Determinisierung von endlichen Automaten

Prof. Dr. David Sabel

LFE Theoretische Informatik

Letzte Änderung der Folien: 10. Mai 2022

Wiederholung: NFA

Definition

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

- Z ist eine endliche Menge von Zuständen,
- Σ ist das (endliche) Eingabealphabet mit $(Z \cap \Sigma) = \emptyset$,
- $\delta : Z \times \Sigma \to \mathcal{P}(Z)$ ist die Zustandsüberführungsfunktion,
- $S \subseteq Z$ ist die Menge der Startzustände und
- $E \subseteq Z$ ist die Menge der Endzustände.

Wiederholung: Akzeptanz beim NFA

„Ein Wort w wird vom NFA akzeptiert, wenn es einen Pfad von einem Startzustand zum Endzustand entlang w **gibt**“

Definition (Akzeptierte Sprache eines NFA)

Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA.
Wir definieren $\widehat{\delta} : (\mathcal{P}(Z) \times \Sigma^*) \to \mathcal{P}(Z)$ induktiv durch:

$$\begin{aligned} \widehat{\delta}(X, \varepsilon) &:= X \text{ für alle } X \subseteq Z \\ \widehat{\delta}(X, aw) &:= \bigcup_{z \in X} \widehat{\delta}(\delta(z, a), w) \text{ für alle } X \subseteq Z \end{aligned}$$

Die von M akzeptierte Sprache ist

$$L(M) = \{w \in \Sigma^* \mid \widehat{\delta}(S, w) \cap E \neq \emptyset\}$$

Jede reguläre Sprache wird durch einen NFA erkannt

Theorem 4.4.1

Für jede reguläre Sprache L gibt es einen NFA M mit $L(M) = L$.

Beweis: Zeige für jede reguläre Grammatik G lässt sich NFA M konstruieren mit $L(G) = L(M)$:

- Sei $G = (V, \Sigma, P, S)$ eine reguläre Grammatik mit $L(G) = L$.

Jede reguläre Sprache wird durch einen NFA erkannt

Theorem 4.4.1

Für jede reguläre Sprache L gibt es einen NFA M mit $L(M) = L$.

Beweis: Zeige für jede reguläre Grammatik G lässt sich NFA M konstruieren mit $L(G) = L(M)$:

- Sei $G = (V, \Sigma, P, S)$ eine reguläre Grammatik mit $L(G) = L$.
- Sei $M = (Z, \Sigma, \delta, S', E)$ ein NFA mit
  - $Z = V \cup \{z_E\}$ (z_E neu), $S' = \{S\}$ und $E = \begin{cases} \{z_E, S\}, & \text{falls } S \to \varepsilon \in P \\ \{z_E\}, & \text{sonst} \end{cases}$
  - $\delta(A, a) := \{B \mid A \to aB \in P\} \cup \{z_E \mid \text{falls } A \to a \in P\}$ und $\delta(z_E, a) := \emptyset$ für alle $a \in \Sigma$.

Jede reguläre Sprache wird durch einen NFA erkannt

Theorem 4.4.1

Für jede reguläre Sprache L gibt es einen NFA M mit $L(M) = L$.

Beweis: Zeige für jede reguläre Grammatik G lässt sich NFA M konstruieren mit $L(G) = L(M)$:

- Sei $G = (V, \Sigma, P, S)$ eine reguläre Grammatik mit $L(G) = L$.
- Sei $M = (Z, \Sigma, \delta, S', E)$ ein NFA mit
  - $Z = V \cup \{z_E\}$ (z_E neu), $S' = \{S\}$ und $E = \begin{cases} \{z_E, S\}, & \text{falls } S \to \varepsilon \in P \\ \{z_E\}, & \text{sonst} \end{cases}$
  - $\delta(A, a) := \{B \mid A \to aB \in P\} \cup \{z_E \mid \text{falls } A \to a \in P\}$ und $\delta(z_E, a) := \emptyset$ für alle $a \in \Sigma$.
- Offensichtlich gilt: $\varepsilon \in L(M) \iff \varepsilon \in L(G)$.

Jede reguläre Sprache wird durch einen NFA erkannt

Theorem 4.4.1

Für jede reguläre Sprache L gibt es einen NFA M mit $L(M) = L$.

Beweis: Zeige für jede reguläre Grammatik G lässt sich NFA M konstruieren mit $L(G) = L(M)$:

- Sei $G = (V, \Sigma, P, S)$ eine reguläre Grammatik mit $L(G) = L$.
- Sei $M = (Z, \Sigma, \delta, S', E)$ ein NFA mit
  - $Z = V \cup \{z_E\}$ (z_E neu), $S' = \{S\}$ und $E = \begin{cases} \{z_E, S\}, & \text{falls } S \to \varepsilon \in P \\ \{z_E\}, & \text{sonst} \end{cases}$
  - $\delta(A, a) := \{B \mid A \to aB \in P\} \cup \{z_E \mid \text{falls } A \to a \in P\}$ und $\delta(z_E, a) := \emptyset$ für alle $a \in \Sigma$.
- Offensichtlich gilt: $\varepsilon \in L(M) \iff \varepsilon \in L(G)$. Für $w = a_1 \cdots a_n$ gilt:

  $w \in L(G)$ g.d.w. $S \Rightarrow_G a_1 A_1 \Rightarrow_G \ldots \Rightarrow_G a_1 \cdots a_{n-1} A_{n-1} \Rightarrow_G a_1 \cdots a_n$

  g.d.w. Es gibt Zustände $A_1, \ldots, A_{n-1}$ mit $A_1 \in \delta(S, a_1)$, $A_{i+1} \in \delta(A_i, a_{i+1})$ für $1 \leq i \leq n-2$ und $z_E \in \delta(A_{n-1}, a_n)$

  g.d.w. $w \in L(M)$

Beispiel: Konstruktion NFA aus Typ 3-Grammatik

Betrachte die reguläre Grammatik $G = (V, \Sigma, P, A)$
mit $V = \{A, B, C, D\}$, $\Sigma = \{a, b, c\}$ und

$$P = \{ \begin{array}{l} A \to \varepsilon \mid aB \mid bB \mid cB \mid aC, \\ B \to aB \mid bB \mid cB \mid aC, \\ C \to aD \mid bD \mid cD, \\ D \to a \mid b \mid c \} \end{array} .$$

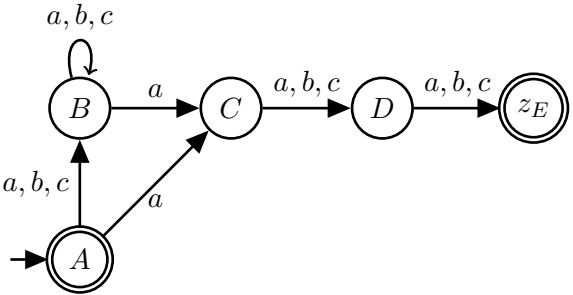
Konstruktion des dazu passenden NFA: $M = (Z, \Sigma, \delta, S, E)$ mit

- $Z = V \cup \{z_E\} = \{A, B, C, D, z_E\}$,
- $E = \{A, z_E\}$, $S = \{A\}$ und

$\delta(A,a)=\{B,C\}$	$\delta(B,a)=\{B,C\}$	$\delta(C,a)=\{D\}$	$\delta(D,a)=\{z_E\}$	$\delta(z_E,a)=\emptyset$
$\delta(A,b)=\{B\}$	$\delta(B,b)=\{B\}$	$\delta(C,b)=\{D\}$	$\delta(D,b)=\{z_E\}$	$\delta(z_E,b)=\emptyset$
$\delta(A,c)=\{B\}$	$\delta(B,c)=\{B\}$	$\delta(C,c)=\{D\}$	$\delta(D,c)=\{z_E\}$	$\delta(z_E,c)=\emptyset$

Beispiel: Konstruktion NFA aus Typ 3-Grammatik (2)

Der Zustandsgraph zu M ist



NFAs in DFAs transformieren

Theorem 4.5.1 (Rabin & Scott 1959)

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

NFAs in DFAs transformieren

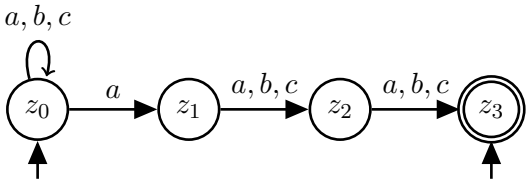
Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*

Z.B. babbabb



NFAs in DFAs transformieren

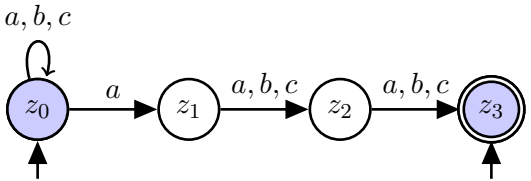
Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*

Z.B. babbabb



NFAs in DFAs transformieren

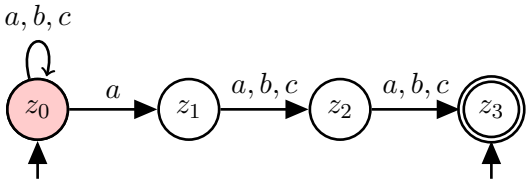
Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*

Z.B. babbabb



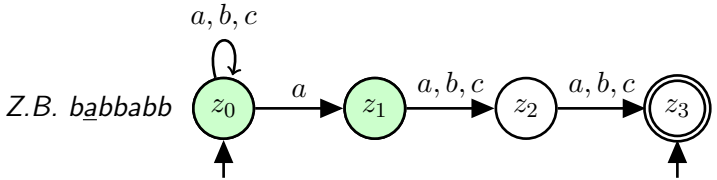
NFAs in DFAs transformieren

Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*



NFAs in DFAs transformieren

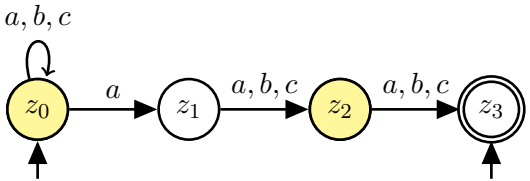
Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*

Z.B. ba<u>b</u>babb



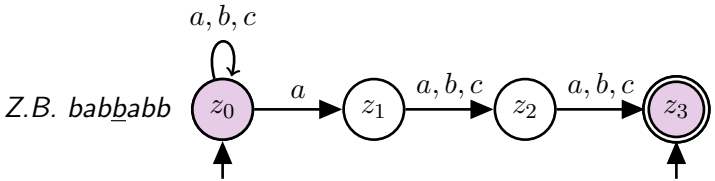
NFAs in DFAs transformieren

Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*



NFAs in DFAs transformieren

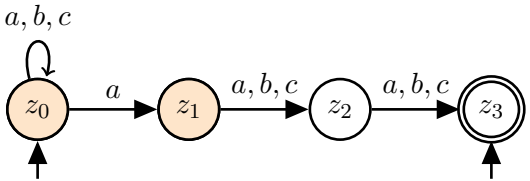
Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*

Z.B. babbabb



NFAs in DFAs transformieren

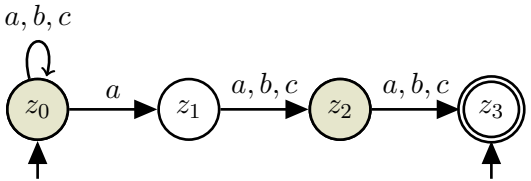
Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*

Z.B. babbabb



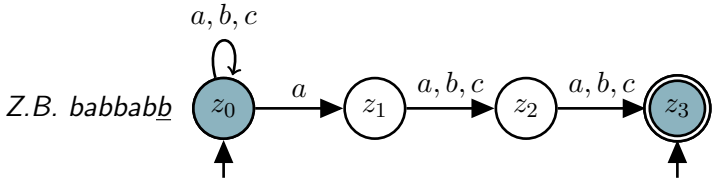
NFAs in DFAs transformieren

Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*



NFAs in DFAs transformieren

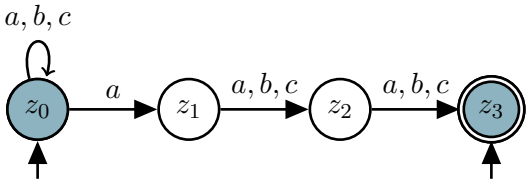
Theorem 4.5.1 (Rabin & Scott 1959)

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

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich *„der DFA* **alle** *Zustände merkt, in denen der NFA sein könnte“*

Z.B. babbabb



- *Konstruktion: Jede Teilmenge von Zuständen des NFA wird zu einem Zustand des DFA (daher: Potenzmengenkonstruktion)*

Potenzmengenkonstruktion

Für NFA $M = (Z, \Sigma, \delta, S, E)$ konstruieren wir den DFA $M' = (Z', \Sigma, \delta', S', E')$ mit

- $Z' = \mathcal{P}(Z)$
  „Zustandsmenge ist Potenzmenge von Z“
- $S' = S$
  „Startzustand ist Menge S aller Startzustände von M“
- $E' = \{X \in Z' \mid (E \cap X) \neq \emptyset\}$
  „Jede Menge, die mind. einen Endzustand von E enthält, ist Endzustand in M'“
- $\delta'(X, a) = \bigcup_{z \in X} \delta(z, a) = \widehat{\delta}(X, a)$
  „$\delta'(X, a)$ berechnet alle von einem Zustand in X aus über a erreichbaren Zustände.“

Korrektheit der Potenzmengenkonstruktion

Wir beweisen, dass $L(M) = L(M')$ gilt, indem wir zeigen:

$$w \in L(M) \text{ g.d.w. } w \in L(M')$$

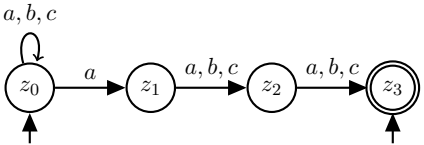
- Fall $w = \varepsilon$:

$$\varepsilon \in L(M) \text{ g.d.w. } S \cap E \neq \emptyset \text{ g.d.w } S \in E' \text{ g.d.w. } \varepsilon \in L(M')$$

- Fall $w = a_1 \cdots a_n \in \Sigma^*$:

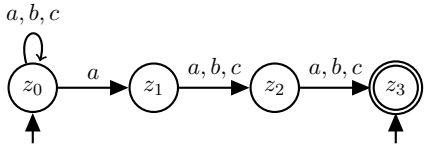
$$\begin{array}{ll} & w \in L(M) \\ \text{g.d.w.} & \widehat{\delta}(S, w) \cap E \neq \emptyset \\ \text{g.d.w.} & \text{Es gibt Teilmengen } Z_1, \ldots, Z_n \text{ von } Z \text{ mit} \\ & \delta(S, a_1) = Z_1,\ \delta(Z_i, a_{i+1}) = Z_{i+1} \text{ für } i = 1, \ldots, n-1 \\ & \text{und } Z_n \cap E \neq \emptyset \\ \text{g.d.w.} & \widehat{\delta'}(S', w) \in E' \\ \text{g.d.w} & w \in L(M') \end{array}$$

Beispiel: Potenzmengenkonstruktion

Für NFA 

wird der DFA konstruiert:

Beispiel: Potenzmengenkonstruktion

Für NFA 

wird der DFA konstruiert:

$M' = (\mathcal{P}(\{z_0, z_1, z_2, z_3\}), \{a, b, c\}, \delta', S', E')$ mit $S' = \{z_0, z_3\}$

$E' = \{\{z_3\}, \{z_0, z_3\}, \{z_1, z_3\}, \{z_2, z_3\}, \{z_0, z_1, z_3\}, \{z_0, z_2, z_3\}, \{z_1, z_2, z_3\}, \{z_0, z_1, z_2, z_3\}\}$

$\delta'(\emptyset, d) = \emptyset$ für $d \in \{a, b, c\}$

$\delta'(\{z_0\}, a) = \{z_0, z_1\}$

$\delta'(\{z_0\}, d) = \{z_0\}$ für $d \in \{b, c\}$

$\delta'(\{z_1\}, d) = \{z_2\}$ für $d \in \{a, b, c\}$

$\delta'(\{z_2\}, d) = \{z_3\}$ für $d \in \{a, b, c\}$

$\delta'(\{z_3\}, d) = \emptyset$ für $d \in \{a, b, c\}$

$\delta'(\{z_0, z_1\}, a) = \{z_0, z_1, z_2\}$

$\delta'(\{z_0, z_1\}, d) = \{z_0, z_2\}$ für $d \in \{b, c\}$

$\delta'(\{z_0, z_2\}, a) = \{z_0, z_1, z_3\}$

$\delta'(\{z_0, z_2\}, d) = \{z_0, z_3\}$ für $d \in \{b, c\}$

$\delta'(\{z_0, z_3\}, a) = \{z_0, z_1\}$

$\delta'(\{z_0, z_3\}, d) = \{z_0\}$ für $d \in \{b, c\}$

$\delta'(\{z_1, z_2\}, d) = \{z_2, z_3\}$ für $d \in \{a, b, c\}$

$\delta'(\{z_1, z_3\}, d) = \{z_2\}$ für $d \in \{a, b, c\}$

$\delta'(\{z_2, z_3\}, d) = \{z_3\}$ für $d \in \{a, b, c\}$

$\delta'(\{z_0, z_1, z_2\}, a) = \{z_0, z_1, z_2, z_3\}$

$\delta'(\{z_0, z_1, z_2\}, d) = \{z_0, z_2, z_3\}$ für $d \in \{b, c\}$

$\delta'(\{z_0, z_1, z_3\}, a) = \{z_0, z_1, z_2\}$

$\delta'(\{z_0, z_1, z_3\}, d) = \{z_0, z_2\}$ für $d \in \{b, c\}$

$\delta'(\{z_0, z_2, z_3\}, a) = \{z_0, z_1, z_3\}$

$\delta'(\{z_0, z_2, z_3\}, d) = \{z_0, z_3\}$ für $d \in \{b, c\}$

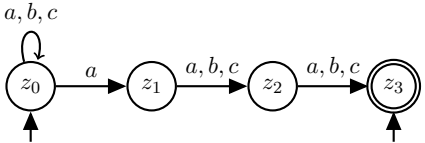
$\delta'(\{z_1, z_2, z_3\}, d) = \{z_2, z_3\}$ für $d \in \{a, b, c\}$

$\delta'(\{z_0, z_1, z_2, z_3\}, a) = \{z_0, z_1, z_2, z_3\}$

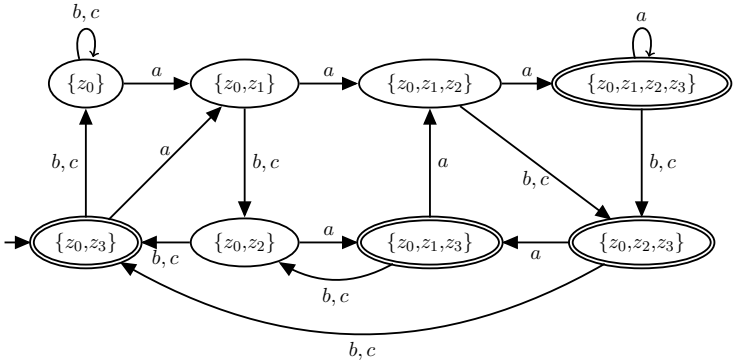
$\delta'(\{z_0, z_1, z_2, z_3\}, b) = \{z_0, z_2, z_3\}$

$\delta'(\{z_0, z_1, z_2, z_3\}, c) = \{z_0, z_2, z_3\}$

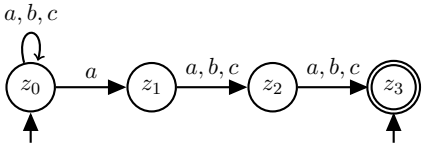
Beispiel: Potenzmengenkonstruktion (2)



DFA als Zustandsgraph (nur erreichbare Zustände):

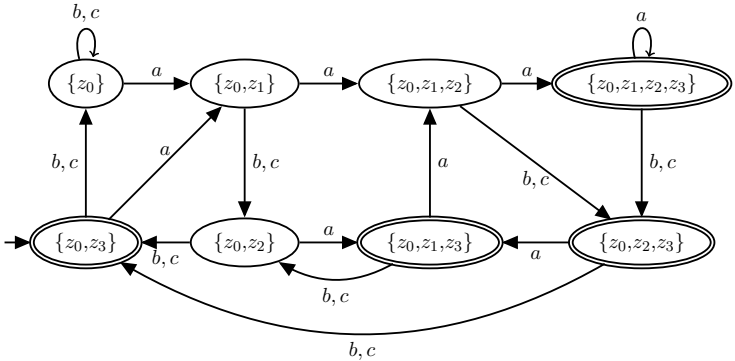


Beispiel: Potenzmengenkonstruktion (2)

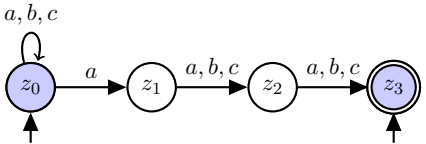


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

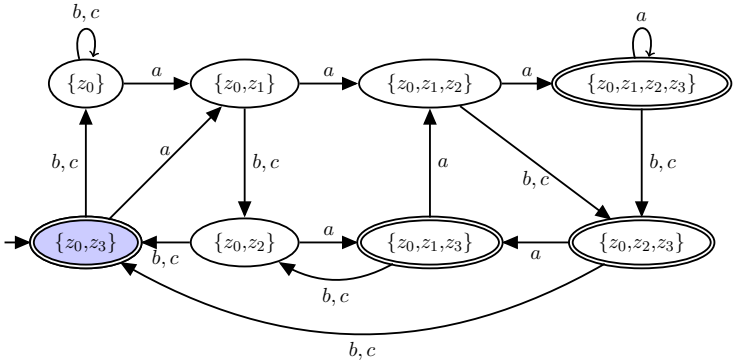


Beispiel: Potenzmengenkonstruktion (2)

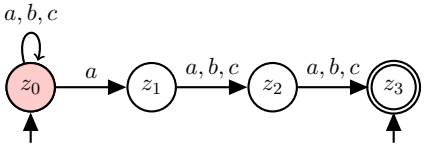


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

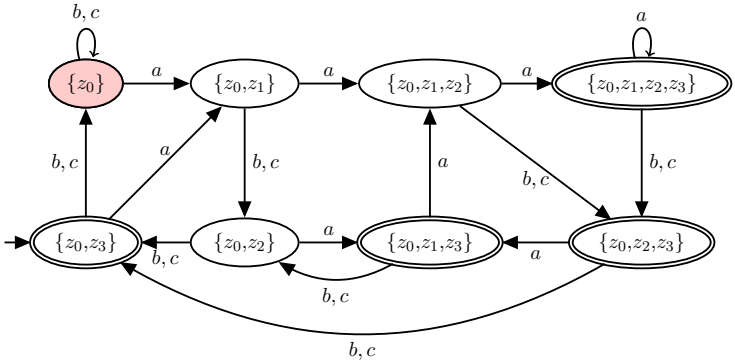


Beispiel: Potenzmengenkonstruktion (2)

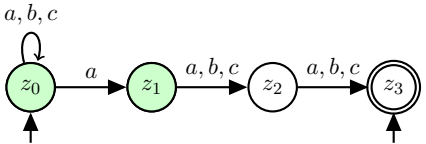


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

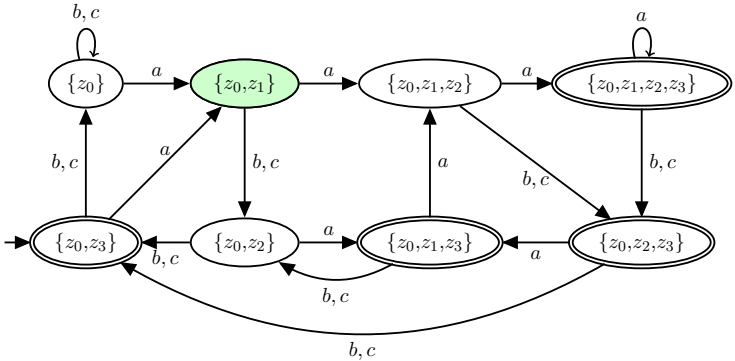


Beispiel: Potenzmengenkonstruktion (2)

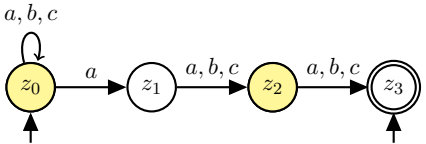


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

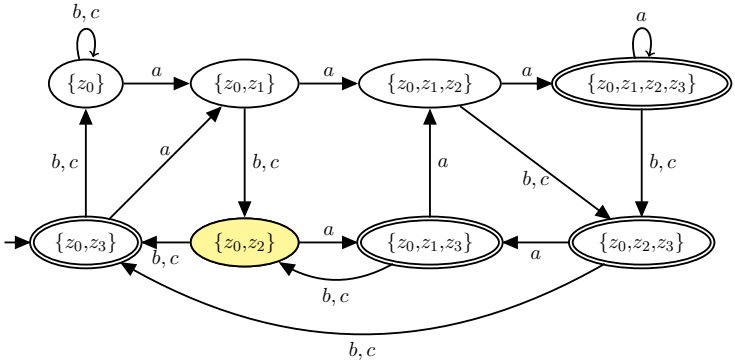


Beispiel: Potenzmengenkonstruktion (2)

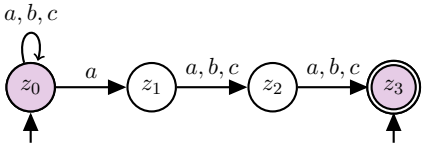


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

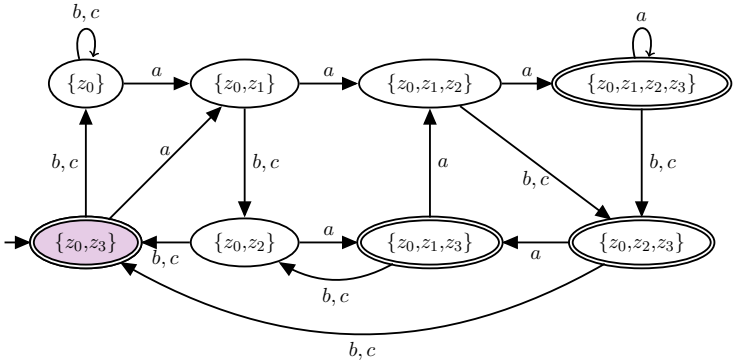


Beispiel: Potenzmengenkonstruktion (2)

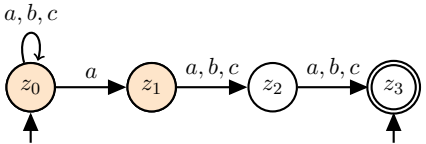


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

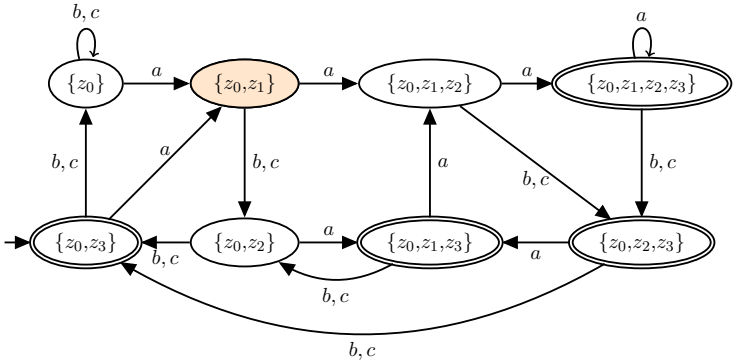


Beispiel: Potenzmengenkonstruktion (2)

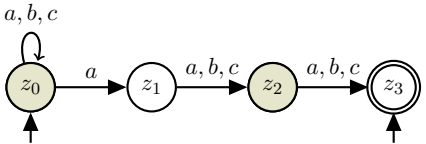


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

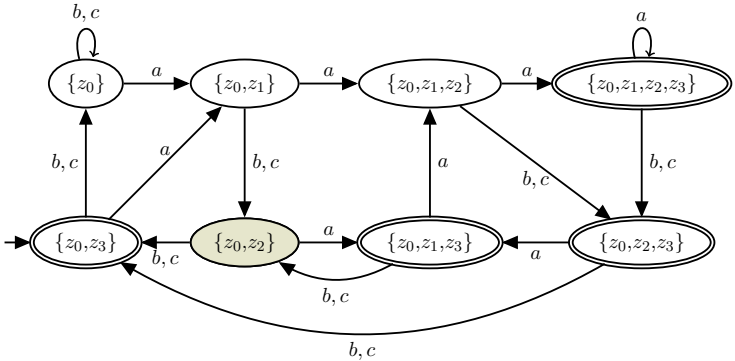


Beispiel: Potenzmengenkonstruktion (2)

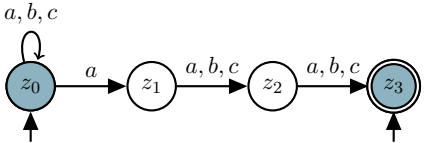


Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):

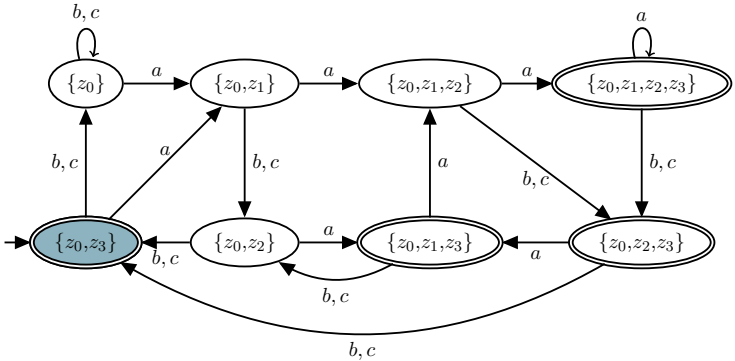


Beispiel: Potenzmengenkonstruktion (2)



Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):



DFAs & NFAs sind Formalismen für Typ 3-Sprachen

Theorem 4.5.4

DFAs und NFAs erkennen genau die regulären Sprachen.

Das folgt aus:

- Theorem 4.2.1: Sei M ein DFA. Dann ist $L(M)$ regulär.
- Theorem 4.4.1: Für jede reguläre Sprache L gibt es einen NFA M mit $L(M) = L$.
- Theorem 4.5.1: Jede von einem NFA akzeptierte Sprache ist auch durch einen DFA akzeptierbar.
- Jeder DFA kann leicht auch als NFA interpretiert werden

Größe des DFAs vs NFAs

- Sei M ein NFA mit n Zuständen.
- Der durch die Potenzmengenkonstruktion erstellte DFA hat 2^n Zustände!
- D.h. der Platz explodiert uns!
- Frage: Geht es besser (unsere Kodierung ist zu einfach) oder nicht?
- Das folgende Lemma zeigt, dass es nicht wirklich besser geht

Lemma

Sei $L_n = \{uav \mid u \in \{a,b\}^*, v \in \{a,b\}^{n-1}\}$ für $n \in \mathbb{N}_{>0}$.
(Sprache aller Wörter aus $\{a,b\}^*$, die an n-letzter Stelle ein a haben).

- Es gibt NFA M_n mit $L(M_n) = L_n$ und M_n hat $n+1$ Zustände.
- Jeder DFA M'_n mit $L(M'_n) = L_n$, hat mindestens 2^n Zustände.

Beweis: Nächste Vorlesung (nur FSK)