Theoretische Informatik

Kapitel 2 – Reguläre Sprachen

Sommersemester 2024

Dozentin: Mareike Mutz

im Wechsel mit

Prof. Dr. M. Leuschel

Prof. Dr. J. Rothe

Beispiel 1

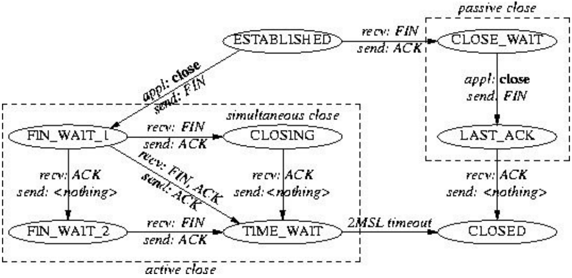


Abbildung: TCP

Beispiel 2

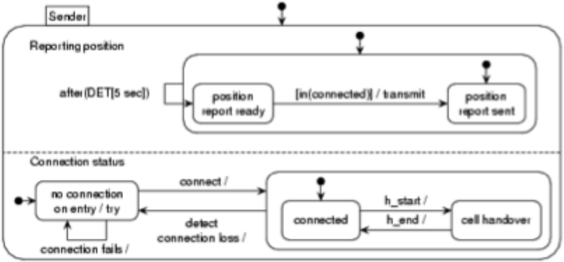


Abbildung: European Train Control System - Spezifikation

Beispiel 3

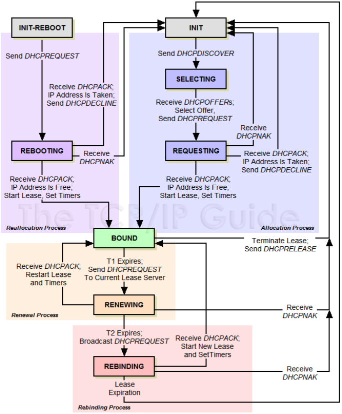


Abbildung: DHCP

Endliche Automaten

Definition

Ein *deterministischer endlicher Automat* (kurz DFA für *"deterministic finite automaton"*) ist ein Quintupel $M = (\Sigma, Z, \delta, z_0, F)$, wobei

- Σ ein Alphabet ist,
- Z eine endliche Menge von Zuständen mit $\Sigma \cap Z = \emptyset$,
- $\delta : Z \times \Sigma \to Z$ die Überführungsfunktion,
- $z_0 \in Z$ der Startzustand und
- $F \subseteq Z$ die Menge der Endzustände (Finalzustände).

Bemerkung: Wir erlauben auch partiell definierte Überführungsfunktionen. Durch Hinzunahme eines weiteren Zustandes kann man solche Funktionen leicht global definieren.

Endliche Automaten

Beispiel: Ein Beispiel für einen DFA ist $M = (\Sigma, Z, \delta, z_0, F)$ mit

$$\Sigma = \{0, 1\}$$
$$Z = \{z_0, z_1, z_2, z_3\}$$
$$F = \{z_2\}$$

δ :

δ	z_0	z_1	z_2	z_3
0	z_1	z_3	z_2	z_3
1	z_3	z_2	z_2	z_3

Arbeitsweise eines DFAs

Ein DFA $M = (\Sigma, Z, \delta, z_0, F)$ akzeptiert bzw. verwirft eine Eingabe x wie folgt:

- M beginnt beim Anfangszustand z_0 und führt insgesamt $|x|$ Schritte aus.
- Der Lesekopf wandert dabei v.l.n.r. über das Eingabewort x, Symbol für Symbol, und ändert dabei seinen Zustand jeweils gemäß der Überführungsfunktion δ:
  Ist M im Zustand $z \in Z$ und liest das Symbol $a \in \Sigma$ und gilt $\delta(z, a) = z'$, so ändert M seinen Zustand in z'.
- Ist der letzte erreichte Zustand (nachdem x abgearbeitet ist)
  - ein Endzustand, so akzeptiert M die Eingabe x;
  - andernfalls lehnt M sie ab.

Arbeitsweise eines DFAs

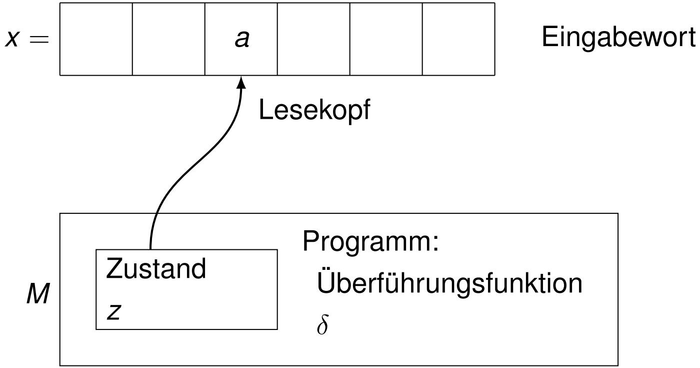


Abbildung: Arbeitsweise eines endlichen Automaten

Zustandsgraph eines DFAs

Ein DFA $M = (\Sigma, Z, \delta, z_0, F)$ lässt sich anschaulich durch seinen *Zustandsgraphen* darstellen,

- dessen Knoten die Zustände von M und
- dessen Kanten Zustandsübergänge gemäß der Überführungsfunktion δ repräsentieren.
- Gilt $\delta(z, a) = z'$ für ein Symbol $a \in \Sigma$ und für zwei Zustände $z, z' \in Z$, so hat dieser Graph eine gerichtete Kante von z nach z', die mit a beschriftet ist.
- Der Startzustand wird durch einen Pfeil auf z_0 dargestellt.
- Endzustände sind durch einen Doppelkreis markiert.

Zustandsgraph eines DFAs

Beispiel:

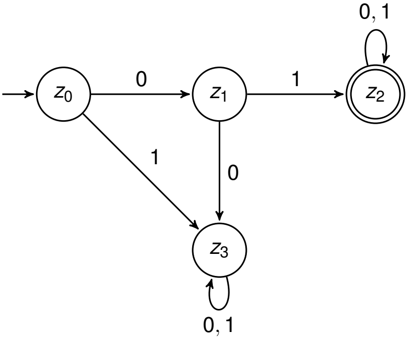


Abbildung: Zustandsgraph eines deterministischen endlichen Automaten

Sprache eines DFAs

Definition

Sei $M = (\Sigma, Z, \delta, z_0, F)$ ein DFA. Die *erweiterte Überführungsfunktion* $\widehat{\delta} : Z \times \Sigma^* \to Z$ von M ist induktiv definiert:

$$\begin{aligned}\widehat{\delta}(z, \lambda) &= z \\ \widehat{\delta}(z, ax) &= \widehat{\delta}(\delta(z, a), x)\end{aligned}$$

für alle $z \in Z$, $a \in \Sigma$ und $x \in \Sigma^*$.

Die *vom DFA M akzeptierte Sprache* ist definiert durch

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

Sprache eines DFAs

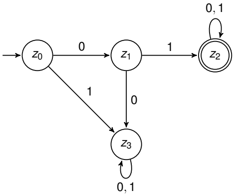
Bemerkung:

- Für $a \in \Sigma$ gilt $\widehat{\delta}(z, a) = \delta(z, a)$, und
- für $x = a_1 a_2 \cdots a_n$ in Σ^* gilt:

$$\widehat{\delta}(z, x) = \delta(\cdots \delta(\delta(z, a_1), a_2) \cdots , a_n).$$

Sprache eines DFAs

Beispiel: Wie arbeitet der DFA aus der vorigen Abbildung:

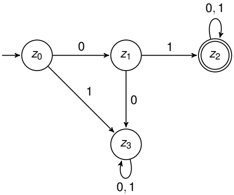


bei Eingabe von 0111?

$$
\begin{array}{rcl}
\widehat{\delta}(z_0, 0111) & = & \delta(\delta(\delta(\delta(z_0, 0), 1), 1), 1) \\
& = & \delta(\delta(\delta(z_1, 1), 1), 1) \\
& = & \delta(\delta(z_2, 1), 1) \\
& = & \delta(z_2, 1) \\
& = & z_2 \qquad \in \quad F.
\end{array}
$$

Sprache eines DFAs

Beispiel: Wie arbeitet der DFA aus der vorigen Abbildung:

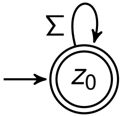
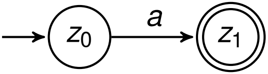


bei Eingabe von 0011?

$$
\begin{aligned}
\widehat{\delta}(z_0, 0011) &= \delta(\delta(\delta(\delta(z_0, 0), 0), 1), 1) \\
&= \delta(\delta(\delta(z_1, 0), 1), 1) \\
&= \delta(\delta(z_3, 1), 1) \\
&= \delta(z_3, 1) \\
&= z_3 \quad \notin \quad F.
\end{aligned}
$$

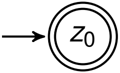
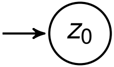
Sprache eines DFAs

Beispiel: Es sei Σ ein Alphabet.

L	DFA M mit $L(M) = L$
Σ^*	$(\Sigma, \{z_0\}, \{\delta(z_0, a) = z_0, a \in \Sigma\}, z_0, \{z_0\})$ 
$\{a\}$, $a \in \Sigma$	$(\Sigma, \{z_0, z_1\}, \{\delta(z_0, a) = z_1\}, z_0, \{z_1\})$ 

Sprache eines DFAs

Beispiel:

L	DFA M mit $L(M) = L$
$\{\lambda\}$	$(\Sigma, \{z_0\}, \emptyset, z_0, \{z_0\})$ 
$\emptyset$	$(\Sigma, \{z_0\}, \emptyset, z_0, \emptyset)$ 

Wiederholung: reguläre Grammatiken

Eine Typ-2-Grammatik G ist *vom Typ 3* (bzw. *regulär* bzw. *rechtslinear*), falls für alle Regeln $p \rightarrow q$ in P gilt: $p \in N$ und $q \in \Sigma \cup \Sigma N$.

DFAs und reguläre Grammatiken

Theorem

Jede von einem DFA akzeptierte Sprache ist regulär (d.h. vom Typ 3).

Beweis: Sei $M = (\Sigma, Z, \delta, z_0, F)$ ein DFA mit der akzeptierten Sprache $L(M)$. Definiere eine reguläre Grammatik $G = (\Sigma, N, S, P)$:

- $N = Z$,
- $S = z_0$,
- P besteht aus genau den folgenden Regeln:
  - Gilt $\delta(z, a) = z'$, so ist $z \rightarrow az'$ in P.
  - Ist dabei $z' \in F$, so ist zusätzlich $z \rightarrow a$ in P.
  - Ist $\lambda \in L(M)$ (d.h., $z_0 \in F$), so ist auch $z_0 \rightarrow \lambda$ in P, und die bisher konstruierte Grammatik wird gemäß der Sonderregel für λ modifiziert.

DFAs und reguläre Grammatiken

Offenbar gilt $\lambda \in L(M) \iff \lambda \in L(G)$.

Für alle $x = a_1 a_2 \cdots a_n \in \Sigma^*$ mit $n \geq 1$ gilt:

$x \in L(M) \iff$ es gibt eine Zustandsfolge $z_0, z_1, \ldots, z_n$, so dass

z_0 Startzustand und $z_n \in F$ ist, und

$\delta(z_{i-1}, a_i) = z_i$ für jedes i, $1 \leq i \leq n$

$\iff$ es gibt eine Folge von Nichtterminalen $z_0, z_1, \ldots, z_n$ mi

$$S = z_0 \vdash a_1 z_1 \vdash a_1 a_2 z_2 \vdash \cdots$$

$$\cdots \vdash a_1 a_2 \cdots a_{n-1} z_{n-1} \vdash a_1 a_2 \cdots a_{n-1} a_n$$

(d.h. $z_{i-1} \rightarrow a_i z_i \in P$)

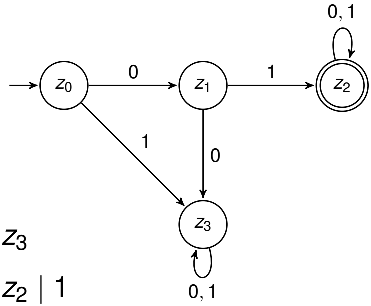
$\iff x \in L(G)$.

Somit ist $L(M) = L(G)$. ❑

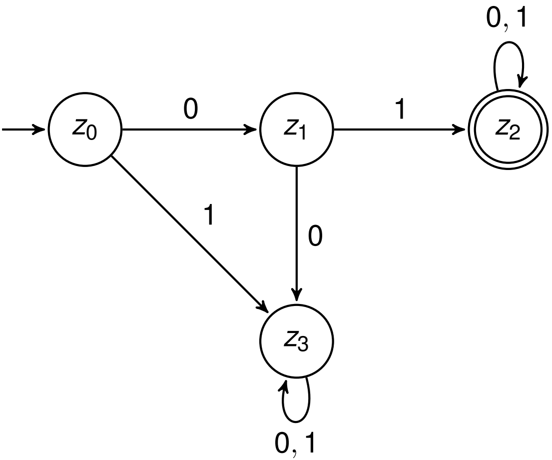
DFAs und reguläre Grammatiken

Beispiel: Reguläre Grammatik $G = (\Sigma, N, S, P)$ für diesen DFA:

$$\begin{array}{rcl}
\Sigma & = & \{0, 1\} \\
N & = & \{z_0, z_1, z_2, z_3\} \\
S & = & z_0 \\
P & = & \{ \quad z_0 \rightarrow 0z_1 \mid 1z_3 \\
 & & \quad\;\; z_1 \rightarrow 0z_3 \mid 1z_2 \mid 1 \\
 & & \quad\;\; z_2 \rightarrow 0z_2 \mid 1z_2 \mid 0 \mid 1 \\
 & & \quad\;\; z_3 \rightarrow 0z_3 \mid 1z_3 \qquad \}
\end{array}$$



DFAs als Code



Wie kann man diesen DFA und $\widehat{\delta}$ in Java umsetzen?

DFAs als Code Teil 1

```
public class DFA {
  static int initial_state = 0;
   public static int delta (int state, char next) {
    switch(state) {
       case 0: if (next == '0') return 1; else return 3;
       case 1: if (next == '1') return 2; else return 3;
       case 2: if (next == '1' || next == '0') return 2; else retu
       default: return 3;
    }
  }
  static boolean isFinal (int state) {
     return (state==2);
  }
 ...
```

DFAs als Code Teil 2

```
public static int deltas(String s) { // delta hut
  int state = initial_state;
  for (int i = 0; i < s.length(); i++){
     state = delta(state,s.charAt(i));
  }
  return state;
}
public static boolean accept(String s) {
  return isFinal(deltas(s));
}
public static void main(String[] args) {
      if (args.length > 0) {
         System.out.println("Accepted: " + accept(args[0]));
      }
 }
```

DFAs als Prolog Programm

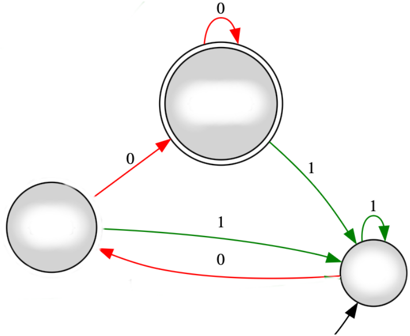
```
delta(s0,0,s1).
delta(s1,1,s2).
delta(s2,0,s2). delta(s2,1,s2).
initial(s0).
final(s2).

% Definition von delta^:
deltas(S,[],S).
deltas(S,[H|T],Sn) :- delta(S,H,S1), deltas(S1,T,Sn).

accept(Word) :- initial(S0),deltas(S0,Word,Sn), final(Sn).
```

Übung: DFA

Schreiben Sie einen DFA M der Binärstrings von Zahlen darstellt die durch 4 teilbar sind.
Also 11 $\notin L(M)$ und 1000 $\in L(M)$ (8 durch 4 teilbar).

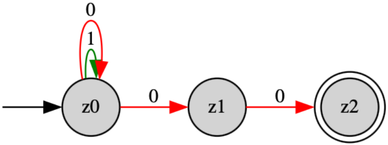
Übung

Ein DFA der Binärstrings akzeptiert die durch 4 teilbar sind.



Übung

Ein anderer Automat der Binärstrings akzeptiert die durch 4 teilbar sind.



Was ist hier anders?

Wiederholung: DFAs

Ein DFA ist ein Quintupel $M = (\Sigma, Z, \delta, z_0, F)$, wobei

- Σ ein Alphabet ist,
- Z eine endliche Menge von Zuständen mit $\Sigma \cap Z = \emptyset$,
- $\delta : Z \times \Sigma \to Z$ die Überführungsfunktion,
- $z_0 \in Z$ der Startzustand und
- $F \subseteq Z$ die Menge der Endzustände (Finalzustände).

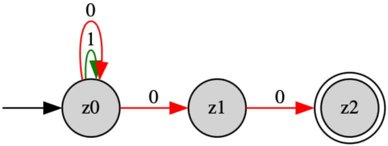
Die vom DFA M akzeptierte Sprache ist definiert durch

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

Wiederholung: DFAs

Wir haben gezeigt: Jede von einem DFA akzeptierte Sprache ist regulär.

Wir wollen (später) zeigen, dass auch jede reguläre Sprache von einem DFA akzeptiert werden kann.



Nichtdeterministischer endlicher Automat

Definition

Ein *nichtdeterministischer endlicher Automat* (kurz NFA) ist ein Quintupel $M = (\Sigma, Z, \delta, S, F)$, wobei

- Σ ein Alphabet ist,
- Z eine endliche Menge von Zuständen mit $\Sigma \cap Z = \emptyset$,
- $\delta : Z \times \Sigma \to \mathfrak{P}(Z)$ die Überführungsfunktion (hier: $\mathfrak{P}(Z)$ ist die Potenzmenge von Z, also die Menge aller Teilmengen von Z),
- $S \subseteq Z$ die Menge der Startzustände und
- $F \subseteq Z$ die Menge der Endzustände (Finalzustände).

Nichtdeterministischer endlicher Automat

Beispiel: Der folgende Automat M ist nichtdeterministisch (da $\|\delta(z_0, 1)\| = \|\{z_0, z_1\}\| = 2 > 1$).

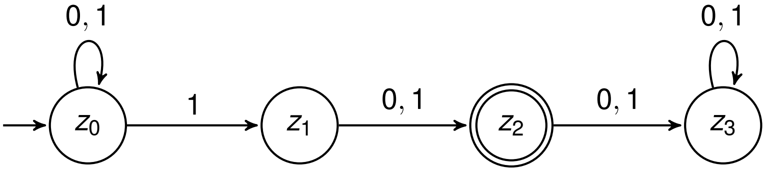


Abbildung: Ein nichtdeterministischer endlicher Automat

Sprache eines NFAs

Definition

Die *erweiterte Überführungsfunktion* $\widehat{\delta} : \mathfrak{P}(Z) \times \Sigma^* \to \mathfrak{P}(Z)$ von M ist induktiv definiert:

$$\begin{array}{rcl} \widehat{\delta}(Z', \lambda) & = & Z' \\ \widehat{\delta}(Z', ax) & = & \bigcup\limits_{z \in Z'} \widehat{\delta}(\delta(z, a), x) \end{array}$$

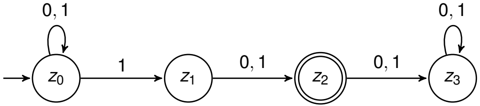
für alle $Z' \subseteq Z$, $a \in \Sigma$ und $x \in \Sigma^*$.

Die *vom NFA M akzeptierte Sprache* ist definiert durch

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

Sprache eines NFAs

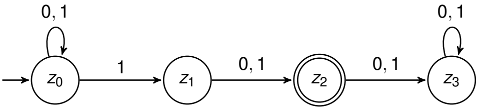
Beispiel: NFA M



bei Eingabe von 0111:

$$
\begin{aligned}
\widehat{\delta}(\{z_0\}, 0111) &= \widehat{\delta}(\delta(z_0, 0), 111) = \widehat{\delta}(\{z_0\}, 111) \\
&= \widehat{\delta}(\delta(z_0, 1), 11) = \widehat{\delta}(\{z_0, z_1\}, 11) \\
&= \widehat{\delta}(\delta(z_0, 1), 1) \cup \widehat{\delta}(\delta(z_1, 1), 1) \\
&= \widehat{\delta}(\{z_0, z_1\}, 1) \cup \widehat{\delta}(\{z_2\}, 1) \\
&= (\widehat{\delta}(\delta(z_0, 1), \lambda) \cup \widehat{\delta}(\delta(z_1, 1), \lambda)) \cup \widehat{\delta}(\delta(z_2, 1), \lambda) \\
&= (\widehat{\delta}(\{z_0, z_1\}, \lambda) \cup \widehat{\delta}(\{z_2\}, \lambda)) \cup \widehat{\delta}(\{z_3\}, \lambda) \\
&= \{z_0, z_1\} \cup \{z_2\} \cup \{z_3\}.
\end{aligned}
$$

Sprache eines NFAs



Beispiel: NFA M

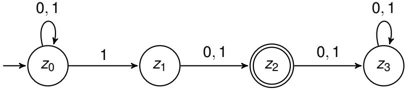
bei Eingabe von 0111:

$$\Longrightarrow \quad \widehat{\delta}(\{z_0\}, 0111) \cap \{z_2\} = (\{z_0, z_1\} \cup \{z_2\} \cup \{z_3\}) \cap \{z_2\} \neq \emptyset$$

$$\Longrightarrow \quad 0111 \in L(M).$$

Sprache eines NFAs

Beispiel: NFA M



bei Eingabe von 01:

$$
\begin{aligned}
\widehat{\delta}(\{z_0\}, 01) &= \widehat{\delta}(\delta(z_0, 0), 1) \\
&= \widehat{\delta}(\{z_0\}, 1) \\
&= \widehat{\delta}(\delta(z_0, 1), \lambda) \\
&= \widehat{\delta}(\{z_0, z_1\}, \lambda) \\
&= \{z_0, z_1\}.
\end{aligned}
$$

$$
\begin{aligned}
&\Longrightarrow \quad \widehat{\delta}(\{z_0\}, 01) \cap \{z_2\} = \{z_0, z_1\} \cap \{z_2\} = \emptyset \\
&\Longrightarrow \quad 01 \notin L(M).
\end{aligned}
$$

$L(M) = \{w \in \{0,1\}^* \mid w = u1v \text{ mit } u \in \{0,1\}^* \text{ und } v \in \{0,1\}\}$.

DFA versus NFA, Satz von Rabin und Scott

Theorem (Rabin und Scott)

Jede von einem NFA akzeptierte Sprache kann auch von einem DFA akzeptiert werden.

Beweis: Sei $M = (\Sigma, Z, \delta, S, E)$ ein NFA. Konstruiere einen zu M äquivalenten DFA $M' = (\Sigma, \mathfrak{P}(Z), \delta', z'_0, F)$ wie folgt:

- Zustandsmenge von M': die Potenzmenge $\mathfrak{P}(Z)$ von Z,
- $\delta'(Z', a) = \bigcup_{z \in Z'} \delta(z, a) = \widehat{\delta}(Z', a)$ für alle $Z' \subseteq Z$ und $a \in \Sigma$,
- $z'_0 = S$,
- $F = \{Z' \subseteq Z \mid Z' \cap E \neq \emptyset\}$.

DFA versus NFA, Satz von Rabin und Scott

Offenbar sind der DFA M' und der NFA M äquivalent, denn für alle $x = a_1 a_2 \cdots a_n$ in Σ^* gilt:

$$
\begin{aligned}
x \in L(M) \quad & \Longleftrightarrow \quad \widehat{\delta}(S, x) \cap E \neq \emptyset \\
& \Longleftrightarrow \quad \text{es gibt eine Folge von Teilmengen} \\
& \qquad\quad Z_1, Z_2, \ldots, Z_n \subseteq Z \text{ mit } \delta'(S, a_1) = Z_1, \\
& \qquad\quad \delta'(Z_1, a_2) = Z_2, \ldots, \delta'(Z_{n-1}, a_n) = Z_n \text{ und} \\
& \qquad\quad Z_n \cap E \neq \emptyset \\
& \Longleftrightarrow \quad Z_n = \widehat{\delta}'(z'_0, x) \in F \\
& \Longleftrightarrow \quad x \in L(M').
\end{aligned}
$$

Somit ist $L(M') = L(M)$. ❑

DFA versus NFA, Satz von Rabin und Scott

Beispiel: Wir modifizieren den NFA aus dem obigen Beispiel etwas und betrachten den folgenden um einen Zustand kleineren NFA $M = (\Sigma, Z, \delta, S, E)$:

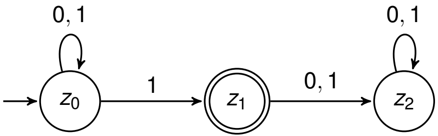


Abbildung: Ein nichtdeterministischer endlicher Automat

Dieser NFA akzeptiert die Sprache aller Wörter über $\{0, 1\}^*$, die an letzter Stelle eine 1 haben.

DFA versus NFA, Satz von Rabin und Scott

Beispiel: Wir konstruieren nun einen zu M äquivalenten DFA $M' = (\Sigma, Z', \delta', S, F)$ gemäß dem Satz von Rabin und Scott:

$$\begin{aligned}
\Sigma &= \{0,1\},\\
Z' &= \{\{z_0,z_1,z_2\},\{z_0,z_1\},\{z_0,z_2\},\{z_1,z_2\},\{z_0\},\{z_1\},\{z_2\},\emptyset\},\\
S &= \{z_0\},\\
F &= \{\{z_0,z_1,z_2\},\{z_0,z_1\},\{z_1,z_2\},\{z_1\}\},
\end{aligned}$$

δ'	$\emptyset$	$\{z_0\}$	$\{z_1\}$	$\{z_2\}$	$\{z_0,z_1\}$	$\{z_0,z_2\}$	$\{z_1,z_2\}$	$\{z_0,z_1,z_2\}$
0	$\emptyset$	$\{z_0\}$	$\{z_2\}$	$\{z_2\}$	$\{z_0,z_2\}$	$\{z_0,z_2\}$	$\{z_2\}$	$\{z_0,z_2\}$
1	$\emptyset$	$\{z_0,z_1\}$	$\{z_2\}$	$\{z_2\}$	$\{z_0,z_1,z_2\}$	$\{z_0,z_1,z_2\}$	$\{z_2\}$	$\{z_0,z_1,z_2\}$

Die Zustände $\emptyset$, $\{z_1\}$, $\{z_2\}$, $\{z_1,z_2\}$ können vom Startzustand $\{z_0\}$ nicht erreicht werden und werden zur Vereinfachung weggelassen.

DFA versus NFA, Satz von Rabin und Scott

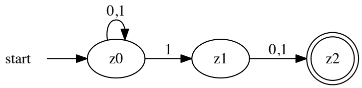
Beispiel: Wir betrachten für $n \geq 1$ die Sprache

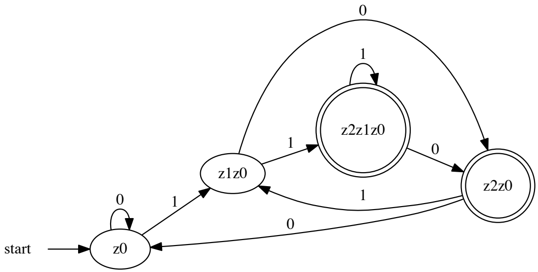
$$\begin{aligned} L_n &= \{x \in \{0,1\}^* \mid \text{der } n\text{-te Buchstabe von hinten in } x \text{ ist } 1\} \\ &= \{x \in \{0,1\}^* \mid x = u1v \text{ mit } u \in \{0,1\}^* \text{ und } v \in \{0,1\}^{n-1}\}. \end{aligned}$$

Es gilt:

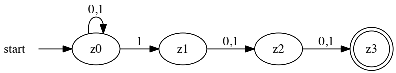
1. Es gibt einen NFA für L_n mit $n+1$ Zuständen.
   (Beweis: Einfache Modifikation der NFAs für L_2 und L_1 in den vorigen Abbildungen.)
2. Jeder DFA für L_n hat mindestens 2^n Zustände.
   (Beweis: siehe Satz von Myhill und Nerode später.)

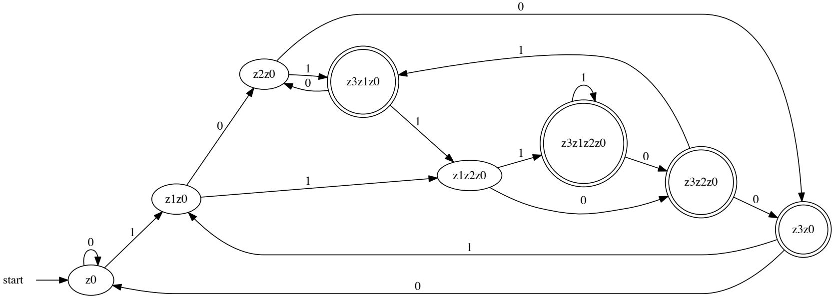
L_2 : DFA versus NFA, Satz von Rabin und Scott



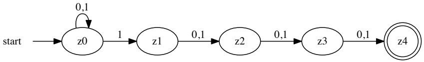


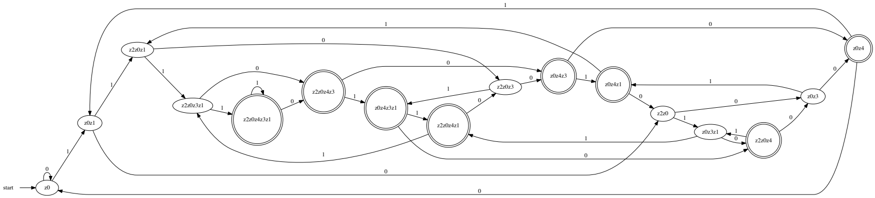
L_3 : DFA versus NFA, Satz von Rabin und Scott



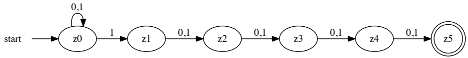


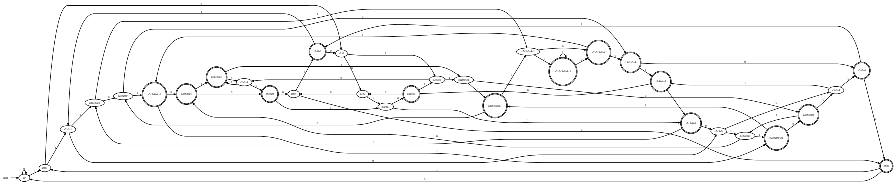
L_4 : DFA versus NFA, Satz von Rabin und Scott



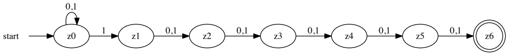


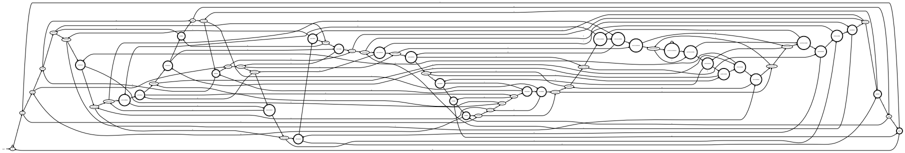
L_5 : DFA versus NFA, Satz von Rabin und Scott



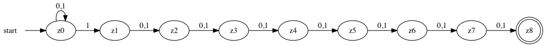


L_6 : DFA versus NFA, Satz von Rabin und Scott





L_8 : DFA versus NFA, Satz von Rabin und Scott



Reguläre Grammatiken und NFAs

Theorem

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

Beweis: Sei $G = (\Sigma, N, S, P)$ eine gegebene reguläre Grammatik.
Konstruiere einen NFA $M = (\Sigma, Z, \delta, S', F)$ so:

- $Z = N \cup \{X\}$, wobei $X \notin N \cup \Sigma$ ein neues Symbol ist,
- $$F = \begin{cases} \{S, X\} & \text{falls } S \rightarrow \lambda \text{ in } P \\ \{X\} & \text{falls } S \rightarrow \lambda \text{ nicht in } P, \end{cases}$$
- $S' = \{S\}$ und
- für alle $A \in N$ und $a \in \Sigma$ sei
$$\delta(A, a) = \left(\bigcup_{A \rightarrow aB \in P} \{B\} \right) \cup \bigcup_{A \rightarrow a \in P} \{X\}.$$

Reguläre Grammatiken und NFAs

Es gilt: $\lambda \in L(G) \iff \lambda \in L(M)$.
Außerdem gilt für alle $n \geq 1$ und $x = a_1 a_2 \cdots a_n$ in Σ^*:

$$
\begin{aligned}
& x \in L(G) \\
\iff \quad & \text{es gibt eine Folge } A_1, A_2, \ldots, A_{n-1} \in N \text{ mit} \\
& S \vdash a_1 A_1 \vdash a_1 a_2 A_2 \vdash \cdots \vdash a_1 a_2 \cdots a_{n-1} A_{n-1} \vdash a_1 a_2 \cdots a_{n-1} a_n \\
\iff \quad & \text{es gibt eine Folge } A_1, A_2, \ldots, A_{n-1} \in Z \text{ mit } A_1 \in \delta(S, a_1), \\
& A_2 \in \delta(A_1, a_2), \ldots, A_{n-1} \in \delta(A_{n-2}, a_{n-1}), X \in \delta(A_{n-1}, a_n) \\
\iff \quad & \widehat{\delta}(S', x) \cap F = \widehat{\delta}(\{S\}, x) \cap F \neq \emptyset \\
\iff \quad & x \in L(M).
\end{aligned}
$$

Somit ist $L(G) = L(M)$. ❑

Reguläre Grammatiken und NFAs

Beispiel: Es sei $G = (\Sigma, N, S, P)$ die folgende reguläre Grammatik:

$$\begin{array}{rcl} \Sigma & = & \{0, 1\}, \\ N & = & \{S\}, \\ P & = & \{S \rightarrow 0S \mid 1S \mid 1\}. \end{array}$$

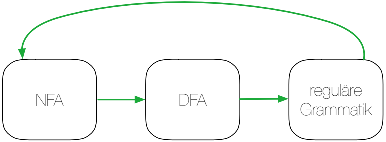
Nach dem obigen Beweis konstruieren wir den NFA
$M = (\Sigma, Z, \delta, S', F)$ mit

$$\begin{array}{rclcrcl} \Sigma & = & \{0, 1\}, & & Z & = & \{S, X\}, \\ F & = & \{X\}, & & S' & = & \{S\}, \\ \delta & : & \delta(S, 0) = \{S\}, & & & & \\ & & \delta(S, 1) = \{S, X\}. & & & & \end{array}$$

Reguläre Grammatiken, DFAs und NFAs

Folgerung: $\mathrm{REG} = \{L(M) \mid M \text{ ist DFA}\} = \{L(M) \mid M \text{ ist NFA}\}$.

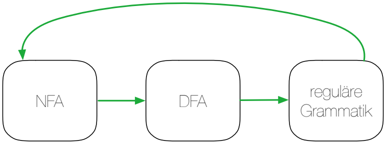
Bemerkung: Zu jeder regulären Sprache gibt es unendlich viele DFAs bzw. NFAs, die sie akzeptieren.



Beispiel:Reguläre Grammatiken, DFAs und NFAs

Wir führen eine Iteration durch diesen Zyklus mit dieser regulären Grammatik als Startpunkt durch (Tafel):

$$
\begin{aligned}
P = \{ \quad & S \rightarrow 0S \mid 1S \mid 1R, \\
& R \rightarrow 0 \mid 1\}.
\end{aligned}
$$



Reguläre Ausdrücke

Woher kennen Sie reguläre Ausdrücke?

- Editoren: Suche, Ersetzen
- Syntax Highlighting
- Grep auf der Kommandozeile: Suche im Dateisystem
- Compilerbau: Lexing Phase
- Syntax und Umfang weichen oft (stark) ab.
- Was ist die Essenz der regulären Ausdrücke?
- Wie stehen diese im Zusammenhang mit DFAs, NFAs, regulären Grammatiken?

Reguläre Ausdrücke

Definition

Sei Σ ein Alphabet. Die *Menge der regulären Ausdrücke* (über Σ) ist definiert durch:

- $\emptyset$ und λ sind reguläre Ausdrücke.
- Jedes $a \in \Sigma$ ist ein regulärer Ausdruck.
- Sind α und β reguläre Ausdrücke, so sind auch
  - $\alpha\beta$,
  - $(\alpha + \beta)$ und
  - $(\alpha)^*$

  reguläre Ausdrücke.
- Nichts sonst ist ein regulärer Ausdruck.

Sprache regulärer Ausdrücke

Definition

Sei Σ ein Alphabet. Jedem regulären Ausdruck γ ist in eindeutiger Weise eine Sprache $L(\gamma) \subseteq \Sigma^*$ zugeordnet (wir sagen: „γ beschreibt $L(\gamma)$“), die so definiert ist:

$$L(\gamma) = \begin{cases} \emptyset & \text{falls } \gamma = \emptyset \\ \{\lambda\} & \text{falls } \gamma = \lambda \\ \{a\} & \text{falls } \gamma = a \text{ für ein } a \in \Sigma \\ L(\alpha)L(\beta) & \text{falls } \gamma = \alpha\beta \\ L(\alpha) \cup L(\beta) & \text{falls } \gamma = (\alpha + \beta) \\ L(\alpha)^* & \text{falls } \gamma = (\alpha)^*. \end{cases}$$

Sprache regulärer Ausdrücke

Definition

Zwei reguläre Ausdrücke α_1 und α_2 heißen *äquivalent* (kurz $\alpha_1 \sim \alpha_2$), falls $L(\alpha_1) = L(\alpha_2)$.

Beispiel: Es sei $\Sigma = \{a, b\}$.

1. Der reguläre Ausdruck $\alpha_1 = (\lambda + a(a + b)^*)$ beschreibt die Sprache

$$L(\alpha_1) = \{\lambda\} \cup \{ax \mid x \in \{a, b\}^*\},$$

d.h. das leere Wort und alle Wörter über Σ, die mit a beginnen.

2. Der reguläre Ausdruck $\alpha_2 = ((a + b)^* ab)$ beschreibt die Sprache

$$L(\alpha_2) = \{xab \mid x \in \{a, b\}^*\},$$

d.h. alle Wörter über Σ, die mit ab enden.

Sprache regulärer Ausdrücke

Bemerkung:

- Wegen $\emptyset^* = \{\lambda\}$ hätte man den regulären Ausdruck λ in der Definition oben auch weglassen können.
- Formal korrekt hätte man die Menge der regulären Ausdrücke unabhängig vom Alphabet (wie Grammatiken und DFAs) definieren müssen, etwa so:

$$
\begin{array}{rcl}
\text{RegAusdruck}(\Sigma) & = & \{\alpha \mid \alpha \text{ ist regulärer Ausdruck über } \Sigma\} \\
\text{RegAusdruck} & = & \bigcup_{\Sigma \text{ ist Alphabet}} \text{RegAusdruck}(\Sigma).
\end{array}
$$

Wir nehmen jedoch an, dass implizit immer ein Alphabet fixiert ist.

Sprache regulärer Ausdrücke

Bemerkung:

- Jede endliche Sprache $A \subseteq \Sigma^*$ wird durch einen regulären Ausdruck beschrieben. Ist etwa $A = \{a_1, a_2, \ldots, a_k\}$, so ist $A = L(\alpha)$ mit

$$\alpha = (\cdots((a_1 + a_2) + a_3) + \cdots + a_k).$$

  Somit ist jede endliche Sprache regulär.
- Jeder reguläre Ausdruck α beschreibt genau eine Sprache, nämlich $L(\alpha)$, aber für jede reguläre Sprache $A = L(\alpha)$ gibt es zum Beispiel wegen

$$L((\alpha + \alpha)) = L(\alpha) \cup L(\alpha) = L(\alpha)$$

  unendlich viele reguläre Ausdrücke, die A beschreiben.

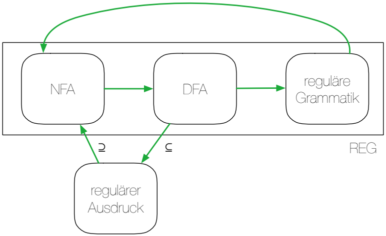
Satz von Kleene

Theorem (Kleene)

$$\mathrm{REG} = \{L(\alpha) \mid \alpha \text{ ist regulärer Ausdruck}\}.$$

Beweisidee:

1. $\mathrm{REG} \supseteq \{L(\alpha) \mid \alpha$ ist regulärer Ausdruck$\}$: Ausdruck in NFA umwandeln
2. $\mathrm{REG} \subseteq \{L(\alpha) \mid \alpha$ ist regulärer Ausdruck$\}$: DFA in regulären Ausdruck umwandeln



Satz von Kleene

Theorem (Kleene)

$\mathrm{REG} = \{L(\alpha) \mid \alpha$ *ist regulärer Ausdruck*$\}$.

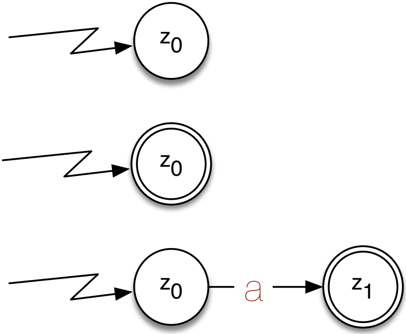
Proof.

($\supseteq$) Sei $A = L(\alpha)$ für einen regulären Ausdruck α. Wir zeigen durch Induktion über den Aufbau von α, dass $A \in \mathrm{REG}$ durch Angabe eines NFAs M mit $L(M) = A$.

- **Induktionsanfang:** $\boldsymbol{\alpha = \emptyset}$ **oder** $\boldsymbol{\alpha = \lambda}$ **oder** $\boldsymbol{\alpha = a \in \Sigma}$**.**
  Der folgende NFA M leistet je nach Fall das Gewünschte:

$$M = \begin{cases} (\Sigma, \{z_0\}, \emptyset, \{z_0\}, \emptyset) & \text{falls } \alpha = \emptyset \\ (\Sigma, \{z_0\}, \emptyset, \{z_0\}, \{z_0\}) & \text{falls } \alpha = \lambda \\ (\Sigma, \{z_0, z_1\}, \{\delta(z_0, a) = \{z_1\}\}, \{z_0\}, \{z_1\}) & \text{falls } \alpha = a \in \Sigma. \end{cases}$$

Satz von Kleene: $\emptyset$, λ, $a \in \Sigma$



Satz von Kleene: Beweis von ($\supseteq$)

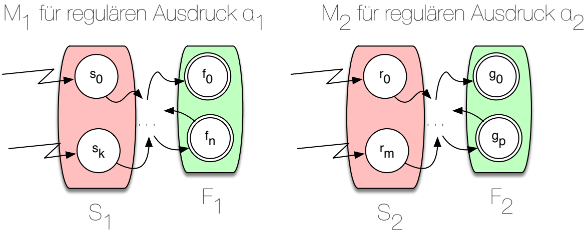
- **Induktionsvoraussetzung:** Seien α_1 und α_2 reguläre Ausdrücke, für die NFAs M_1, M_2 existieren mit $L(M_1) = L(\alpha_1)$, $L(M_2) = L(\alpha_2)$.
- **Induktionsschritt: $\alpha \in \{\alpha_1\alpha_2, (\alpha_1 + \alpha_2), (\alpha_1)^*\}$, für reguläre Ausdrücke α_1 und α_2.**

  Nach Induktionsvoraussetzung existieren NFAs M_1 und M_2 mit $L(M_i) = L(\alpha_i)$, für $i \in \{1, 2\}$. Sei für $i \in \{1, 2\}$:

  $$M_i = (\Sigma, Z_i, \delta_i, S_i, F_i),$$

  wobei o.B.d.A. $Z_1 \cap Z_2 = \emptyset$ gelte; andernfalls werden Zustände entsprechend umbenannt.

Satz von Kleene: M_1 und M_2



Satz von Kleene: Beweis von ($\supseteq$)

- **Fall 1: $\boldsymbol{\alpha = \alpha_1\alpha_2}$.** Definiere $M = (\Sigma, Z, \delta, S, F)$ als die „Hintereinanderschaltung“ von M_1 und M_2 durch:

$$Z = Z_1 \cup Z_2; \quad S = \begin{cases} S_1 & \text{falls } \lambda \notin L(\alpha_1) = L(M_1) \\ S_1 \cup S_2 & \text{falls } \lambda \in L(\alpha_1) = L(M_1); \end{cases}$$

$$F = \begin{cases} F_2 & \text{falls } \lambda \notin L(\alpha_2) = L(M_2) \\ F_1 \cup F_2 & \text{falls } \lambda \in L(\alpha_2) = L(M_2); \end{cases}$$

$$\delta(z,a) = \begin{cases} \delta_1(z,a) & \text{falls } z \in Z_1 \text{ und } \delta_1(z,a) \cap F_1 = \emptyset \\ \delta_1(z,a) \cup S_2 & \text{falls } z \in Z_1 \text{ und } \delta_1(z,a) \cap F_1 \neq \emptyset \\ \delta_2(z,a) & \text{falls } z \in Z_2 \end{cases}$$

für alle $a \in \Sigma$ und $z \in Z$.

Satz von Kleene: Beweis von ($\supseteq$)

- **Fall 2: $\alpha = (\alpha_1 + \alpha_2)$.**
  Der „Vereinigungsautomat“

$$M = (\Sigma, Z_1 \cup Z_2, \delta_1 \cup \delta_2, S_1 \cup S_2, F_1 \cup F_2)$$

von M_1 und M_2 leistet das Gewünschte.

Satz von Kleene: Beweis von ($\supseteq$)

- **Fall 3: $\alpha = (\alpha_1)^*$.**
  Der „Rückkopplungsautomat“

$$M = (\Sigma, Z, \delta, S, F)$$

von M_1 leistet das Gewünschte, der folgendermaßen definiert ist:

  - Zunächst fügen wir λ hinzu, falls nötig:

$$\widehat{M} = \begin{cases} M_1 & \text{falls } \lambda \in L(M_1) \\ (\Sigma, \widehat{Z}, \delta, \widehat{S}, \widehat{F}) & \text{falls } \lambda \notin L(M_1) \end{cases}$$

mit $\widehat{Z} = Z_1 \cup \{\hat{z}\}$, $\widehat{S} = S_1 \cup \{\hat{z}\}$, $\widehat{F} = F_1 \cup \{\hat{z}\}$, wobei $\hat{z} \notin Z_1$.
Offenbar ist

$$L(\widehat{M}) = L(M_1) \cup \{\lambda\}.$$

Satz von Kleene: Beweis von ($\supseteq$)

Weiter zu **Fall 3: $\alpha = (\alpha_1)^*$.**

- Rückkopplung: Definiere

$$M = (\Sigma, Z, \delta, S, F),$$

wobei Z, S und F die Menge der Zustände, Anfangszustände und Endzustände von $\widehat{M}$ sind, und für alle $a \in \Sigma$ und $z \in Z$:

$$\delta(z, a) = \begin{cases} \delta_1(z, a) & \text{falls } \delta_1(z, a) \cap F = \emptyset \\ \delta_1(z, a) \cup S & \text{falls } \delta_1(z, a) \cap F \neq \emptyset. \end{cases}$$

Offenbar gilt

$$L(M) = L(M_1)^* = L((\alpha_1)^*) = L(\alpha).$$

Somit ist $\mathrm{REG} \supseteq \{L(\alpha) \mid \alpha \text{ ist regulärer Ausdruck}\}$ gezeigt.

Satz von Kleene: Beweis von ($\supseteq$)

Beispiel für regulären Ausdruck (a+b)c (Tafel).

Satz von Kleene: Beweis von ($\subseteq$)

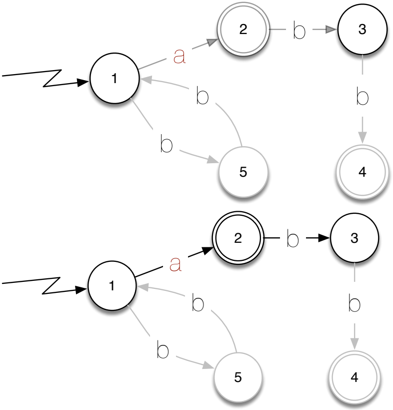
($\subseteq$) Sei $A \in \mathrm{REG}$, und sei M ein DFA mit $L(M) = A$, wobei $M = (\Sigma, Z, \delta, z_1, E)$ mit $Z = \{z_1, z_2, \ldots, z_n\}$.
Konstruiere einen regulären Ausdruck α mit $L(\alpha) = A$.
Definiere dazu Sprachen $R^k_{i,j}$ mit $i, j \in \{1, 2, \ldots, n\}$ und $k \in \{0, 1, \ldots, n\}$, die sich durch reguläre Ausdrücke beschreiben lassen:

$$R^k_{i,j} = \left\{ x \in \Sigma^* \;\middle|\; \begin{array}{l} \widehat{\delta}(z_i, x) = z_j \text{ und keiner der dabei durchlaufenen} \\ \text{Zustände (außer } z_i \text{ und } z_j \text{ selbst) hat einen Index} \\ \text{größer als } k. \end{array} \right\}.$$

$R^k_{i,j}$ = Wörter die man mit i als Anfangs- und j als Endzustand und mit Hilfe der Zustände $1 \ldots k$ generieren kann.

Satz von Kleene: $R^k_{i,j}$

$R^k_{i,j}$ = Wörter die man mit i als Anfangs- und j als Endzustand und mit Hilfe der Zustände $1 \ldots k$ generieren kann.

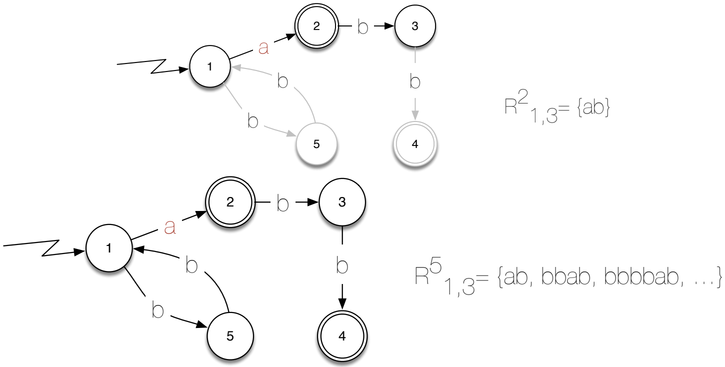


$R^0_{1,3} = \{\}$

$R^2_{1,3} = \{ab\}$

Satz von Kleene: $R^k_{i,j}$

$R^k_{i,j}$ = Wörter die man mit i als Anfangs- und j als Endzustand und mit Hilfe der Zustände $1 \ldots k$ generieren kann.



Satz von Kleene: Beweis von ($\subseteq$)

Wir zeigen durch Induktion über k: Für jedes $k \in \{0, 1, \ldots, n\}$ lässt sich jede Sprache $R^k_{i,j}$ durch einen regulären Ausdruck beschreiben.

- **Induktionsanfang: $k = 0$.** Ist $i \neq j$, so ist

$$R^0_{i,j} = \{a \in \Sigma \mid \delta(z_i, a) = z_j\}.$$

  Ist $i = j$, so ist

$$R^0_{i,i} = \{\lambda\} \cup \{a \in \Sigma \mid \delta(z_i, a) = z_i\}.$$

  In beiden Fällen ist $R^0_{i,j}$ endlich und daher durch einen regulären Ausdruck beschreibbar.

- **Induktionsvoraussetzung:** Sei nun $k \in \mathbb{N}$ und $R^k_{i,j}$ lässt sich durch einen regulären Ausdruck beschreiben.

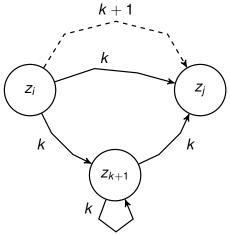
Satz von Kleene: Beweis von ($\subseteq$)

- **Induktionsschritt: $k \mapsto k+1$.** Es gilt

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

denn:

  - Entweder braucht man den Zustand z_{k+1} gar nicht, um von z_i nach z_j zu kommen,
  - oder z_{k+1} wird dabei einmal oder mehrmals in Schleifen durchlaufen:



Satz von Kleene: Beweis von ($\subseteq$)

Nach Induktionsvoraussetzung gibt es reguläre Ausdrücke $\alpha_{i,j}^{k}$ für die Mengen $R_{i,j}^{k}$. Somit ist

$$\alpha_{i,j}^{k+1} = \left(\alpha_{i,j}^{k} + \alpha_{i,k+1}^{k}(\alpha_{k+1,k+1}^{k})^{*}\alpha_{k+1,j}^{k}\right)$$

ein regulärer Ausdruck für $R_{i,j}^{k+1}$. Ende der Induktion.

Ferner gilt:

$$A = L(M) = \bigcup_{z_i \in E} R_{1,i}^{n} \quad \text{mit } n = |Z|.$$

Ist also $E = \{z_{i_1}, z_{i_2}, \ldots, z_{i_m}\}$, so ist

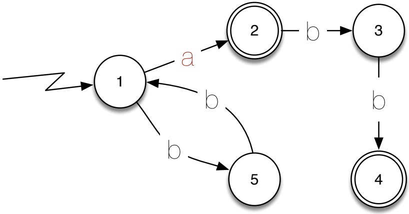
$$\alpha = \left(\alpha_{1,i_1}^{n} + \alpha_{1,i_2}^{n} + \cdots + \alpha_{1,i_m}^{n}\right)$$

ein regulärer Ausdruck für A. ❑

Satz von Kleene: Beispiel für $RA_{i,j}^{k}$

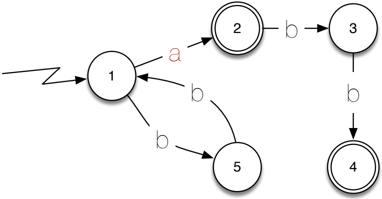
$RA_{i,j}^{k}$ ist ein regulärer Ausdruck für $R_{i,j}^{k}$

- $RA_{i,j}^{0} = \emptyset$ außer $RA_{i,i}^{0} = \lambda$, $RA_{1,2}^{0} = a$, $RA_{1,5}^{0} = RA_{2,3}^{0} = RA_{3,4}^{0} = RA_{5,1}^{0} = b$,
- $RA_{i,j}^{1} \sim RA_{i,j}^{0}$ außer für:
  $RA_{5,2}^{1} = RA_{5,2}^{0} + RA_{5,1}^{0}(RA_{1,1}^{0})^{*}RA_{1,2}^{0}$
  $RA_{5,2}^{1} = \emptyset + b(\lambda)^{*}a \sim ba$
  $RA_{5,5}^{1} = RA_{5,5}^{0} + RA_{5,1}^{0}(RA_{1,1}^{0})^{*}RA_{1,5}^{0}$
  $RA_{5,5}^{1} = \lambda + b(\lambda)^{*}b \sim \lambda + bb$



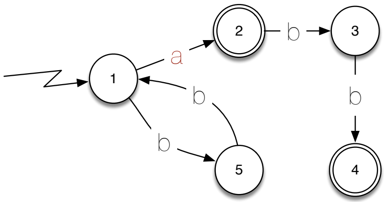
Satz von Kleene: Beispiel für $RA^k_{i,j}$

- $RA^2_{i,j} \sim RA^1_{i,j}$ außer für: $RA^2_{1,3} = \emptyset + a(\lambda)^* b \sim ab$
  $RA^2_{5,3} = \emptyset + ba(\lambda)^* b \sim bab$
- $RA^3_{i,j} \sim RA^2_{i,j}$ außer für: $RA^3_{2,4} = \emptyset + b(\lambda)^* b \sim bb$
  $RA^3_{1,4} = \emptyset + ab(\lambda)^* b \sim abb$
  $RA^3_{5,4} = \emptyset + bab(\lambda)^* b \sim babb$
- $RA^5_{i,j} \sim RA^4_{i,j} \sim RA^3_{i,j}$ außer zB für:
  $RA^5_{1,2} = RA^4_{1,2} + RA^4_{1,5}(RA^4_{5,5})^* RA^4_{5,2} = a + b(\lambda + bb)^* ba \sim (bb)^* a$
  $RA^5_{1,4} = abb + b(\lambda + bb)^* babb \sim (bb)^* abb$



Satz von Kleene: Beispiel für $RA_{i,j}^k$

- $RA_{1,2}^5 = b(\lambda + bb)^* ba \sim (bb)^* a$
  $RA_{1,4}^5 = abb + b(\lambda + bb)^* babb \sim (bb)^* abb$
- regulärer Ausdruck für den DFA:
  $RA_{1,2}^5 + RA_{1,4}^5 \sim (bb)^* a + (bb)^* abb \sim (bb)^* a(\lambda + bb)$



Satz von Kleene: Beispiel für $RA^{k}_{i,j}$

Ohne Vereinfachung (zB $\lambda+\emptyset \sim \lambda$) ist der Ausdruck deutlich komplizierter:

a+∅+((λ+∅)(*(λ)(a+∅))+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(λ+∅+(∅(*(λ)(a+∅))+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+

((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(∅+(∅(*(λ)(a+∅))+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(λ+∅+(∅

(*(λ)(a+∅))+∅)))+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)+((∅+((λ+∅)

(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))

(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅))(*(λ)(∅+(∅(*(λ)(a+∅))+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(λ+∅+(∅(*(λ)(a+∅))+∅)))+∅)+((∅+

(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(∅+(∅(*(λ)(a+∅))+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(λ+∅+(∅(*(λ)(a

∅)(*(λ)(b+∅))+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+

∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)))+∅)))+∅)+((∅+

((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+

∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+

(∅(*(λ)(b+∅))+∅)))+∅)+((∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)+

((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)))+∅)))+∅)))+∅))(*(λ)(∅+((b+∅)(*(λ)(a+∅))+∅)+((∅+((b+∅)(*(λ)(a+∅))+∅))(*(λ)(λ+∅+

Satz von Kleene: Beispiel für $RA_{i,j}^k$

Ohne Vereinfachung ist der Ausdruck deutlich komplizierter (Teil 2):

(∅(*(λ)(a+∅))+∅)))+∅)+((∅+((b+∅)(*(λ)∅)+∅)+((∅+((b+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(∅+(∅(*(λ)(a+∅))+∅)+

((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(λ+∅+(∅(*(λ)(a+∅))+∅)))+∅)))+∅)+((∅+((b+∅)(*(λ)∅)+∅)+((∅+((b+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅

(*(λ)∅)+∅)))+∅)+((∅+((b+∅)(*(λ)∅)+∅)+((∅+((b+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+

(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅))(*(λ)(∅+(∅(*(λ)(a+∅))+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(λ+∅+(∅(*(λ)(a+∅))+

∅)))+∅)+((∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(∅+(∅(*(λ)(a+∅))+∅)+((∅+(∅(*(λ)

(a+∅))+∅))(*(λ)(λ+∅+(∅(*(λ)(a+∅))+∅)))+∅)))+∅)))+∅)))+∅)+(∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅

(*(λ)∅)+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+

((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅

(*(λ)∅)+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+

((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅))(*(λ)(λ+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+

∅)))+∅)+((∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+

∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅)))+∅)+((b+∅+((λ+∅)(*(λ)(b+∅))+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)))+∅)+

Satz von Kleene: Beispiel für $RA_{i,j}^{k}$

Ohne Vereinfachung ist der Ausdruck deutlich komplizierter (Teil 3/3):

∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)))+∅)))+∅)+((∅+((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)+((∅+

((λ+∅)(*(λ)∅)+∅)+((a+∅+((λ+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)

(∅+(∅(*(λ)∅)+∅)))+∅)))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)))+∅)+((∅+(∅(*(λ)∅)+∅)+

((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)(b+∅))+∅)))+

∅)))+∅)))+∅))(*(λ)(∅+((b+∅)(*(λ)∅)+∅)+((∅+((b+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)+((∅+((b+∅)(*(λ)∅)+∅)+((∅+((b+∅)(*(λ)

(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅)+((∅

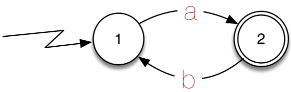
+((b+∅)(*(λ)∅)+∅)+((∅+((b+∅)(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)+((∅+((b+∅)(*(λ)∅)+∅)+((∅+((b+∅)(*(λ)(a+∅))+∅))(*(λ)(b+∅+

(∅(*(λ)∅)+∅)))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅))(*(λ)(λ+∅+(∅(*(λ)∅)+

∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)+((∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(b+∅+(∅(*(λ)∅)+∅)))+∅))

(*(λ)(b+∅+(∅(*(λ)∅)+∅)+((∅+(∅(*(λ)(a+∅))+∅))(*(λ)(∅+(∅(*(λ)∅)+∅)))+∅)))+∅)))+∅)+∅)

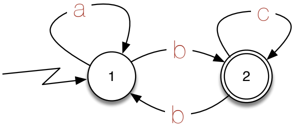
Satz von Kleene: weiteres Beispiel für $RA_{i,j}^k$



- $RA_{1,1}^0 = \lambda$, $RA_{1,2}^0 = a$, $RA_{2,1}^0 = b$, $RA_{2,2}^0 = \lambda$
- $RA_{2,2}^1 = RA_{2,2}^0 + RA_{2,1}^0(RA_{1,1}^0)^* RA_{1,2}^0 = \lambda + b\lambda^* a \sim \lambda + ba$
- $RA_{1,2}^2 = RA_{1,2}^1 + RA_{1,2}^1(RA_{2,2}^1)^* RA_{2,2}^1$
  $\sim a + a(\lambda + ba)^*(\lambda + ba)$
  $\sim a + a(ba)^*(\lambda + ba) \sim a + a(ba)^* \sim a(ba)^*$

	I=1,J=1	I=1, J=2	I=2, J=1	I=2, J=2
K=2		a+a((ba)*)(λ+ba)		
K=1	λ	a	b	λ+ba
K=0	λ	a	b	λ

Beispiel 3: Satz von Kleene: $RA_{i,j}^{k}$



- $RA_{1,2}^{2} = \text{b} + (\lambda\text{+a})(\text{a}^*)\text{b} + (\text{b}+(\lambda\text{+a})(\text{a}^*)\text{b})\ ((\lambda\text{+c+b}(\text{a}^*)\text{b})^*)$
  $(\lambda\text{+c+b}(\text{a}^*)\text{b})$
  $\sim \text{a}^*\text{b}(\text{c+b}(\text{a}^*\text{b}))^*$

	I=1,J=1	I=1, J=2	I=2, J=1	I=2, J=2
K=2		b+(λ+a)(a*)b+(b+(λ+a)(a*)b)((λ+c+b(a*)b)*)(λ+c+b(a*)b)		
K=1	λ+a+(λ+a)(a*)(λ+a)	b+(λ+a)(a*)b	b+b(a*)(λ+a)	λ+c+b(a*)b
K=0	λ+a	b	b	λ+c

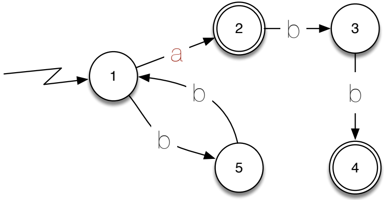
Pumping-Lemma

- Wir können zeigen, dass eine Sprache regulär, also vom Typ 3 ist. (Angabe einer reg. Grammatik, Angabe eines DFA/NFA/Regex)
- Aber wir wollen auch zeigen können, dass eine Sprache nicht regulär ist.
- Eine Grammatik anzugeben, die nicht von Typ 3 ist, reicht nicht aus.

$\Rightarrow$ Wir brauchen ein Verfahren, um das zu zeigen.

Pumping-Lemma

- Eine reguläre Sprache kann von einem DFA oder NFA akzeptiert werden
- Was folgt daraus für “längere” Wörter der Sprache?



- Was kann man hier für Wörter mit mindestens 5 Buchstaben aussagen?
- L = $\{a, abb, bba, bbabb, bbbba, bbbbabb, bbbbbba, bbbbbbabb, \ldots\}$
- bbabb $\in L$, bbbbabb $\in L$, bbbbbbabb $\in L$, . . .

Pumping-Lemma

Ziel: Nachweis der Nichtregularität von Sprachen.

Theorem (Pumping-Lemma für reguläre Sprachen)

Sei $L \in \mathrm{REG}$. Dann existiert eine (von L abhängige) Zahl $n \geq 1$, so dass sich alle Wörter $x \in L$ mit $|x| \geq n$ zerlegen lassen in $x = uvw$, wobei gilt:

1. $|uv| \leq n$,
2. $|v| \geq 1$,
3. $(\forall i \geq 0)\,[uv^i w \in L]$.

Pumping-Lemma: Beweis

Beweis: Sei $L \in \mathrm{REG}$, und sei M ein DFA für L.

- Wähle als n die Zahl der Zustände von M.
- Sei $x \in L$ beliebig mit $|x| \geq n$.
- Beim Abarbeiten von x durchläuft M genau $|x| + 1 \geq n + 1$ Zustände einschließlich des Startzustands.
- Folglich gibt es einen Zustand z, der zweimal durchlaufen wird. Das heißt, M hat bei Eingabe x eine Schleife durchlaufen.

Pumping-Lemma: Beweis

- Wähle nun die Zerlegung $x = uvw$ so, dass nach Lesen von u und von uv derselbe Zustand z erreicht ist und die Bedingungen (1) und (2) erfüllt sind. Es ist klar, dass das geht.

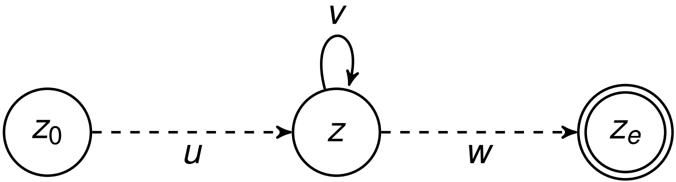


Abbildung: Beweis des Pumping-Lemmas für reguläre Sprachen

Pumping-Lemma: Beweis

- M erreicht somit denselben Endzustand z_e bei Eingabe von

$$\begin{aligned} uv^0w &= uw \\ uv^1w &= uvw = x \\ uv^2w & \\ \vdots & \\ uv^iw & \\ \vdots & \end{aligned}$$

(siehe Abbildung auf der vorherigen Folie).

- Somit sind alle diese Wörter in L, und auch Eigenschaft (3) ist bewiesen. ❑

Pumping-Lemma

Bemerkung: Man beachte, dass das Pumping-Lemma keine Charakterisierung von REG liefert, sondern lediglich eine Implikation:

$$L \in \mathrm{REG} \quad \Rightarrow \quad (\exists n \geq 1)\,(\forall x \in L, |x| \geq n)\,(\exists u, v, w \in \Sigma^*)$$
$$[x = uvw \wedge (1) \wedge (2) \wedge (3)].$$

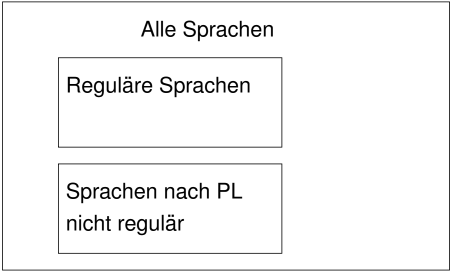


Abbildung: Anwendbarkeit des Pumping-Lemmas (PL)

Pumping-Lemma: Kleine Beispiele

$$L \in \mathrm{REG} \quad \Rightarrow \quad (\exists n \geq 1)\,(\forall x \in L, |x| \geq n)\,(\exists u, v, w \in \Sigma^*)$$
$$[x = uvw \wedge (1) \wedge (2) \wedge (3)].$$

Gegeben sei die Sprache L definiert durch den reg. Ausdruck $ab(b^*)$, wie können wir für folgende Wörter in der Sprache das n bzw. die Zerlegung uvw wählen, die es nach dem PL geben muss:

- ab
- abb
- $abbb$

Pumping-Lemma: Kleine Beispiele

$L(ab(b^*))$

$$L \in \mathrm{REG} \quad \Rightarrow \quad (\exists n \geq 1)\,(\forall x \in L, |x| \geq n)\,(\exists u, v, w \in \Sigma^*)$$
$$[x = uvw \wedge (1) \wedge (2) \wedge (3)].$$

Wir können $n = 3$ wählen und haben:

- ab : $|ab| < n$
- abb = uvw mit $u = ab$ und $v = b$ und $w = \lambda$ und $|uv| \leq 3$. Eine andere valide Zerlegung ist $u = a$, $v = b$ und $w = b$ ($|uv| \leq 3$, $|w| \geq 1$ und $uv^iw \in L$). Nicht valide sind $u = a, v = bb, w = \lambda$ oder $u = \lambda, v = ab, w = b$,...
- $abbbb$ =uvw mit $u = ab$ und $v = b$ und $w = bb$ und $|uv| \leq 3$.

Pumping-Lemma: Kleine Beispiele

$$L \in \mathrm{REG} \quad \Rightarrow \quad (\exists n \geq 1)\,(\forall x \in L, |x| \geq n)\,(\exists u, v, w \in \Sigma^*)$$
$$[x = uvw \wedge (1) \wedge (2) \wedge (3)].$$

Ist das Pumping Lemma für diese Sprache L = $\{a, b, aa, aaa, aaaa\}$ und diese Wörter der Sprache erüllt?

- a
- b
- aaa

Pumping-Lemma: Anwendungen

Behauptung: $L = \{a^m b^m \mid m \geq 1\}$ ist nicht regulär.

Beweis: Der Beweis wird indirekt geführt.

- Angenommen, $L \in \mathrm{REG}$.
- Sei n die Zahl, die nach dem Pumping-Lemma für L existiert.
- Betrachte das Wort

$$x = a^n b^n$$

mit $|x| = 2n > n$.

Pumping-Lemma: Anwendungen

- Da $x \in L$, lässt sich x so in

$$x = uvw = a^n b^n$$

zerlegen, dass gilt:

1. $|uv| \leq n$, d.h., $uv = a^m$ für $m \leq n$;
2. $|v| \geq 1$, d.h., $v = a^k$ mit $k \geq 1$;
3. $(\forall i \geq 0)\,[uv^i w \in L]$.

- Insbesondere gilt für $i = 0$:

$$uv^0 w = uw = a^{n-k} b^n \in L,$$

was ein Widerspruch zur Definition von L ist.

- Also ist die Annahme falsch und L nicht regulär. ❑

Pumping-Lemma: Anwendungen

Achtung: Bei dem Beweis, dass eine Sprache nicht-regulär ist, darf man sich die Zahl n und die Zerlegung $x = uvw$ **nicht** aussuchen.

- Wir führen die Gültigkeit des PL für die Sprache L zum Widerspruch.
- $(\exists n \geq 1)(\forall x \in L, |x| \geq n)(\exists u, v, w : x = uvw \wedge |uv| \leq n \wedge |v| \geq 1)(\forall i \geq 0)\,[uv^i w \in L]$
- Wir wissen aber nicht, was n bzw. uvw sind, und können nicht einfach einen Wert annehmen.
- Alternativ: Wir beweisen die Negation des PL:
  $(\forall n \geq 1)(\exists x \in L, |x| \geq n)(\forall u, v, w : x = uvw \wedge |uv| \leq n \wedge |v| \geq 1)$
  $\Rightarrow (\exists i \geq 0)\,[uv^i w \notin L]$

Pumping-Lemma: Anwendungen

Behauptung: $L = \{0^m \mid m \text{ ist Quadratzahl}\}$ ist nicht regulär.

Beweis: Der Beweis wird wieder indirekt geführt.

- Angenommen, $L \in \mathrm{REG}$.
- Sei n die Zahl, die nach dem Pumping-Lemma für L existiert.
- Betrachte das Wort
$$x = 0^{n^2}$$
mit $|x| = n^2 \geq n$.

Pumping-Lemma: Anwendungen

- Da $x \in L$, lässt sich x so in

$$x = uvw = 0^{n^2}$$

  zerlegen, dass die Bedingungen (1), (2) und (3) aus dem Pumping-Lemma gelten. Aus den Bedingungen (1) und (2) folgt:

$$1 \leq |v| \leq |uv| \leq n.$$

- Aus Bedingung (3) folgt für $i = 2$, dass uv^2w ein Wort in L ist.
- Andererseits gilt:

$$n^2 = |x| = |uvw| < |uv^2w| \leq n^2 + n < n^2 + 2n + 1 = (n+1)^2.$$

- Folglich ist die Länge des Wortes $uv^2w \in L$ keine Quadratzahl, was ein Widerspruch zur Definition von L ist.
- Also ist die Annahme falsch und L nicht regulär. ❑

Pumping-Lemma: für eine nichtreguläre Sprache

Beispiel: Definiere die Sprache

$$L = \{x \in \{0,1\}^* \mid x = 1^k \text{ mit } k \geq 0 \text{ oder } x = 0^j 1^{k^2} \text{ mit } j \geq 1 \text{ und } k \geq 1\}.$$

Ein paar Wörter in L:

- 1
- 11
- 111111
- 01
- 0000001111

Pumping-Lemma: für eine nichtreguläre Sprache

Beispiel: Definiere die Sprache

$$L = \{x \in \{0,1\}^* \mid x = 1^k \text{ mit } k \geq 0 \text{ oder } x = 0^j 1^{k^2} \text{ mit } j \geq 1 \text{ und } k \geq 1\}.$$

Wir versuchen wieder (wie sich hier zeigt: vergeblich!) einen Widerspruchsbeweis für $L \notin \mathrm{REG}$ und nehmen also $L \in \mathrm{REG}$ an.

- Sei $x \in L$ beliebig mit $|x| \geq n$, wobei n die Pumping-Lemma-Zahl für L ist.
- Nach dem Pumping-Lemma gibt es eine Zerlegung $x = uvw$, die die Bedingungen (1), (2) und (3) erfüllt.

Pumping-Lemma für eine nichtreguläre Sprache

Erinnerung: $L = \{x \in \{0,1\}^* \mid (x = 1^k, k \geq 0) \vee (x = 0^j 1^{k^2}, j \geq 1 \text{ und } k \geq 1)\}$

Beispiel:

Fall 1: $\boldsymbol{x = 1^k}$. Für $x = 1^k = uvw$ liefern diese drei Bedingungen (insbesondere $uv^iw = 1^{k'} \in L$ für alle $i \geq 0$) jedoch keinen Widerspruch.

Fall 2: $\boldsymbol{x = 0^j 1^{k^2}}$. Auch für $x = 0^j 1^{k^2} = uvw$ ergibt sich kein Widerspruch aus den drei Bedingungen; insbesondere könnte die Zerlegung $x = uvw$ mit $u = \lambda$ und $v = 0$ (da $j \geq 1$) gewählt worden sein, so dass $uv^iw = 0^{j+i-1}1^{k^2} \in L$ für alle $i \geq 0$ tatsächlich gilt.
(Bei $j + i - 1$=0 ist es wichtig $x = 1^k$ in der Sprache zu haben. Für $x = 0^j 1^{k^2}$ alleine gelingt ein Widerspruchsbeweis.)

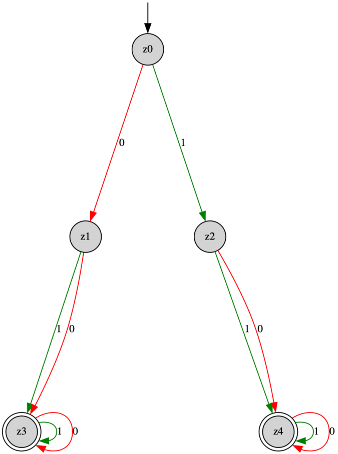
Pumping-Lemma: für eine nichtreguläre Sprache

Beispiel:

Auch wenn das Pumping-Lemma hier nicht den gewünschten Widerspruch liefert, kann man mit einer Verallgemeinerung des Pumping-Lemmas tatsächlich zeigen, dass L nicht regulär ist.

Satz von Myhill und Nerode

Warum ist dieser Automat nicht minimal?



Definition (Minimalautomat)

Ein DFA M mit totaler Überführungsfunktion heißt *Minimalautomat*, falls es keinen zu M äquivalenten DFA mit totaler Überführungsfunktion und weniger Zuständen gibt.

Satz von Myhill und Nerode

Was sind minimale Automaten für folgende Sprachen

- $L_1 = \{ac, bc\}$
- $L_2 = \{aca, bbca\}$
- $L_3 = \{abc, abcbc, abcbcbc, \ldots\}$

Satz von Myhill und Nerode

Man kann einer Sprache $L \subseteq \Sigma^*$ wie folgt eine Äquivalenzrelation R_L auf Σ^* zuordnen.

Definition (Myhill-Nerode-Relation)

Sei $L \subseteq \Sigma^*$ gegeben. Für $x, y \in \Sigma^*$ gelte xR_Ly genau dann, wenn

$$(\forall z \in \Sigma^*)\,[xz \in L \iff yz \in L].$$

Insbesondere gilt für Wörter $x, y \in \Sigma^*$ mit xR_Ly:

$$x \in L \iff y \in L,$$

nämlich für $z = \lambda$.

Satz von Myhill und Nerode

Für welche $x, y \in \Sigma^*$ gilt $(\forall z \in \Sigma^*)\,(xz \in L_i \iff yz \in L_i)$ in folgenden Sprachen:

- $L_1 = \{ac, bc\}$
- $L_2 = \{aca, bbca\}$
- $L_3 = \{abc, abcbc, abcbcbc, \ldots\}$
- $a\ R_{L_1}\ b$, $ac\ R_{L_1}\ bc$, $c\ R_{L_1}\ bb\ R_{L_1}\ aca\ R_{L_1}\ \ldots$
- $a\ R_{L_2}\ bb$, $ac\ R_{L_2}\ bbc$, $aca\ R_{L_2}\ bbca$
- $abc\ R_{L_3}\ abcbc\ R_{L_3}\ abcbcbc \ldots$, $ab\ R_{L_3}\ abcb\ R_{L_3}\ abcbcb \ldots$, aber nicht $a\ R_{L_3}\ abc$ und nicht $\lambda\ R_{L_3}\ abc$

Satz von Myhill und Nerode

Die binäre Relation R_L auf Σ^* mit

$$(\forall z \in \Sigma^*)\, [xz \in L \iff yz \in L].$$

ist in der Tat eine Äquivalenzrelation auf Σ^*:

- Reflexivität: xR_Lx
- Symmetrie: $xR_Ly \Rightarrow yR_Lx$
- Transitivität: $xR_Ly \wedge yR_Lz \Rightarrow xR_Lz$

Satz von Myhill und Nerode

Definition

Wie jede Äquivalenzrelation induziert R_L eine Zerlegung (d.h. eine disjunkte und vollständige Überdeckung) von Σ^* in Äquivalenzklassen:

$$[x] = \{y \in \Sigma^* \mid xR_Ly\},$$

die repräsentantenunabhängig ist.

Die Anzahl der verschiedenen Äquivalenzklassen wird bezeichnet mit

$$\mathrm{Index}(R_L) = \|\{[x] \mid x \in \Sigma^*\}\|.$$

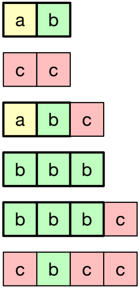
Satz von Myhill und Nerode

Für welche $x, y \in \Sigma^*$ gilt xR_Ly, d.h. $(\forall z \in \Sigma^*)\,(xz \in L \iff yz \in L)$
gleich bedeutend mit: $\{z \mid xz \in L\} = \{z \mid yz \in L\}$

a	b		
c	c		
a	b	c	
b	b	b	
b	b	b	c
c	b	c	c

Satz von Myhill und Nerode

Für welche $x, y \in \Sigma^*$ gilt xR_Ly, d.h. $(\forall z \in \Sigma^*)\,(xz \in L \iff yz \in L)$
gleich bedeutend mit: $\{z \mid xz \in L\} = \{z \mid yz \in L\}$

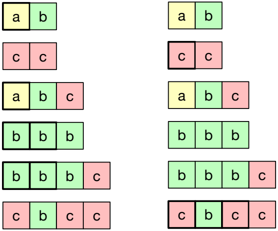


abR_Lbbb da $\{z \mid abz \in L\} = \{\lambda, c\}$ = $\{z \mid bbbz \in L\}$,
Was ist mit aR_Lbb? cR_Lcbc?

Satz von Myhill und Nerode

Für welche $x, y \in \Sigma^*$ gilt xR_Ly, d.h. $(\forall z \in \Sigma^*)\,(xz \in L \iff yz \in L)$

gleich bedeutend mit: $\{z \mid xz \in L\} = \{z \mid yz \in L\}$



abR_Lbbb da $\{z \mid abz \in L\} = \{\lambda, c\}$ = $\{z \mid bbbz \in L\}$,

aR_Lbb da $\{z \mid az \in L\} = \{b, bc\}$ = $\{z \mid bbz \in L\}$,

$\Rightarrow$ nach a und bb muss der Minimalautomat im selben Zustand sein

nicht: cR_Lcbc da $\{z \mid cz \in L\} = \{c, bcc\}$, $\{z \mid cbcz \in L\} = \{c\}$

Satz von Myhill und Nerode

Was sind die Äquivalenzklassen? Wie sieht der Minimalautomat aus?

a b

c c

a b c

b b b

b b b c

c b c c

Satz von Myhill und Nerode

Was sind die Äquivalenzklassen? Wie sieht der Minimalautomat aus?

a b

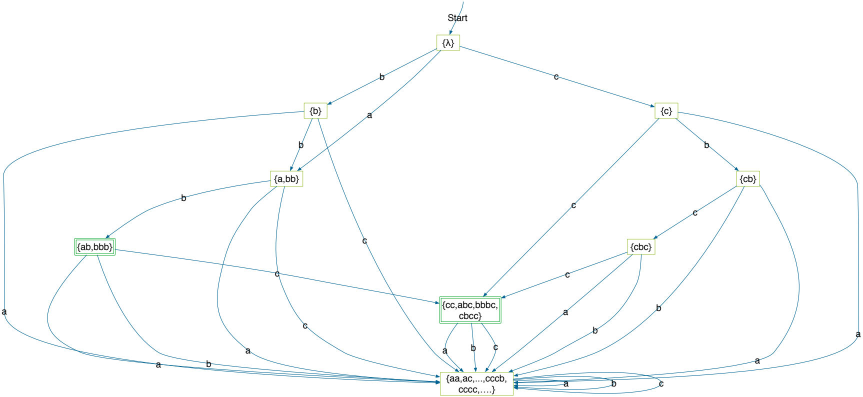
c c

a b c

b b b

b b b c

c b c c



Satz von Myhill und Nerode

Theorem (Myhill und Nerode)

Es sei L eine Sprache über einem beliebigen Alphabet Σ.

$$L \in \mathrm{REG} \iff \mathrm{Index}(R_L) < \infty.$$

Satz von Myhill und Nerode: Beweis

Beweis: **($\Rightarrow$)** Sei $L \in \mathrm{REG}$, und sei $M = (\Sigma, Z, \delta, z_0, E)$ ein totaler DFA mit $L(M) = L$.

- Definiere eine Äquivalenzrelation R_M auf Σ^* wie folgt:
  xR_My gelte genau dann, wenn
  $$\widehat{\delta}(z_0, x) = \widehat{\delta}(z_0, y),$$
  d.h., M ist nach dem Lesen von x im selben Zustand wie nach dem Lesen von y.

- Es gilt:

$$\mathrm{Index}(R_M) \quad = \quad \text{Anzahl der von } z_0 \text{ aus erreichbaren Zustände.} \qquad (1)$$

Satz von Myhill und Nerode: Beweis

- Wir zeigen nun:

$$(\forall x, y \in \Sigma^*)\,[xR_My \Rightarrow xR_Ly], \tag{2}$$

d.h., $R_M \subseteq R_L$ (also ist R_M eine Verfeinerung von R_L mit möglicherweise mehr Äquivalenzklassen).

Sei xR_My, d.h., $\widehat{\delta}(z_0, x) = \widehat{\delta}(z_0, y)$. Sei $z \in \Sigma^*$ beliebig. Dann gilt:

$$\begin{aligned} xz \in L &\iff \widehat{\delta}(z_0, xz) \in E && \iff \widehat{\delta}(\widehat{\delta}(z_0, x), z) \in E \\ &\iff \widehat{\delta}(\widehat{\delta}(z_0, y), z) \in E && \iff \widehat{\delta}(z_0, yz) \in E \\ &\iff yz \in L. \end{aligned}$$

Folglich gilt xR_Ly und somit (2).

- Aus (1) und (2) folgt:

$$\mathrm{Index}(R_L) \leq \mathrm{Index}(R_M) \leq \|Z\| < \infty.$$

Satz von Myhill und Nerode: Beweis

($\Leftarrow$) Sei $\mathrm{Index}(R_L) = k < \infty$ für ein $k \in \mathbb{N}$. Dann lässt sich Σ^* in k Äquivalenzklassen bezüglich R_L zerlegen:

$$[x_1] \cup [x_2] \cup \cdots \cup [x_k] = \Sigma^*,$$

wobei $x_1, x_2, \ldots, x_k \in \Sigma^*$.

Der *Äquivalenzklassen-Automat* für L ist der DFA $M = (\Sigma, Z, \delta, z_0, F)$, der definiert ist durch:

$$\begin{array}{rcl} Z & = & \{[x_1], [x_2], \ldots, [x_k]\}, \\ \delta([x], a) & = & [xa] \ \text{ für jedes } [x] \in Z \text{ und } a \in \Sigma, \\ z_0 & = & [\lambda], \\ F & = & \{[x] \mid x \in L\}. \end{array}$$

Satz von Myhill und Nerode: Beweis

Lemma

Für alle $x' \in [x]$ und $a \in \Sigma$ gilt $[x'a] = [xa]$. Insbesondere gilt $\widehat{\delta}([\lambda], x) = [x]$ für jedes $x \in \Sigma^*$.

Beispiel: $\widehat{\delta}([\lambda], abc) = \widehat{\delta}([x_1], bc) = \widehat{\delta}([a], bc) = \widehat{\delta}([x_2], c) = \widehat{\delta}([ab], c) = \widehat{\delta}([x_3], \lambda) = \widehat{\delta}([abc], \lambda) = [abc]$. Es folgt:

Es folgt:

$$
\begin{aligned}
x \in L(M) &\iff \widehat{\delta}(z_0, x) \in F \\
&\iff \widehat{\delta}([\lambda], x) \in F \ \ \text{(nach Definition von } z_0\text{)} \\
&\iff [x] \in F \ \ \text{(nach Lemma)} \\
&\iff x \in L. \ \ (\text{da} \ \ F = \{[x] \mid x \in L\})
\end{aligned}
$$

Also ist $L \in \mathrm{REG}$. ❑

Satz von Myhill und Nerode: Anwendungen

Beispiel: [Satz von Myhill und Nerode: Nachweis der Nichtregularität]

- Wir zeigen mit dem Satz von Myhill und Nerode, dass

$$L = \{a^m b^m \mid m \geq 1\}$$

  nicht regulär ist.

- Zu den Äquivalenzklassen von L bzgl. R_L gehören:

$$\begin{array}{rcl}
[ab] & = & \{x \in \Sigma^* \mid ab R_L x\} = \{ab, a^2b^2, a^3b^3, \ldots\} = L \\
[a^2b] & = & \{a^2b, a^3b^2, a^4b^3, \ldots\} \\
[a^3b] & = & \{a^3b, a^4b^2, a^5b^3, \ldots\} \\
 & \vdots & \\
[a^kb] & = & \{a^kb, a^{k+1}b^2, a^{k+2}b^3, \ldots\} \\
 & \vdots &
\end{array}$$

Satz von Myhill und Nerode: Anwendungen

Beispiel: [Satz von Myhill und Nerode: Nachweis der Nichtregularität]

- Diese Äquivalenzklassen sind alle paarweise verschieden, d.h.,

$$[a^i b] \neq [a^j b] \quad \text{für } i \neq j.$$

  Denn für $z = b^{i-1}$ gilt

$$a^i bz \in L, \quad \text{aber} \quad a^j bz \notin L.$$

- Folglich ist $\mathrm{Index}(R_L) = \infty$.
- Nach dem Satz von Myhill und Nerode ist $L \notin \mathrm{REG}$. ❑

Satz von Myhill und Nerode: Anwendungen

Beispiel: [Satz von Myhill und Nerode: Nachweis der Regularität]

- Wir zeigen, dass die folgende Sprache L regulär ist:

$$L = \{x \in \{a,b\}^* \mid x \text{ endet mit } aa\}.$$

- Die Äquivalenzklassen von L bzgl. R_L sind:

$$\begin{array}{rcl} [aa] & = & \{x \in \{a,b\}^* \mid aaR_Lx\} \\ & = & \{a^2, a^3, a^4, \ldots, ba^2, ba^3, ba^4, \ldots\} = L \\ [a] & = & \{a, ba, b^2a, \ldots\} \\ & = & \{x \in \{a,b\}^* \mid x \text{ endet mit } a, \text{ aber nicht mit } aa\} \\ [\lambda] & = & \{\lambda, b, ab, \ldots\} \ = \ \{x \in \{a,b\}^* \mid x \text{ endet nicht mit } a\}. \end{array}$$

- Da $\{a,b\}^* = [aa] \cup [a] \cup [\lambda]$, ist $\mathrm{Index}(R_L) = 3$.
- Nach dem Satz von Myhill und Nerode ist $L \in \mathrm{REG}$. ❑

Satz von Myhill und Nerode: Anwendungen

Beispiel: [Satz von Myhill und Nerode: Untere Schranken für die Anzahl der Zustände eines DFAs]

- Es sei L eine reguläre Sprache. Für jeden totalen DFA

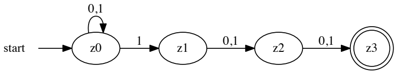
$$M = (\Sigma, Z, \delta, z_0, E)$$

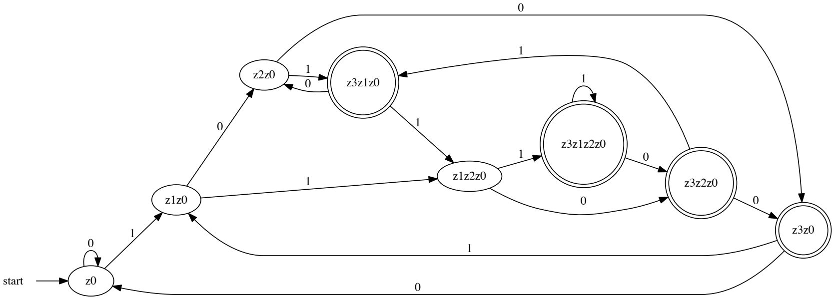
mit $L(M) = L$ gilt nach der „($\Rightarrow$)"-Richtung im Beweis des Satzes von Myhill und Nerode:

$$\begin{aligned} \mathrm{Index}(R_L) &\leq \mathrm{Index}(R_M) \\ &= \text{Anzahl der in } M \text{ von } z_0 \text{ aus erreichbaren Zustände} \\ &\leq \|Z\|. \end{aligned}$$

Das heißt, M hat mindestens so viele Zustände wie R_L Äquivalenzklassen.

Erinnerung: L_3 : dritte Buchstabe von hinten ist 1: DFA versus NFA





Satz von Myhill und Nerode: Anwendungen

Beispiel: [Satz von Myhill und Nerode: Untere Schranken für die Anzahl der Zustände eines DFAs]

- Wir wenden dieses Ergebnis an, um eine untere Schranke für die Anzahl der Zustände der DFAs anzugeben, die die Sprache

$$\begin{aligned} L_n &= \{x \in \{0,1\}^* \mid \text{der } n\text{-te Buchstabe von hinten in } x \text{ ist } 1\} \\ &= \{x \in \{0,1\}^* \mid x = u1v \text{ mit } u \in \{0,1\}^* \text{ und } v \in \{0,1\}^{n-1}\}. \end{aligned}$$

  erkennen.
- Wir betrachten die Wörter aus $\{0,1\}^n$, d.h. die Wörter der Länge n über dem Alphabet $\{0,1\}$.
- Es seien $u, v \in \{0,1\}^n$ mit $u \neq v$. Dann gibt es ein i, $1 \leq i \leq n$, so dass sich u und v an der Position i unterscheiden.

Satz von Myhill und Nerode: Anwendungen

Beispiel: [Satz von Myhill und Nerode: Untere Schranken für die Anzahl der Zustände eines DFAs]

- Wir nehmen o.B.d.A. an, dass u an der Position i eine 0 und v an der Position i eine 1 hat:

$$\begin{array}{ccccccccc|ccc}
u & = & & & & 0 & & & & 0^{i-1} & \notin & L \\
v & = & & & & 1 & & & & 0^{i-1} & \in & L \\
 & & 1 & 2 & \cdots & i & & \cdots & n & & &
\end{array}$$

- Es gilt nun:
  - $u0^{i-1} \notin L$, da der n-te Buchstabe von hinten in $u0^{i-1}$ eine 0 ist, und
  - $v0^{i-1} \in L$, da der n-te Buchstabe von hinten in $v0^{i-1}$ eine 1 ist.

Satz von Myhill und Nerode: Anwendungen

Beispiel: [Satz von Myhill und Nerode: Untere Schranken für die Anzahl der Zustände eines DFAs]

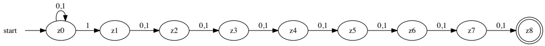
- Somit sind u und v nicht äquivalent bezüglich R_{L_n}.
- Da u und v beliebig waren, sind alle Wörter aus $\{0, 1\}^n$ nicht äquivalent bezüglich R_{L_n}, und es folgt:

$$\mathrm{Index}(R_{L_n}) \geq 2^n.$$

- Somit hat jeder DFA für L_n mindestens 2^n Zustände. ❑

(Erinnerung: Es gibt einen NFA für L_n mit $n + 1$ Zuständen.)

L_8 : DFA versus NFA



Satz von Myhill und Nerode: Minimalautomaten

Definition (Minimalautomat)

Ein DFA M mit totaler Überführungsfunktion heißt *Minimalautomat*, falls es keinen zu M äquivalenten DFA mit totaler Überführungsfunktion und weniger Zuständen gibt.

Bemerkung: Es sei $L \subseteq \Sigma^*$ eine reguläre Sprache.

- Der in der Rückrichtung im Beweis des Satzes von Myhill und Nerode bestimmte Äquivalenzklassen-Automat M_0 ist der DFA mit einer kleinstmöglichen Anzahl von Zuständen, der L akzeptiert.

Satz von Myhill und Nerode: Minimalautomaten

- Begründung:
  - die Anzahl der Zustände von M_0 ist gleich $\mathrm{Index}(R_L)$ (Anzahl der Äquivalenzklassen der Myhill-Nerode-Relation) und
  - für jeden totalen DFA $M = (\Sigma, Z, \delta, z_0, E)$ mit $L(M) = L = L(M_0)$ gilt nach der „($\Rightarrow$)“-Richtung im Beweis des Satzes von Myhill und Nerode:

$$\begin{aligned} \mathrm{Index}(R_L) &\leq \mathrm{Index}(R_M) \\ &= \text{Anzahl der in } M \text{ von } z_0 \text{ aus erreichbaren Zustände} \\ &\leq \|Z\|. \end{aligned}$$

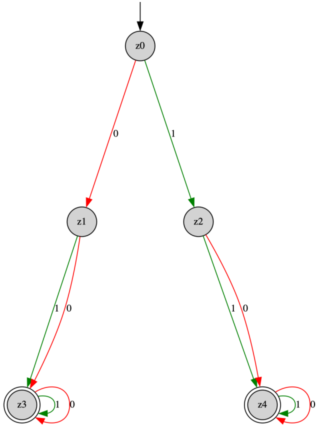
Folglich hat M_0 höchstens so viele Zustände wie M. ❑

Satz von Myhill und Nerode: Minimalautomaten

Bemerkung: Alle DFAs, die L akzeptieren und genau $\mathrm{Index}(R_L)$ Zustände haben, sind bis auf Umbenennung der Zustände äquivalent, d.h., Minimalautomaten sind (bis auf Isomorphie) eindeutig bestimmt.

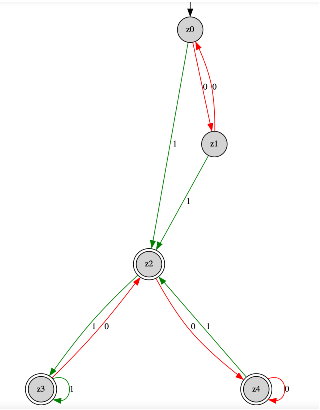
Algorithmus Minimalautomat: Nachdenken 1

Mit welchem Algorithmus kann man den Minimalautomaten berechnen?



Algorithmus Minimalautomat: Nachdenken 2

Mit welchem Algorithmus kann man den Minimalautomaten berechnen?



Algorithmus Minimalautomat

Eingabe: totaler DFA $M = (\Sigma, Z, \delta, z_0, F)$.

Ausgabe: Ein zu M äquivalenter Minimalautomat.

Algorithmus:

1. Entferne alle von z_0 nicht erreichbaren Zustände aus Z.
2. Erstelle eine Tabelle aller (ungeordneten) Zustandspaare $\{z, z'\}$ von M mit $z \neq z'$.
3. Markiere alle Paare $\{z, z'\}$ mit
$$z \in F \iff z' \notin F.$$
4. Sei $\{z, z'\}$ ein unmarkiertes Paar. Prüfe für jedes $a \in \Sigma$, ob $\{\delta(z, a), \delta(z', a)\}$ bereits markiert ist. Ist mindestens ein Test erfolgreich, so markiere auch $\{z, z'\}$.
5. Wiederhole Schritt 4, bis keine Änderung mehr eintritt.
6. Bilde maximale Mengen paarweise nicht disjunkter unmarkierter Zustandspaare und verschmelze jeweils alle Zustände einer Menge zu einem neuen Zustand.

Algorithmus Minimalautomat: Beispiel

Bemerkung: Der Algorithmus Minimalautomat kann mit einer Laufzeit von $O(\|Z\|^2)$ implementiert werden.

Beispiel: Minimalautomat zu gegebenem DFA.
Wir wollen einen Minimalautomaten zum DFA $M = (\Sigma, Z, \delta, z_0, F)$ bestimmen, wobei

$$\begin{aligned} \Sigma &= \{0, 1\} \\ Z &= \{z_0, z_1, z_2, z_3, z_4\} \\ F &= \{z_3, z_4\} \end{aligned}$$

Überführungsfunktion δ					
Σ \ Z	z_0	z_1	z_2	z_3	z_4
0	z_1	z_3	z_4	z_3	z_4
1	z_2	z_3	z_4	z_3	z_4

Algorithmus Minimalautomat: Beispiel

(1.) Es sind alle Zustände von z_0 aus erreichbar.

(2.)+(3.) Markiere alle Paare $\{z, z'\}$ mit $z \in F \iff z' \notin F$:

	z_0	z_1	z_2	z_3
z_4	$\times$	$\times$	$\times$	
z_3	$\times$	$\times$	$\times$	
z_2				
z_1				

Algorithmus Minimalautomat: Beispiel

(4.)

- Da $\{\delta(z_0, 0), \delta(z_2, 0)\}$ markiert ist, wird $\{z_0, z_2\}$ markiert, und
- da $\{\delta(z_0, 0), \delta(z_1, 0)\}$ markiert ist, wird $\{z_0, z_1\}$ markiert:

	z_0	z_1	z_2	z_3
z_4	$\times$	$\times$	$\times$	
z_3	$\times$	$\times$	$\times$	
z_2	$\times$			
z_1	$\times$			

Algorithmus Minimalautomat: Beispiel

(5.) Es ergeben sich nun keine weiteren Änderungen mehr.

(6.) Wir können die Zustände z_1 und z_2 bzw. z_3 und z_4 zu einem Zustand z_{12} bzw. z_{34} zusammenfassen.

Damit erhalten wir einen zu M äquivalenten DFA $M' = (\Sigma, Z', \delta', z_0, F')$:

$$\begin{aligned} \Sigma &= \{0, 1\} \\ Z' &= \{z_0, z_{12}, z_{34}\} \\ F' &= \{z_{34}\} \end{aligned}$$

Überführungsfunktion δ'

Σ \ Z'	z_0	z_{12}	z_{34}
0	z_{12}	z_{34}	z_{34}
1	z_{12}	z_{34}	z_{34}

Abschlusseigenschaften regulärer Sprachen

Definition

Seien Σ ein Alphabet und $\mathcal{C} \subseteq \mathfrak{P}(\Sigma^*)$ eine Sprachklasse über Σ.
$\mathcal{C}$ heißt *abgeschlossen unter*

- *Vereinigung*, falls $(\forall A, B \subseteq \Sigma^*)\, [(A \in \mathcal{C} \;\wedge\; B \in \mathcal{C}) \Rightarrow A \cup B \in \mathcal{C}]$;
- *Komplement*, falls $(\forall A \subseteq \Sigma^*)\, [A \in \mathcal{C} \Rightarrow \overline{A} \in \mathcal{C}]$;
- *Schnitt*, falls $(\forall A, B \subseteq \Sigma^*)\, [(A \in \mathcal{C} \;\wedge\; B \in \mathcal{C}) \Rightarrow A \cap B \in \mathcal{C}]$;
- *Differenz*, falls $(\forall A, B \subseteq \Sigma^*)\, [(A \in \mathcal{C} \;\wedge\; B \in \mathcal{C}) \Rightarrow A - B \in \mathcal{C}]$;
- *Konkatenation*, falls $(\forall A, B \subseteq \Sigma^*)\, [(A \in \mathcal{C} \;\wedge\; B \in \mathcal{C}) \Rightarrow AB \in \mathcal{C}]$;
- *Iteration* (*Kleene-Hülle*), falls $(\forall A \subseteq \Sigma^*)\, [A \in \mathcal{C} \Rightarrow A^* \in \mathcal{C}]$;
- *Spiegelung*, falls $(\forall A \subseteq \Sigma^*)\, [A \in \mathcal{C} \Rightarrow sp(A) \in \mathcal{C}]$.

Abschlusseigenschaften regulärer Sprachen

Theorem

REG *ist unter allen in der obigen Definition angegebenen Operationen abgeschlossen.*

Behauptung:

1. Für jedes $n \geq 1$ ist die Sprache L'_n regulär:

$$
\begin{aligned}
L'_n &= \{x \in \{0,1\}^* \mid \text{der } n\text{-te Buchstabe in } x \text{ ist } 1\} \\
&= \{x \in \{0,1\}^* \mid x = u1v \text{ mit } u \in \{0,1\}^{n-1} \text{ und } v \in \{0,1\}^*\}.
\end{aligned}
$$

- $L'_n = \{0,1\}^{n-1}\{1\}\{0,1\}^*$ ((n-1)-fache Konkatenation von $\{0,1\}$, $\{0,1\}^*$ ist regulär,...)

Abschlusseigenschaften regulärer Sprachen

Behauptung:

2. Die folgende Sprache L ist nicht regulär:

$$\{x \in \{0,1\}^* \mid x \text{ enthält gleich viele 0en und 1en}\}.$$

- $L(0^*1^*)$ ist regulär
- Wäre L regulär, so nach Abschluss bzgl. Schnitt auch $L \cap L(0^*1^*) = \{0^n1^n \mid n \geq 0\}$, dies ist ein Widerspruch.

Charakterisierungen regulärer Sprachen

Folgerung: Es sei $L \subseteq \Sigma^*$ eine Sprache. Dann sind die folgenden Aussagen äquivalent:

1. Es gibt eine rechtslineare Grammatik G mit $L(G) = L$.
   (Form der Regeln: $A \to aB$ oder $A \to a$.)
2. Es gibt eine linkslineare Grammatik G mit $L(G) = L$.
   (Form der Regeln: $A \to Ba$ oder $A \to a$.)
3. Es gibt einen DFA M mit $L(M) = L$.
4. Es gibt einen NFA M mit $L(M) = L$.
5. Es gibt einen regulären Ausdruck α mit $L(\alpha) = L$.
6. Für die Myhill-Nerode-Relation R_L gilt: $\mathrm{Index}(R_L) < \infty$.