiert.

Reguläre Ausdrücke nicht mächtiger als NFAs

Satz

Für jede durch einen regulären Ausdruck beschreibbare Sprache gibt es einen NFA, der sie akzeptiert.

Beweis.

Sei γ ein regulärer Ausdruck. Wir zeigen die Aussage per Induktion über die Struktur der regulären Ausdrücke.

Für $\gamma = \emptyset, \gamma = \varepsilon$ und $\gamma = a$ sind die NFAs, die $\mathcal{L}(G)$ akzeptieren, offensichtlich. ...

Reguläre Ausdrücke nicht mächtiger als NFAs

Satz

Für jede durch einen regulären Ausdruck beschreibbare Sprache gibt es einen NFA, der sie akzeptiert.

Beweis (Fortsetzung).

Hat γ die Form $\gamma = (\alpha\beta)$, seien M_α bzw. M_β NFAs, die (nach Induktionsvoraussetzung) $\mathcal{L}(\alpha)$ bzw. $\mathcal{L}(\beta)$ akzeptieren. Wir konstruieren NFA M für $\mathcal{L}(\gamma)$, indem wir M_α und M_β „in Reihe schalten“: Die Zustände von M sind die Zustände von M_α und M_β. Davon sind die Startzustände von M_α Startzustände. Falls $\varepsilon \in \mathcal{L}(\alpha)$, sind zudem die Startzustände von M_β Startzustände von M. Die Endzustände von M sind die Endzustände von M_β.
Alle Zustandsübergänge von M_α und M_β sind Übergänge von M. Zudem enthält M für jeden Übergang zu einem Endzustand in M_α genauso beschriftete Übergänge zu allen Startzuständen von M_β.

...

Reguläre Ausdrücke nicht mächtiger als NFAs

Satz

Für jede durch einen regulären Ausdruck beschreibbare Sprache gibt es einen NFA, der sie akzeptiert.

Beweis (Fortsetzung).

Hat γ die Form $\gamma = (\alpha|\beta)$, seien $M_\alpha = (Z_\alpha, \Sigma, \delta_\alpha, S_\alpha, E_\alpha)$ bzw. $M_\beta = (Z_\beta, \Sigma, \delta_\beta, S_\beta, E_\beta)$ NFAs, die $\mathcal{L}(\alpha)$ bzw. $\mathcal{L}(\beta)$ akzeptieren. Dann akzeptiert der „Vereinigungsautomat“

$$M = (Z_\alpha \cup Z_\beta, \Sigma, \delta_\alpha \cup \delta_\beta, S_\alpha \cup S_\beta, E_\alpha \cup E_\beta)$$

die Sprache $\mathcal{L}(\gamma)$. ...

Reguläre Ausdrücke nicht mächtiger als NFAs

Satz

Für jede durch einen regulären Ausdruck beschreibbare Sprache gibt es einen NFA, der sie akzeptiert.

Beweis (Fortsetzung).

Hat γ die Form $\gamma = (\alpha^*)$, sei $M_\alpha = (Z_\alpha, \Sigma, \delta_\alpha, S_\alpha, E_\alpha)$ ein NFAs, der $\mathcal{L}(\alpha)$ akzeptiert.
Falls $\varepsilon \notin \mathcal{L}(\alpha)$, füge zu M_α einen zusätzlichen Zustand hinzu, der Start- und Endzustand ist, aber nicht mit den anderen Zuständen verbunden. M_α erkennt nun $\mathcal{L}(\alpha) \cup \{\varepsilon\}$.
M entsteht aus M_α, indem jedem Zustand, der einen Übergang in einen Endzustand hat, ein Übergang mit dem gleichen Zeichen zu allen Startzuständen hinzugefügt wird.
Dann ist $\mathcal{L}(M) = \mathcal{L}(\gamma)$.

DFAs nicht mächtiger als reguläre Ausdrücke

Satz

Jede von einem DFA akzeptierte Sprache kann von einem regulären Ausdruck beschrieben werden.

DFAs nicht mächtiger als reguläre Ausdrücke

Satz

Jede von einem DFA akzeptierte Sprache kann von einem regulären Ausdruck beschrieben werden.

Beweis.

Sei $M = (\{z_1, \ldots, z_n\}, \Sigma, \delta, z_1, E)$ ein DFA.
Für $i, j \in \{1, \ldots, n\}$ und $k \in \{0, 1, \ldots, n\}$ definieren wir Sprachen $R^k_{i,j}$ und zeigen, dass sie durch reguläre Ausdrücke beschreibbar sind.

$$R^k_{i,j} = \{x \in \Sigma^* \mid \text{die Eingabe } x \text{ überführt den Automaten, gestartet in Zustand } z_i \text{ in den Zustand } z_j \text{ so dass keiner der „Zwischenzustände“ – ausser } z_i \text{ und } z_j \text{ – einen Index grösser } k \text{ hat}\}$$

...

DFAs nicht mächtiger als reguläre Ausdrücke

Satz

Jede von einem DFA akzeptierte Sprache kann von einem regulären Ausdruck beschrieben werden.

Beweis (Fortsetzung).

Für $k = 0$ und $i \neq j$ gilt $R^k_{i,j} = \{a \in \Sigma \mid \delta(z_i, a) = z_j\}$.
Für $k = 0$ und $i = j$ gilt $R^k_{i,j} = \{\varepsilon\} \cup \{a \in \Sigma \mid \delta(z_i, a) = z_j\}$.
In diesen Fällen ist $R^k_{i,j}$ endlich und kann daher durch einen regulären Ausdruck beschrieben werden. ...

DFAs nicht mächtiger als reguläre Ausdrücke

Satz

Jede von einem DFA akzeptierte Sprache kann von einem regulären Ausdruck beschrieben werden.

Beweis (Fortsetzung).

Beobachtung:

$$R_{i,j}^{k+1} = R_{i,j}^{k} \cup R_{i,k+1}^{k} (R_{k+1,k+1}^{k})^* R_{k+1,j}^{k}$$

Zustand z_{k+1} wird entweder nicht benötigt, um von z_i zu z_j zu gelangen, oder er wird ein oder mehrfach (in Schleifen) durchlaufen. ...

DFAs nicht mächtiger als reguläre Ausdrücke

Satz

Jede von einem DFA akzeptierte Sprache kann von einem regulären Ausdruck beschrieben werden.

Beweis (Fortsetzung).

Falls $\alpha_{i,j}^k$ regulärer Ausdruck für $R_{i,j}^k$ ist, lässt sich obige Formel beschreiben mit $\alpha_{i,j}^{k+1} = (\alpha_{i,j}^k | \alpha_{i,k+1}^k (\alpha_{k+1,k+1}^k)^* \alpha_{k+1,j}^k)$.

Die von M akzeptierte Sprache lässt sich mittels der Sprachen $R_{i,j}^k$ beschreiben als $\mathcal{L}(M) = \bigcup_{z_i \in E} R_{1,i}^n$.

Sind $i_1, i_2, \ldots, i_m$ die Indizes der Endzustände, beschreibt daher der reguläre Ausdruck $(\alpha_{1,i_1}^n | \alpha_{1,i_2}^n | i \ldots | \alpha_{1,i_m}^n)$ die Sprache $\mathcal{L}(M)$. □

Reguläre Sprachen

Satz (Kleene)

Die Menge der durch reguläre Ausdrücke beschreibbaren Sprachen ist genau die Menge der regulären Sprachen.

Dies folgt direkt aus den vorherigen zwei Sätzen.

Zusammenfassung

Zusammenfassung

- Wir kennen nun vier Formalismen, die alle genau die regulären Sprachen beschreiben.
- DFAs sind Automaten, bei denen jeder Zustandsübergang genau festgelegt ist.
- NFAs erkennen ein Wort, wenn es mindestens eine akzeptierende Zustandsfolge gibt.
- Reguläre Ausdrücke beschreiben Zusammensetzung der Wörter in der Sprache.