Hochschule RheinMain

Automatentheorie und Formale Sprachen

für die Studiengänge
- Angewandte Informatik
- Informatik - Technische Systeme

03 Reguläre Sprachen: Endliche Automaten

Prof. Dr. David Sabel
Sommersemester 2025

Stand der Folien: 7. Juli 2025

Motivation

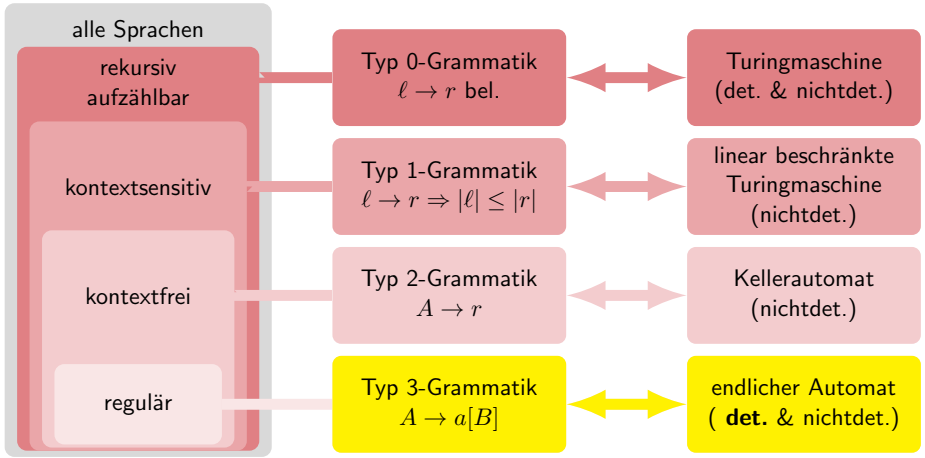
Hochschule RheinMain

- Formalismen, um Sprachen zu repräsentieren:
  - Grammatiken: Sie erzeugen Wörter einer formale Sprache.
  - Maschinenmodelle: Sie erkennen Wörter einer formalen Sprache.
- Welcher Formalismus „besser“ ist, hängt oft von der konkreten Fragestellung ab.

Wichtige Fragestellung: Welche Maschine akzeptiert welche Sprachklasse?

Grammatiken & Maschinen für die Chomsky-Hierarchie

Hochschule RheinMain



Reguläre Sprachen

Hochschule RheinMain

Wiederholung:

- Eine Sprache heißt **regulär** (bzw. vom Typ 3), wenn sie von einer Typ 3-Grammatik erzeugt wird.
- Eine Grammatik $G = (V, \Sigma, P, S)$ ist vom Typ 3 (bzw. regulär), wenn alle Produktionen von der Form

$$A \to a \text{ oder } A \to aB$$

mit $A, B \in V$ und $a \in \Sigma$ sind.

DETERMINISTISCHE ENDLICHE AUTOMATEN

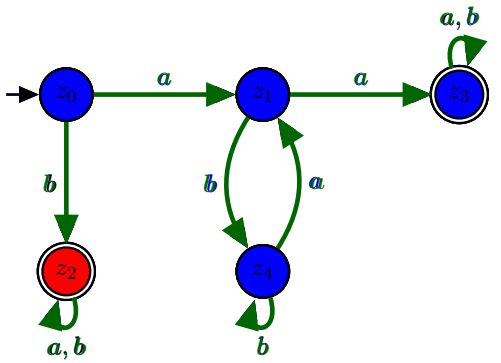
Hochschule RheinMain

Deterministische endliche Automaten

Hochschule RheinMain

Ein DFA besteht aus:

- Zuständen
- Startzustand (genau einer)
- Endzuständen (mehrere erlaubt)
- Übergängen
  von **jedem** Zustand und für **jedes** Zeichen aus dem Alphabet (hier $\{a, b\}$)

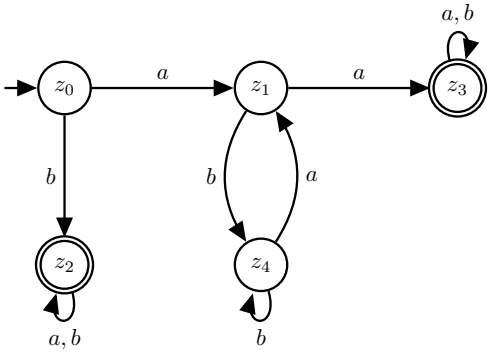


Lauf des Automaten auf Eingabewort w

- starte im Startzustand
- arbeite w zeichenweise ab:
  Zustand wechseln mit Übergängen
- wenn danach in Endzustand,
  dann akzeptieren,
  sonst verwerfen

Quiz 1

Hochschule RheinMain

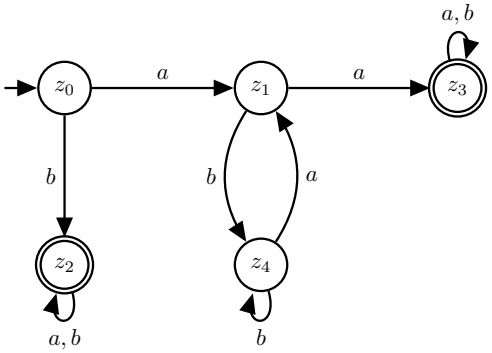


Wird bbb vom Automat akzeptiert?

arsnova.hs-rm.de
6750 1376

Quiz 2

Hochschule RheinMain

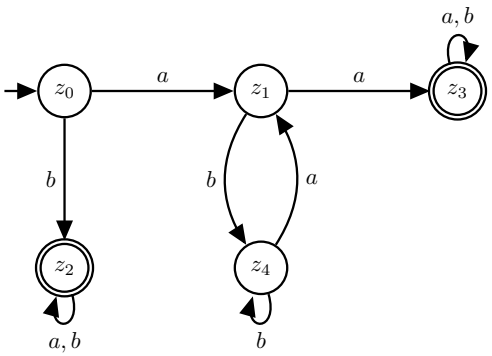


Wird abb vom Automat akzeptiert?

arsnova.hs-rm.de
6750 1376

Quiz 3

Hochschule RheinMain



Welche Wörter akzeptiert der DFA?

arsnova.hs-rm.de
6750 1376

DFA: Definition

Hochschule RheinMain

Definition (Deterministischer Endlicher Automat, DFA)

Ein deterministischer endlicher Automat (deterministic finite automaton, DFA) ist ein 5-Tupel

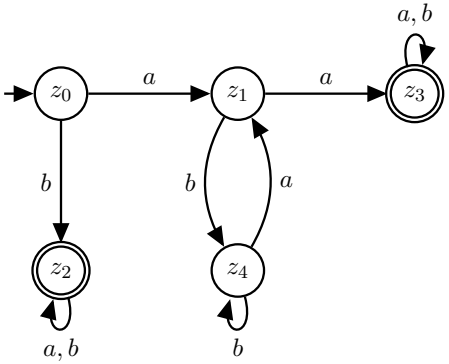
$$M = (Z, \Sigma, \delta, z_0, E)$$

wobei

- Z ist eine endliche Menge von Zuständen,
- Σ ist das (endliche) Eingabealphabet mit $(Z \cap \Sigma) = \emptyset$,
- $\delta : Z \times \Sigma \to Z$ ist die Zustandsüberführungsfunktion (oder Überführungsfunktion),
- $z_0 \in Z$ ist der Startzustand und
- $E \subseteq Z$ ist die Menge der Endzustände (oder auch akzeptierende Zustände).

Beispiel

Hochschule RheinMain



$$M = (Z, \Sigma, \delta, z_0, E)$$
$$= (\{z_0, z_1, z_2, z_3, z_4\}, \{a, b\}, \delta, z_0, \{z_2, z_3\})$$

mit

$$\begin{array}{ll} \delta(z_0, a) = z_1 & \delta(z_0, b) = z_2 \\ \delta(z_1, a) = z_3 & \delta(z_1, b) = z_4 \\ \delta(z_2, a) = z_2 & \delta(z_2, b) = z_2 \\ \delta(z_3, a) = z_3 & \delta(z_3, b) = z_3 \\ \delta(z_4, a) = z_1 & \delta(z_4, b) = z_4 \end{array}$$

Lauf

Hochschule RheinMain

Definition

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA und $w \in \Sigma^*$ mit $|w| = n$.

Die Folge von Zuständen $q_0, \ldots, q_n$ mit $q_0 = z_0$ und $q_i = \delta(q_{i-1}, w[i])$

bezeichnet man als **Lauf** von M für Wort w.

Für einen solchen Lauf schreiben wir auch:

$$q_0 \xrightarrow{w[1]} q_1 \xrightarrow{w[2]} \cdots \xrightarrow{w[n-1]} q_{n-1} \xrightarrow{w[n]} q_n$$

Beispiele für Läufe

Hochschule RheinMain

$M = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_0\})$ mit

δ	a	b
z_0	z_1	z_0
z_1	z_2	z_1
z_2	z_0	z_2

Lauf für Wort $abbaaabba$:

$$z_0 \xrightarrow{a} z_1 \xrightarrow{b} z_1 \xrightarrow{b} z_1 \xrightarrow{a} z_2 \xrightarrow{a} z_0 \xrightarrow{a} z_1 \xrightarrow{b} z_1 \xrightarrow{b} z_1 \xrightarrow{a} z_2$$

Lauf für Wort $bbababa$:

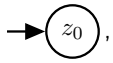
$$z_0 \xrightarrow{b} z_0 \xrightarrow{b} z_0 \xrightarrow{a} z_1 \xrightarrow{b} z_1 \xrightarrow{a} z_2 \xrightarrow{b} z_2 \xrightarrow{a} z_0$$

Zustandsgraph eines DFA

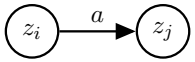
Hochschule RheinMain

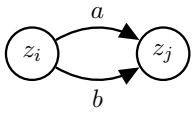
Für DFA $M = (Z, \Sigma, \delta, z_0, E)$

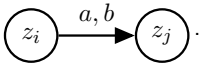
- Für $z \in Z$ gibt es Knoten 

- Startzustand $z_0 \in Z$: eingehender Pfeil 

,
- Endzustände $z \in E$: doppelte Kreise 

,
- Übergänge $\delta(z_i, a) = z_j$ als Kante 

und statt 

zeichnen wir 

.

Akzeptierte Sprache eines DFA

Hochschule RheinMain

Definition (Akzeptierte Sprache eines DFA)

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA.
Wir definieren die Funktion $\widehat{\delta} : Z \times \Sigma^* \to Z$ durch

$$\widehat{\delta}(z, \varepsilon) := z \quad \text{und} \quad \widehat{\delta}(z, aw) := \widehat{\delta}(\delta(z, a), w)$$

Die von M **akzeptierte Sprache** ist

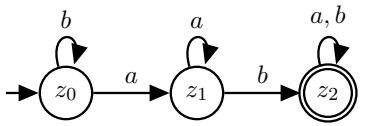
$$L(M) := \{w \in \Sigma^* \mid \widehat{\delta}(z_0, w) \in E\}.$$

$\widehat{\delta}$ wendet δ solange an, bis das Eingabewort abgearbeitet ist

Beispiel

Hochschule RheinMain

DFA M

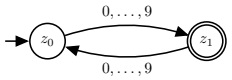


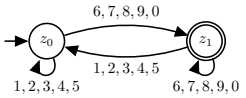
$$L(M) = \{uabv \mid uv \in \{a, b\}^*\}$$

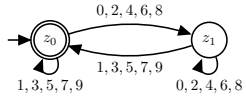
Quiz 4

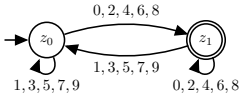
Hochschule RheinMain

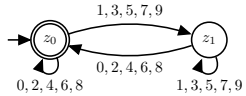
Welcher der folgenden DFAs akzeptiert genau die Wörter, die gerade Zahlen sind (führende 0en erlaubt)?

A)


B)


C)


D)


E)


arsnova.hs-rm.de
6750 1376

DFA konstruieren

Hochschule RheinMain

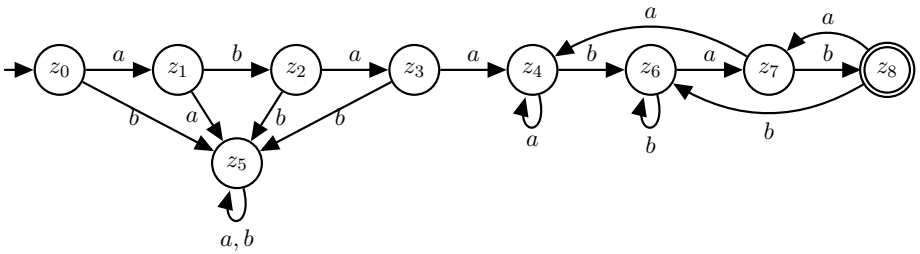
Für gegebene Sprache L einen DFA konstruieren:

- Falls möglich: Wörter zerlegen und Teilautomaten konstruieren und zusammensetzen
- Je nach Sprache:
  - Pfade konstruieren für akzeptierte Wörter (z.B. L enthält alle Wörter, die mit a beginnen) oder
  - Pfade konstruieren für Wörter, die verworfen werden (z.B. L enthält nur Wörter, die nicht mit a beginnen)
- Zustände verwenden, um sich Informationen zu merken
  Achtung: Endlich viele Zustände = Endliche Information
- Vervollständigen aller anderen Pfade
  eventuell Müllzustand verwenden

Beispiel: DFA konstruieren

Hochschule RheinMain

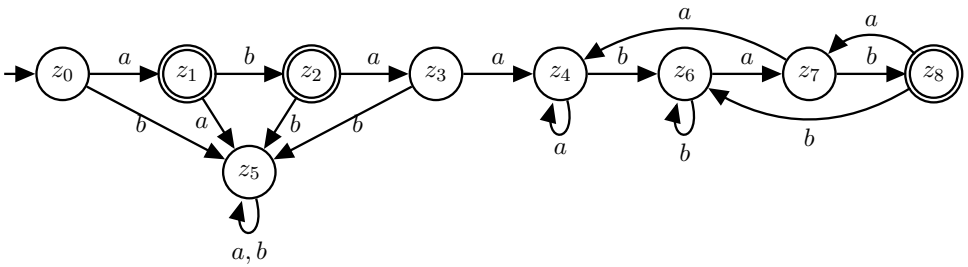
Konstruiere DFA über $\Sigma = \{a, b\}$, der alle Wörter akzeptiert, die mit $abaa$ beginnen und mit bab enden:



Beispiel: DFA konstruieren (2)

Hochschule RheinMain

Zusätzlich die Wörter a und ab akzeptieren



Zusammenfassung: DFAs

Hochschule RheinMain

Deterministische endliche Automaten

- haben endlich viele Zustände
- starten im Startzustand
- lesen zeichenweise ein Eingabewort
- wechseln dabei den Zustand (eindeutig)

Nach Lesen der Eingabe: akzeptieren oder verwerfen

- Akzeptieren = in einem Endzustand
- Verwerfen = in keinem Endzustand

Akzeptierte Sprache = alle Wörter, für die der Automat akzeptiert

Hochschule RheinMain

MINIMIERUNG VON DFAS

- Äquivalenzklassenautomat
- Sprachäquivalenz und Minimalität
- Zustandsminimierung von DFAs
- Korrektheit und Laufzeit

Minimale DFAs

Hochschule RheinMain

- Zur kompakten Darstellung und Implementierung ist man interessiert an DFAs mit möglichst wenigen Zuständen.
- Ein DFA M ist minimal, wenn jeder DFA M' mit $L(M) = L(M')$ nicht weniger Zustände als M hat.
- Jetzt: Verfahren, um aus einem gegebenen DFA einen minimalen DFA zu erzeugen.

Äquivalenzklassenautomat

Hochschule RheinMain

Definition (Äquivalenzklassenautomat)

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA.
Wir nennen zwei Zustände $z, z' \in Z$ äquivalent und schreiben $z \equiv z'$ falls gilt:

$$\text{für alle } w \in \Sigma^* : \widehat{\delta}(z, w) \in E \iff \widehat{\delta}(z', w) \in E.$$

„Akzeptanz von z und z' aus ist gleich für alle Wörter“

Der Äquivalenzklassenautomat zu M ist der DFA $M' = (Z', \Sigma, \delta', z_0', E')$ mit

$$Z' = \{[z]_\equiv \mid z \in Z\},\ z_0' = [z_0]_\equiv,\ E' = \{[z]_\equiv \mid z \in E\} \text{ und } \delta'([z]_\equiv, a) = [\delta(z, a)]_\equiv.$$

„ersetze Zustände durch Äquivalenzklassen“

Äquivalenzklassenautomat: Korrektheit und Minimalität

Hochschule RheinMain

Satz

Für DFA $M = (Z, \Sigma, \delta, z_0, E)$ und $M' = (Z', \Sigma, \delta', z_0', E')$ der Äquivalenzklassenautomat zu M gilt:

1. $L(M) = L(M')$
2. Falls alle Zustände in Z vom Startzustand z_0 erreichbar sind, dann ist M' minimal.

Beweis (Teil 1): Sei $w \in \Sigma^*$. Dann gilt:

- M durchläuft die Zustandsfolge $q_0, \ldots, q_{|w|}$ entlang w und akzeptiert w genau dann, wenn $q_{|w|} \in E$ gilt.
- M' durchläuft die Zustandsfolge $[q_0]_\equiv, \ldots, [q_{|w|}]_\equiv$ und akzeptiert w genau dann, wenn $[q_{|w|}]_\equiv \in E'$ gilt.

Da per Definition $[q_{|w|}]_\equiv \in E'$ gdw. $q_{|w|} \in E$, folgt M und M' akzeptieren dieselben Wörter.

Äquivalenzklassenautomat: Minimalität

Hochschule RheinMain

Beweis (Teil 2):

Für jedes Paar von Zuständen $[z_i]_\equiv \neq [z_j]_\equiv$ in Z' gibt es Wörter u, v, w mit

- $\widehat{\delta}(z_0', u) = [z_i]_\equiv$ und $\widehat{\delta}(z_0', v) = [z_j]_\equiv$ (da alle Zustände erreichbar)
- $\widehat{\delta}([z_i]_\equiv, w) \in E' \iff \neg(\widehat{\delta}([z_j]_\equiv, w) \in E')$ (da $[z_i]_\equiv \neq [z_j]_\equiv$)

Das Tripel (u, v, w) zeigt:

- In jedem DFA M'' mit Startzustand z_0'' und $L(M'') = L(M')$ gilt die Ungleichheit $\widehat{\delta}(z_0'', u) \neq \widehat{\delta}(z_0'', v)$.
- Dies gilt für alle Paare von unterschiedlichen Zuständen aus M'.
- Daher kann M'' nicht weniger Zustände als M' haben.

Zustandsminimierung von DFAs

Hochschule RheinMain

Idee:

- Entferne nicht-erreichbare Zustände
- Berechne äquivalente Zustände (bezüglich $\equiv$)
- Bilde Äquivalenzklassenautomat, indem äquivalente Zustände verschmolzen werden.

Berechnung äquivalenter Zustände

Hochschule RheinMain

Ideen

- Markiere Paare von Zuständen, die verschieden sein müssen
  markiere initial: alle $\{z, z'\}$ mit $z \in E$, $z' \notin E$.
- Vervollständige das Markieren durch Untersuchen von Übergängen:
  - Wenn $\{z, z'\}$ noch nicht markiert:
    Prüfe für jedes $a \in \Sigma$, ob die beiden Nachfolger $\{\delta(z, a), \delta(z', a)\}$ markiert sind.
  - Falls ja, dann markiere $\{z, z'\}$.
  - Wiederhole, bis sich nichts mehr ändert.
- Alle am Ende unmarkierten Paare sind äquivalente Zustände..

Algorithmus 2: Berechnung aller äquivalenten Zustände

Hochschule RheinMain

Eingabe: DFA $M = (Z, \Sigma, \delta, z_0, E)$, der keine unerreichbaren Zustände hat
Ausgabe: Zustandspaare $\{z, z'\}$ mit $z \neq z'$ für die gilt $z \equiv z'$
Beginn
- stelle Tabelle T aller Zustandspaare $\{z, z'\}$ mit $z \neq z'$ und $z, z' \in Z$ auf;
- markiere alle Paare $\{z, z'\}$ in T mit $z \in E$ und $z' \notin E$;
- **wiederhole**
  - **für jedes** *unmarkierte Paar* $\{z, z'\}$ *in* T **tue**
    - **für jedes** $a \in \Sigma$ **tue**
      - **wenn** $\{\delta(z, a), \delta(z', a)\}$ *in* T *markiert ist* **dann**
        - markiere $\{z, z'\}$ in T;
- **bis** *sich* T *nicht mehr verändert*;
- **return** $\{\{z, z'\} \mid \{z, z'\}$ *ist nicht markiert in* $T\}$

Korrektheit

Hochschule RheinMain

Satz
Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA, der keine unerreichbaren Zustände hat. Algorithmus 2 berechnet äquivalente Zustandspaare von M und es gibt keine weiteren äquivalenten Paare.

Beweis (Teil 1): Wird $\{z, z'\}$ markiert, dann gilt $z \not\equiv z'$.
- Wir zeigen: Ist Paar $\{z, z'\}$ markiert, dann gibt es einen Zeugen w mit $\neg(\widehat{\delta}(z, w) \in E \iff \widehat{\delta}(z', w) \in E)$
- Induktion über Anzahl Schleifeniterationen bis $\{z, z'\}$ markiert wird.
- Basis: 0 Iterationen, $\{z, z'\}$ wird vor der Schleife markiert, $w = \varepsilon$ erfüllt Behauptung.
- Schritt: Mehr als 0 Iterationen. Dann wird $\{z, z'\}$ markiert, weil es $a \in \Sigma$ und ein markiertes Paar $\{q, q'\}$ gibt mit $\delta(z, a) = q$ und $\delta(z', a) = q'$. Induktionsannahme liefert Zeugen w' mit $\neg(\widehat{\delta}(q, w') \in E \iff \widehat{\delta}(q', w') \in E)$.
- Mit $w = aw'$ folgt: $\neg(\widehat{\delta}(z, w) \in E \iff \widehat{\delta}(z', w) \in E)$.

Korrektheit (2)

Hochschule RheinMain

Satz
Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA, der keine unerreichbaren Zustände hat. Algorithmus 2 berechnet äquivalente Zustandspaare von M und es gibt keine weiteren äquivalenten Paare.

Beweis (Teil 2): Wenn $z \not\equiv z'$, dann markiert Alg. 2 das Paar $\{z, z'\}$
- Beweis durch Widerspruch. Annahme es gibt Paare $z \not\equiv z'$, die Alg. 3 nicht markiert. Wähle Paar $\{z, z'\}$, für welches es ein minimal langes Wort w gibt mit $\neg(\widehat{\delta}(z, w) \in E \iff \widehat{\delta}(z', w) \in E)$.
- Wenn $w = \varepsilon$, dann wird $\{z, z'\}$ vor Schleife markiert. Widerspruch!
- Wenn $w = aw'$ mit $a \in \Sigma$, dann gilt: Wenn $\{\delta(z, a), \delta(z', a)\}$ vom Algorithmus markiert wird, dann auch $\{z, z'\}$. Daher: $\{\delta(z, a), \delta(z', a)\}$ wird nicht markiert. Aber dann gilt für w': $\neg\left(\widehat{\delta}(\delta(z, a), w') \in E \iff \widehat{\delta}(\delta(z', a'), w') \in E\right)$ und $|w'| < |w|$.
  Widerspruch zur Minimalität!

Laufzeit

Hochschule RheinMain

- Darstellung der Tabelle T:
  zweidimensionales Array der Größe $|Z| \times |Z|$
- ermöglicht konstanten Zugriff auf Markierungen
- Pro Durchlauf der Schleife: $O(|Z|^2|\Sigma|)$
- Anzahl der Durchläufe ist durch $|Z|^2$ begrenzt, da es nur $|Z|^2$ Paare gibt und mindestens 1 Paar pro Durchlauf markiert wird
- Restliche Schritte: Konstante Laufzeit
- Daher: Algorithmus 2 kann in Zeit $O(|Z|^4 \cdot |\Sigma|)$ implementiert werden
- Tatsächlich gibt es effizientere Implementierungen: $O(|\Sigma| \cdot |Z| \log |Z|)$

Algorithmus 3: Minimierung von DFAs

Hochschule RheinMain

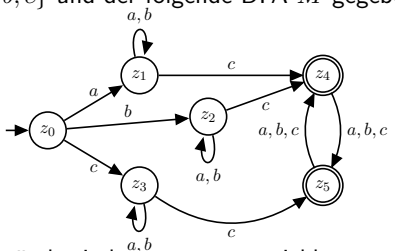
Eingabe: DFA $M = (Z, \Sigma, \delta, z_0, E)$
Ausgabe: Minimaler DFA M' mit $L(M) = L(M')$
Beginn

entferne Zustände aus M, die nicht vom Startzustand aus erreichbar sind;
berechne äquivalente Zustände mit Algorithmus 3;
erzeuge den Äquivalenzklassenautomat, indem die berechneten äquivalenten Zustände vereinigt werden;

Beispiel zur Minimierung

Hochschule RheinMain

Sei $\Sigma = \{a, b, c\}$ und der folgende DFA M gegeben:



z_1	X			
z_2	X			
z_3	X			
z_4	X	X	X	X
z_5	X	X	X	X	
	z_0	z_1	z_2	z_3	z_4

- alle Zustände sind von z_0 aus erreichbar
- äquivalente Zustände berechnen: Tabelle T erstellen
- Initiales Markieren: $\{z, z'\}$ mit $z \in \{z_0, z_1, z_2, z_3\}$ und $z' \in \{z_4, z_5\}$
- $\{z_0, z_1\}$, da $\{\delta(z_0, c), \delta(z_1, c)\} = \{z_3, z_4\}$ bereits markiert ist,
- $\{z_0, z_2\}$, da $\{\delta(z_0, c), \delta(z_2, c)\} = \{z_3, z_4\}$ bereits markiert ist, und
- $\{z_0, z_3\}$, da $\{\delta(z_0, c), \delta(z_3, c)\} = \{z_3, z_5\}$ bereits markiert

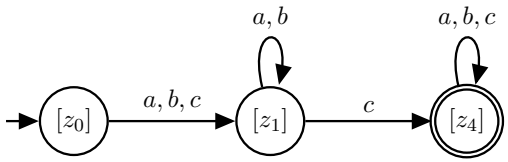
Beispiel zur Minimierung (2)

Hochschule RheinMain

Ergibt $z_1 \equiv z_3$, $z_1 \equiv z_2$, $z_2 \equiv z_3$, $z_4 \equiv z_5$ und daher die Äquivalenzklassen

$$[z_0]_{\equiv} = \{z_0\},\ [z_1]_{\equiv} = \{z_1, z_2, z_3\},\ z_{4\equiv} = \{z_4, z_5\}$$

und den Minimalautomaten



Hochschule RheinMain

DFAS AKZEPTIEREN REGULÄRE SPRACHEN

DFAs akzeptieren reguläre Sprachen

Hochschule RheinMain

Theorem 3.1.15

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA. Dann ist $L(M)$ regulär.

Beweis: Konstruiere für DFA M eine reguläre Grammatik G, sodass $L(M) = L(G)$:
Sei $G = (V, \Sigma, P, S)$ die reguläre Grammatik mit $V = Z$, $S = z_0$ und

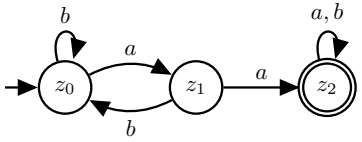
$$P = \{z_i \to az_j \mid \delta(z_i, a) = z_j\} \cup \{z_i \to a \mid \delta(z_i, a) = z_j \wedge z_j \in E\} \cup \{z_0 \to \varepsilon \mid \text{falls } z_0 \in E\}$$

Leeres Wort: Offensichtlich gilt $\varepsilon \in L(M) \iff \varepsilon \in L(G)$.
Wörter w mit $|w| = m \geq 1$: $w = a_1 \cdots a_m \in L(M)$
g.d.w. es gibt $z_1, \ldots, z_m \in Z$ mit $\delta(z_{i-1}, a_i) = z_i$ und $z_m \in E$.
g.d.w. $z_0 \Rightarrow_G a_1 z_1$, für $1 \leq i < m$: $a_1 \cdots a_{i-1} z_{i-1} \Rightarrow_G a_1 \cdots a_i z_i$ und
$a_1 \cdots a_{m-1} z_{m-1} \Rightarrow_G a_1 \cdots a_m$, d.h. $z_0 \Rightarrow_G^* a_1 \cdots a_m$
g.d.w. $w = a_1 \cdots a_m \in L(G)$
Daher gilt $L(M) = L(G)$ und somit ist $L(M)$ regulär. □

Quiz 5

Hochschule RheinMain

Erzeugen Sie die Produktionen einer regulären Grammatik G mit $L(G) = L(A)$ wobei A der gezeigte DFA ist:



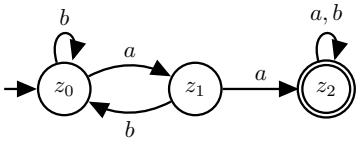
Zur Erinnerung:

$$P = \{z_i \to az_j \mid \delta(z_i, a) = z_j\} \cup \{z_i \to a \mid \delta(z_i, a) = z_j \wedge z_j \in E\} \cup \{z_0 \to \varepsilon \mid \text{falls } z_0 \in E\}$$

arsnova.hs-rm.de
6750 1376

Reguläre Grammatik zum DFA

Hochschule RheinMain



DFA $M = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_2\})$ mit

$$\begin{array}{ll} \delta(z_0, a) = z_1 & \delta(z_1, b) = z_0 \\ \delta(z_0, b) = z_0 & \delta(z_2, a) = z_2 \\ \delta(z_1, a) = z_2 & \delta(z_2, b) = z_2 \end{array}$$

Die erzeugte reguläre Grammatik dazu ist:

$$G = (\{z_0, z_1, z_2\}, \{a, b\}, P, z_0) \text{ mit}$$
$$P = \{z_0 \to az_1 \mid bz_0,$$
$$z_1 \to az_2 \mid a \mid bz_0,$$
$$z_2 \to az_2 \mid a \mid bz_2 \mid b\}$$

Wird jede reguläre Sprache durch einen DFA akzeptiert?

Hochschule RheinMain

- Der vorherige Beweis konstruiert:
  „für jeden DFA gibt es eine äquivalente reguläre Grammatik“
- Für die andere Richtung wäre notwendig
  „für jede reguläre Grammatik gibt es einen äquivalenten DFA“

Problem:

- Produktionen: $A \to aA_1$ und $A \to aA_2$ können in Grammatiken vorkommen
- Konstruktion des deterministischen Automaten zunächst unklar

Daher: Beweis, dass DFAs alle regulären Sprachen akzeptieren, erfolgt auf Umwegen und verwendet nichtdeterministische endliche Automaten

Hochschule RheinMain

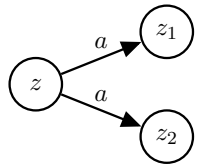
NICHTDETERMINISTISCHE ENDLICHE AUTOMATEN

Nichtdeterministische Endliche Automaten

Hochschule RheinMain

Ideen:

- Zustandswechsel nicht eindeutig, sondern nichtdeterministisch in einen von mehreren möglichen
- D.h. der Automat darf sozusagen „raten“, welchen Nachfolgezustand er wählt
- Im Zustandsgraph erlaubt:



- Technisch:
  - DFA $\delta : Z \times \Sigma \to Z$ und ein Startzustand
  - NFA $\delta : Z \times \Sigma \to \mathcal{P}(Z)$ und Menge von Startzuständen

Definition NFA

Hochschule RheinMain

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.

Akzeptanz beim NFA

Hochschule RheinMain

„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

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

Beispiel: Leere Menge von Startzuständen

Hochschule RheinMain

Sei $M = (Z, \Sigma, \delta, \emptyset, E)$ ein NFA.
Dann ist $L(M) = \emptyset$.

Lauf beim NFA

Hochschule RheinMain

Definition

Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA und $w \in \Sigma^*$ mit $|w| = n$.
Eine Folge von Zuständen $q_0, \ldots, q_n$ mit $q_0 \in S$ und $q_i \in \delta(q_{i-1}, w[i])$ bezeichnet man als **Lauf** von M für Wort w.

Ein Lauf der mit einem Endzustand endet, nennen wir auch **akzeptierender Lauf**.

Beachte: Während es bei DFAs genau einen Lauf pro Wort gibt, kann es bei NFAs mehrere geben.

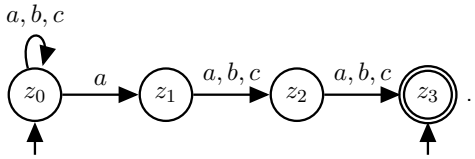
Beispiel

Hochschule RheinMain

Sei $M = (\{z_0, z_1, z_2, z_3\}, \{a, b, c\}, \delta, \{z_0, z_3\}, \{z_3\})$ ein NFA mit

$$\begin{array}{llll} \delta(z_0, a) = \{z_0, z_1\} & \delta(z_1, a) = \{z_2\} & \delta(z_2, a) = \{z_3\} & \delta(z_3, a) = \emptyset \\ \delta(z_0, b) = \{z_0\} & \delta(z_1, b) = \{z_2\} & \delta(z_2, b) = \{z_3\} & \delta(z_3, b) = \emptyset \\ \delta(z_0, c) = \{z_0\} & \delta(z_1, c) = \{z_2\} & \delta(z_2, c) = \{z_3\} & \delta(z_3, c) = \emptyset \end{array}$$

Der Zustandsgraph zu M ist



.

$$L(M) = \{\varepsilon\} \cup \{uaw \mid u \in \{a, b, c\}^*, w \in \{a, b, c\}^2\}$$

Hochschule RheinMain

REGULÄRE SPRACHEN SIND DURCH NFAS ERKENNBAR

Jede reguläre Sprache wird durch einen NFA erkannt

Hochschule RheinMain

Theorem 3.3.1

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

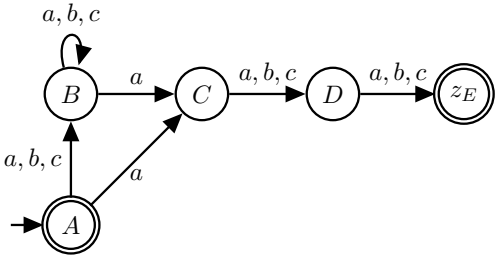
Hochschule RheinMain

Betrachte die reguläre Grammatik

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

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

Konstruktion des dazu passenden NFA:



Hochschule RheinMain

DETERMINISIERUNG VON NFAs

- Potenzmengenkonstruktion
- Jede reguläre Sprache ist durch DFA erkennbar
- Größenvergleich: NFA vs. DFA

NFAs in DFAs transformieren

Hochschule RheinMain

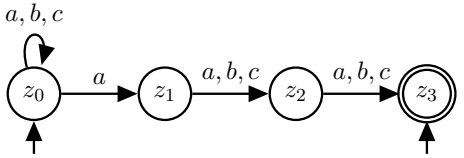
Theorem 3.4.1 (Rabin & Scott 1959)

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)*

Quiz 6

Hochschule RheinMain

Die Potenzmenge von $\{q_0, q_1, q_2\}$ ist...?

arsnova.hs-rm.de
6750 1376

Potenzmengenkonstruktion

Hochschule RheinMain

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

Hochschule RheinMain

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

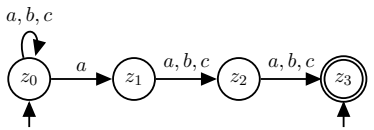
$$w \in L(M) \text{ g.d.w. } w \in L(M')$$

- Fall $w = \varepsilon$: $\varepsilon \in L(M)$ g.d.w. $S \cap E \neq \emptyset$ g.d.w $S \in E'$ g.d.w. $\varepsilon \in L(M')$
- Fall $w = a_1 \cdots a_n \in \Sigma^*$:

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

Beispiel: Potenzmengenkonstruktion

Hochschule RheinMain

Für NFA 

wird der DFA konstruiert:

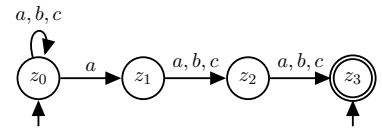
$M' = (\mathcal{P}(\{z_0, z_1, z_2, z_3\}), \{a, b, c\}, \delta', S', E')$ mit $S' = \{z_0, z_3\}$

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

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

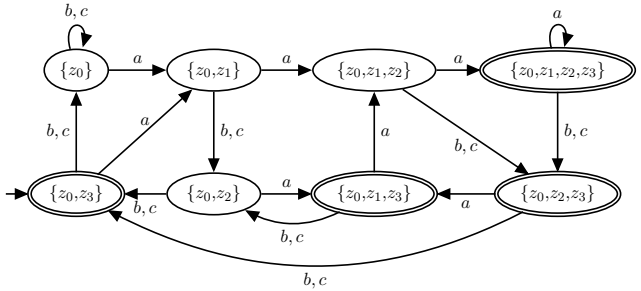
Beispiel: Potenzmengenkonstruktion (2)

Hochschule RheinMain



Ablauf bei Eingabe babbabb

DFA als Zustandsgraph (nur erreichbare Zustände):



DFAs & NFAs sind Formalismen für Typ 3-Sprachen

Hochschule RheinMain

Theorem 3.4.4

DFAs und NFAs erkennen genau die regulären Sprachen.

Das folgt aus:

- Theorem 3.1.15: Sei M ein DFA. Dann ist $L(M)$ regulär.
- Theorem 3.3.1: Für jede reguläre Sprache L gibt es einen NFA M mit $L(M) = L$.
- Theorem 3.4.1: Jede von einem NFA akzeptierte Sprache ist auch durch einen DFA akzeptierbar.
- Jeder DFA kann leicht auch als NFA interpretiert werden

Größe DFAs vs NFAs (1)

Hochschule RheinMain

- 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

Größe DFAs vs NFAs (2)

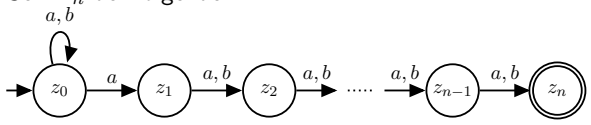
Hochschule RheinMain

Lemma

Sei $L_n = \{uav \mid u \in \{a,b\}^*, v \in \{a,b\}^{n-1}\}$ für $n \in \mathbb{N}$.
(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 (Teil 1): Sei M_n der folgende NFA:



$L(M_n) = L_n$, denn:

- zum Akzeptieren müssen $z_0, z_1, \ldots z_n$ nacheinander durchlaufen werden, was genau mit Wörtern av mit $v \in \{a,b\}^{n-1}$ möglich ist
- In z_0 kann zuvor jedes $u \in \{a,b\}^*$ gelesen werden (Verbleib in z_0).

Größe des DFAs vs NFAs (3)

Hochschule RheinMain

Beweis (Teil 2): Beweis durch Widerspruch.

- Annahme: Es gibt $n \in \mathbb{N}$ und DFA $M' = (Z, \{a,b\}, \delta, z_0, E)$ mit $L(M') = L_n = \{uav \mid u \in \{a,b\}^*, v \in \{a,b\}^{n-1}\}$ und $|Z| < 2^n$
- Menge $W = \{a,b\}^n$ enthält 2^n Wörter der Länge n und da $|Z| < 2^n$, muss es $w \neq w' \in W$ geben mit $\widehat{\delta}(z_0, w) = \widehat{\delta}(z_0, w') = z_i$
- Sei j die erste Position, an der sich w und w' unterscheiden.

  Falls $j = 1$, dann ist o.B.d.A. $w = au \in L_n$ aber $w' = bu' \notin L_n$ und $z_i \in E$ und $z_i \notin E$ müsste gleichzeitig gelten. **Widerspruch!**

  Falls $j > 1$: O.B.d.A. $w = uav$ und $w' = ubv'$ mit $|v| = |v'| = n - j$

  Sei $w_0 = wb^{j-1} = uavb^{j-1}$
  $w'_0 = w'b^{j-1} = ubv'b^{j-1}$

  Dann muss gelten $\widehat{\delta}(w_0) = \widehat{\delta}(w'_0)$, da $\widehat{\delta}(uav) = z_i = \widehat{\delta}(ubv')$.

  Aber $w_0 \in L_n$ und $w'_0 \notin L_n$, **Widerspruch!** □

NFAS MIT ε-ÜBERGÄNGEN

Hochschule RheinMain

NFAs mit ε-Übergängen

Hochschule RheinMain

- ε-Übergänge erlauben Zustandswechsel ohne Lesen eines Zeichens (es wird sozusagen das leere Wort ε gelesen)
- Ausdruckskraft ändert sich mit ε-Übergängen nicht
- ε-Übergänge machen manche Konstruktionen einfacher.

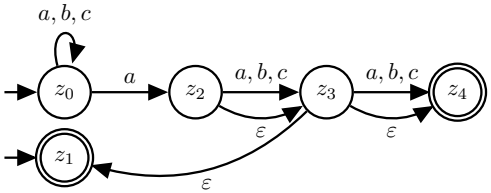
Definition (NFA mit ε-Übergängen)

Ein nichtdeterministischer endlicher Automat mit ε-Übergängen (NFA mit ε-Übergängen) ist ein Tupel $M = (Z, \Sigma, \delta, S, E)$ wobei

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

Quiz 7

Hochschule RheinMain



Welche Sprache akzepiert der Automat?

arsnova.hs-rm.de
6750 1376

ε-Hülle

Hochschule RheinMain

Definition (ε-Hülle)

Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA mit ε-Übergängen. Die ε-Hülle $clos_\varepsilon(z)$ eines Zustands $z \in Z$ ist induktiv definiert als die kleinste Menge von Zuständen, welche die folgenden Eigenschaften erfüllt:

1. $z \in clos_\varepsilon(z)$.
2. Wenn $z' \in clos_\varepsilon(z)$ und $z'' \in \delta(z', \varepsilon)$, dann ist auch $z'' \in clos_\varepsilon(z)$.

Für eine Zustandsmenge $X \subseteq Z$ definieren wir $clos_\varepsilon(X) := \bigcup_{z \in X} clos_\varepsilon(z)$.

Die ε-Hülle fügt für eine Zustandsmenge alle durch ε-Übergänge erreichbaren Zustände hinzu.

ε-Hülle (2)

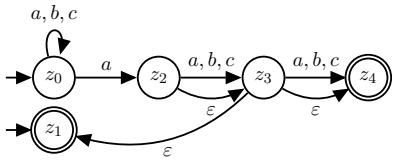
Hochschule RheinMain

Die ε-Hülle für eine Zustandsmenge $X \subseteq Z$ kann auch berechnet werden durch:

$$clos_\varepsilon(X) := \begin{cases} X, & \text{wenn } \bigcup_{z \in X} \delta(z, \varepsilon) \subseteq X \\ clos_\varepsilon(X \cup \bigcup_{z \in X} \delta(z, \varepsilon)), & \text{sonst} \end{cases}$$

Beispiel

Hochschule RheinMain



$$clos_\varepsilon(z_0) = \{z_0\}$$
$$clos_\varepsilon(z_1) = \{z_1\}$$
$$clos_\varepsilon(z_4) = \{z_4\}$$
$$clos_\varepsilon(z_3) = \{z_1, z_3, z_4\}$$
$$clos_\varepsilon(z_2) = \{z_1, z_2, z_3, z_4\}$$

NFA mit ε-Übergängen: Akzeptierte Sprache

Hochschule RheinMain

Akzeptierte Sprache eines NFA mit ε-Übergängen

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

$$\widetilde{\delta}(X, \varepsilon) := X$$
$$\widetilde{\delta}(X, aw) := \bigcup_{z \in X} \widetilde{\delta}(clos_\varepsilon(\delta(z, a)), w) \text{ für alle } X \subseteq Z$$

Die von M akzeptierte Sprache ist

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

ε-Übergänge ändern die Ausdruckskraft nicht (1)

Hochschule RheinMain

Satz 3.5.7

NFAs mit ε-Übergängen akzeptieren genau die regulären Sprachen.

Beweis „$\Leftarrow$“:

- Jede reguläre Sprache wird von einem „normalen“ NFA akzeptiert.
- Transformiere diesen NFA in einen NFA mit ε-Übergängen:

  Setze $\delta(z, \varepsilon) = \emptyset$ für alle Zustände z

  Offensichtlich ist die akzeptierte Sprache dieselbe.
- Daher wird jede reguläre Sprache von einem NFA mit ε-Übergängen akzeptiert.

ε-Übergänge ändern die Ausdruckskraft nicht (2)

Hochschule RheinMain

Beweis „$\Rightarrow$“: Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA mit ε-Übergängen.

- Konstruiere NFA M' mit $L(M) = L(M')$. Dann ist $L(M)$ regulär.
- $M' = (Z, \Sigma, \delta', S', E)$ mit $S' = clos_\varepsilon(S)$, $\delta'(z, a) = clos_\varepsilon(\delta(z, a))$.

$L(M) = L(M')$:

- Wir zeigen $\widetilde{\delta}(clos_\varepsilon(X), w) = \widehat{\delta'}(clos_\varepsilon(X), w)$ für alle $X \subseteq Z$ und $w \in \Sigma^*$. Wir verwenden Induktion über die Wortlänge $|w|$.
- Basis: $w = \varepsilon$. Dann gilt $\widetilde{\delta}(clos_\varepsilon(X), \varepsilon) = clos_\varepsilon(X) = \widehat{\delta'}(clos_\varepsilon(X), \varepsilon)$
- Schritt: Sei $w = au$ mit $a \in \Sigma$. Wir formen um:

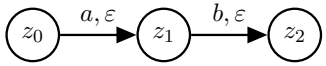
$$\widetilde{\delta}(clos_\varepsilon(X), au) \overset{\text{Def. } \widetilde{\delta}}{=} \bigcup_{z \in clos_\varepsilon(X)} \widetilde{\delta}(clos_\varepsilon(\delta(z, a)), u) \overset{\text{I.H.}}{=} \bigcup_{z \in clos_\varepsilon(X)} \widehat{\delta'}(clos_\varepsilon(\delta(z, a)), u)$$

$$\overset{\text{Def. } \delta'}{=} \bigcup_{z \in clos_\varepsilon(X)} \widehat{\delta'}(\delta'(z, a), u) \overset{\text{Def. } \widehat{\delta}}{=} \widehat{\delta'}(clos_\varepsilon(X), au)$$

Nachtrag: Entfernen der ε-Übergänge

Hochschule RheinMain

- Die Konstruktion ist anders, als das letzte Mal vorgeführt:

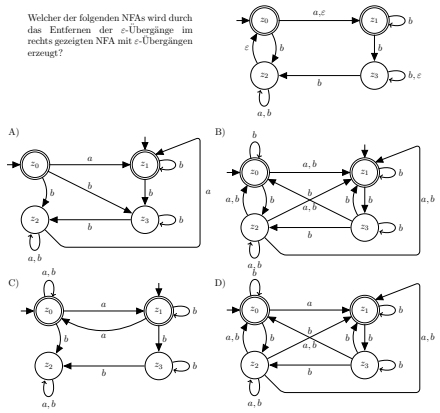


  Dann: $\delta(z_0, a) = clos_\varepsilon(\{z_1\}) = \{z_1, z_2\}$

  Aber: $\delta(z_0, b) = clos_\varepsilon(\emptyset) = \emptyset$
- D.h. erst a-Übergang benutzen, dann alles was über ε erreichbar ist.
  Nicht umgekehrt!

Quiz 8

Hochschule RheinMain



arsnova.hs-rm.de

6750 1376

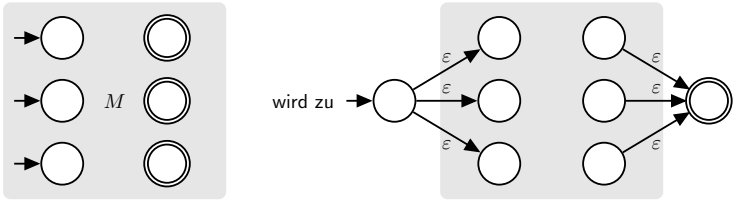
Eindeutige Start- und Endzustände

Hochschule RheinMain

Satz 4.6.8

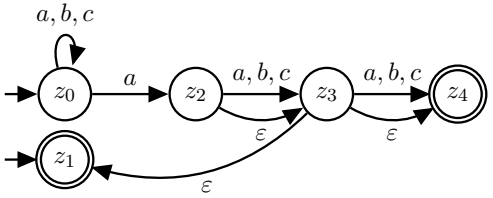
Für jeden NFA M mit ε-Übergängen gibt es einen NFA M' mit ε-Übergängen, sodass $L(M) = L(M')$ und M' genau einen Startzustand und genau einen Endzustand hat, wobei diese beiden Zustände verschieden sind.

Beweis: Konstruiere M' aus M, durch Hinzufügen eines neuen Start- und eines neuen Endzustand mit ε-Übergängen:



Beispiel

Hochschule RheinMain



wird zu