Strukturelle Induktion über den Aufbau regulärer Ausdrücke

Σ ein Alphabet. Betrachte reguläre Ausdrücke über Σ.
Eine *Eigenschaft regulärer Ausdrücke* $\mathcal{P}$ ist eine Menge regulärer Ausdrücke.

Theorem

Falls

- $\emptyset, \epsilon \in \mathcal{P}$,
- $a \in \mathcal{P}$ *für alle* $a \in \Sigma$,
- $r + s,\ rs,\ r^* \in \mathcal{P}$ *für* $r, s \in \mathcal{P}$

dann enthält $\mathcal{P}$ *alle regulären Ausdrücke über* Σ.

Einige algebraische Gesetze

Theorem

Es seien $A, B, C \subseteq \Sigma^*$.

1. $A(BC) = (AB)C$
2. $\epsilon A = A\epsilon = A$
3. $(A^*)^* = A^*$
4. $A(B \cup C) = AB \cup AC$
5. $(A \cup B)C = AC \cup BC$
6. $A^+ \cup \{\epsilon\} = A^*$

Reguläre Ausdrücke in UNIX

- $|$ statt $+$
- $a*$ statt a^*
- $a+$ statt a^+
- . ein Zeichen
- ^ Anfang einer Zeile
- $ Ende einer Zeile
- Viele Erweiterungen!

Beispiele:

```
(0|(11)*|(10(1|(00)*)*01))*(0|11|(10(1|00)*01))*
```

```
sed 's/\(.*\),\(.*\)/\2,\1/'
```

Reguläre Sprachen

Definition

Eine Sprache $L \subseteq \Sigma^*$ ist *regulär*, falls es einen regulären Ausdruck r über Σ gibt mit $L = L(r)$.

Die Klasse der regulären Sprachen besteht aus allen regulären Sprachen über allen Alphabeten.

Fragen:
Wieviele reguläre Sprachen gibt es über einem festen Alphabet?
Wieviele Sprachen gibt es insgesamt über einem festen Alphabet?
Antwort:
Es gibt nur abzählbar viele reguläre Sprachen, da es nur abzählbar viele reguläre Ausdrücke gibt.
Es gibt aber überabzählbar viele Sprachen bei festem Alphabet.
Die „meisten“ Sprachen sind also nicht regulär.

Abschlußeigenschafter regulärer Sprachen

Theorem

Falls A und B reguläre Sprachen sind, dann auch

1. $A \cup B$
2. AB
3. A^*
4. A^+
5. $h(A)$ *falls* $A \subseteq \Sigma^*$ *und* $h \colon \Sigma^* \to \Gamma^*$ *ein Homomorphismus*

Was ist mit $A \cap B$?

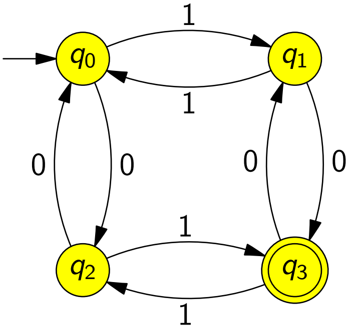
Beweis.

r_A und r_B seien reguläre Ausdrücke für A und B.

1. $r_A r_B$ ist r. A. für AB
2. $r_A + r_B$ ist r. A. für $A \cup B$
3. r_A^* ist r. A. für A^*
4. $r_A r_A^*$ ist r. A. für A^+
5. Strukturelle Induktion:
    - $r_A = \emptyset \rightarrow f(r_A) = \emptyset$
    - $r_A = \epsilon \rightarrow f(r_A) = \epsilon$
    - $r_A = a \rightarrow f(r_A) = h(a)$
    - $r_A = r_1 r_2 \rightarrow f(r_A) = f(r_1) f(r_2)$
    - $r_A = r_1 + r_2 \rightarrow f(r_A) = f(r_1) + f(r_2)$
    - $r_A = r_1^* \rightarrow f(r_A) = f(r_1)^*$

$\square$

Deterministische endliche Automaten



DFA: Modell für Spracherkennung

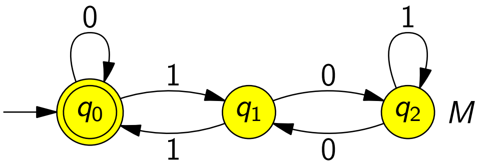
Formale Definition eines DFAs

Definition

Ein *deterministischer endlicher Automat* (DFA) ist ein 5-Tupel $M = (Q, \Sigma, \delta, q_0, F)$ mit

- Σ, dem *Eingabealphabet*,
- Q, der endlichen Menge der *Zustände*,
- $\delta\colon Q \times \Sigma \to Q$, der *Übergangsfunktion*,
- $q_0 \in Q$, dem *Anfangszustand* und
- $F \subseteq Q$, der Menge der *Endzustände*.

Beispiel



$M = (Q, \Sigma, \delta, q_0, F)$ wobei

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- $\delta\colon Q \times \Sigma \to Q$, $(q_0, 0) \mapsto q_0$, $(q_0, 1) \mapsto q_1$,
  $(q_1, 0) \mapsto q_2$, $(q_1, 1) \mapsto q_0$,
  $(q_2, 0) \mapsto q_1$, $(q_2, 1) \mapsto q_2$
- $F = \{q_0\}$

Die Sprache eines DFA

Definition

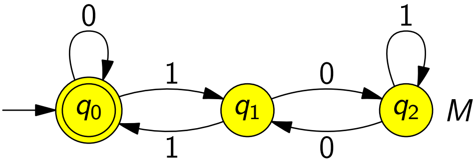
Es sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

Erweitere $\delta\colon Q \times \Sigma \to Q$ auf $\hat{\delta}\colon Q \times \Sigma^* \to Q$:

1. $\hat{\delta}(q, \epsilon) = q$
2. $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$ für $a \in \Sigma$, $w \in \Sigma^*$.

Definiere die *Sprache von M*, in Zeichen $L(M)$ als

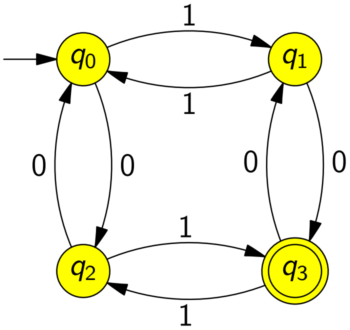
$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}.$$



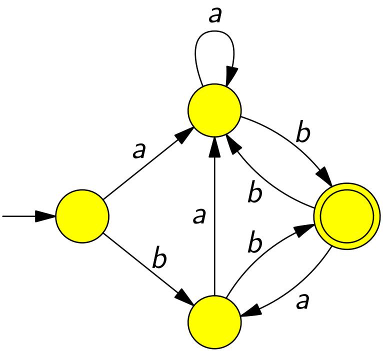
Wird das Wort 010 akzeptiert?

$$
\begin{aligned}
\hat{\delta}(q_0, 010) &= \delta(\hat{\delta}(q_0, 01), 0) \\
&= \delta(\delta(\delta(q_0, 0), 1), 0) \\
&= \delta(\delta(q_0, 1), 0) \\
&= \delta(q_1, 0) = q_2
\end{aligned}
$$

Nein, denn $\hat{\delta}(q_0, 010) \notin F$



Anschaulich: Welche Sprache wird akzeptiert?



Als regulärer Ausdruck: Welche Sprache wird akzeptiert?

$M = (Q, \Sigma, \delta, q_1, F)$ mit $Q = \{q_1, \cdots, q_n\}$.

Konstruiere einen regulären Ausdruck, der $L(M)$ erzeugt.

Definiere $L_{ij}^k \subseteq \Sigma^*$ für $1 \leq i, j \leq n$, $0 \leq k \leq n$:

$$L_{ij}^0 := \begin{cases} \{\, a \in \Sigma \mid \delta(q_i, a) = q_j \,\} & \text{falls } i \neq j, \\ \{\, a \in \Sigma \mid \delta(q_i, a) = q_i \,\} \cup \{\epsilon\} & \text{falls } i = j \end{cases}$$

$$L_{ij}^k := L_{ij}^{k-1} \cup L_{ik}^{k-1}(L_{kk}^{k-1})^* L_{kj}^{k-1} \text{ für } k > 0$$

Spätere Behauptung:

$$L(M) = \bigcup_{q_j \in F} L_{1j}^n$$

DFAs erkennen reguläre Sprachen

Theorem

M ein DFA $\Rightarrow L(M)$ regulär

Beweis

Idee: $w \in L_{ij}^k$ genau dann wenn

1. $\hat{\delta}(q_i, w) = q_j$
2. $\hat{\delta}(q_i, u) = q_m$ mit $m \leq k$ für alle $u \sqsubseteq w$, $u \neq \epsilon$, $u \neq w$ ($u \sqsubseteq v$ gdw. $ux = v$ für ein x)

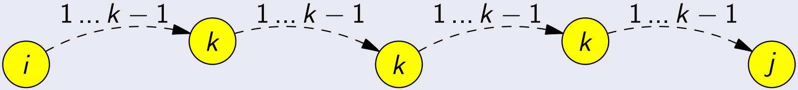
Anschaulich:

1. w bringt M von q_i nach q_j.
2. Dazwischen durchläuft M nur Zustände aus $\{q_1, \dots, q_k\}$.

Beweis.

$$L_{ij}^0 := \begin{cases} \{\, a \in \Sigma \mid \delta(q_i, a) = q_j \,\} & \text{falls } i \neq j, \\ \{\, a \in \Sigma \mid \delta(q_i, a) = q_i \,\} \cup \{\epsilon\} & \text{falls } i = j, \end{cases}$$

$$L_{ij}^k := L_{ij}^{k-1} \cup L_{ik}^{k-1} (L_{kk}^{k-1})^* L_{kj}^{k-1} \text{ für } k > 0$$



Korrektheit: Induktion über k.

Es ist leicht reguläre Ausdrücke für L_{ij}^k zu finden.

Regulärer Ausdruck für $L(M) = \bigcup_{q_j \in F} L_{1j}^n$. $\square$

Der Komplementärautomat

Theorem

Es sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA.
Dann ist $\Sigma^* - L(M)$ regulär.

$\Sigma^* - L$ ist das Komplement von L.

Beweis.

$M' := (Q, \Sigma, \delta, q_0, Q - F)$.
Es gilt $L(M') = \Sigma^* - L(M)$:

$$\hat{\delta}(q_0, w) \in F \Leftrightarrow \hat{\delta}(q_0, w) \notin Q - F$$

$\square$

Der Produktautomat

Definition

Es seien $M' = (\Sigma, Q', \delta', q'_0, F')$ und $M'' = (\Sigma, Q'', \delta'', q''_0, F'')$ zwei DFAs.

Wir definieren den *Produktautomaten* $M = M' \times M''$:

$M = (\Sigma, Q' \times Q'', \delta, (q'_0, q''_0), F' \times F'')$
mit $\delta((q, p), a) = (\delta'(q, a), \delta''(p, a))$.

Theorem

Wenn M' und M'' DFAs sind und $M = M' \times M''$, dann $L(M) = L(M') \cap L(M'')$.

Beweis.

$\hat{\delta}((q_0', q_0''), w) = (\hat{\delta}'(q_0', w), \hat{\delta}''(q_0'', w))$
(Induktion über $|w|$)
und damit

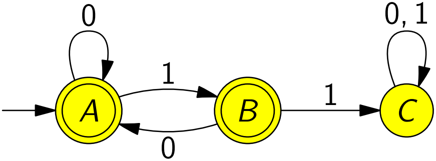
$$\begin{aligned}
w \in L(M) &\Leftrightarrow \hat{\delta}((q_0', q_0''), w) \in F' \times F'' \\
&\Leftrightarrow (\hat{\delta}'(q_0', w), \hat{\delta}''(q_0'', w)) \in F' \times F'' \\
&\Leftrightarrow \hat{\delta}'(q_0', w) \in F' \text{ und } \hat{\delta}''(q_0'', w) \in F'' \\
&\Leftrightarrow w \in L(M') \text{ und } w \in L(M'').
\end{aligned}$$

Daher ist $L(M) = L(M') \cap L(M'')$. $\square$

Beispiel

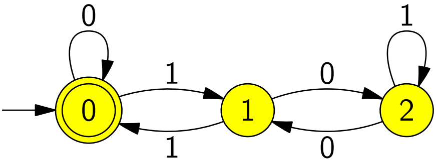
Konstruiere DFA für Sprache aller w mit:

1. Es kommt 11 nicht als Unterwort in w vor.



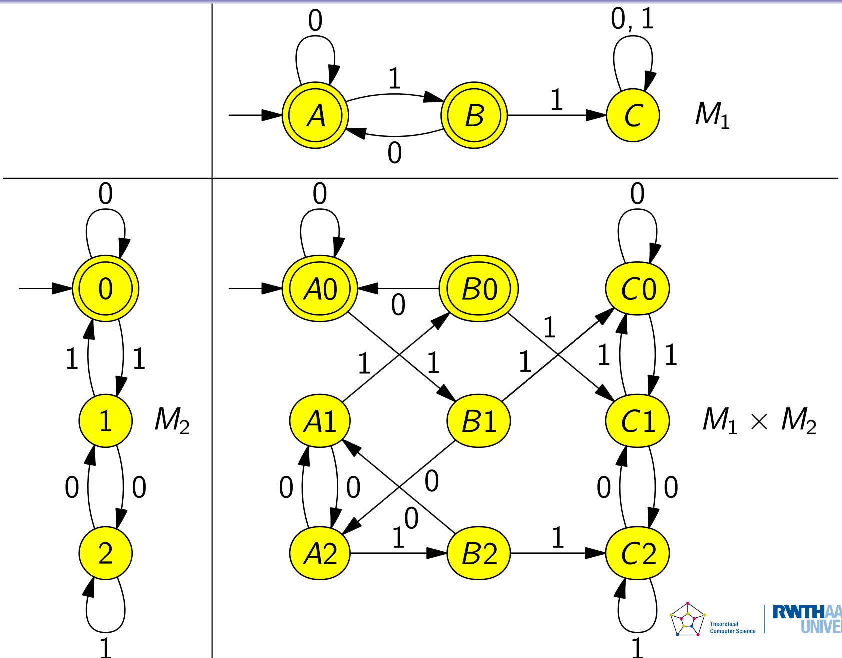
M_1

2. Als Binärzahl ist w durch drei teilbar.



M_2

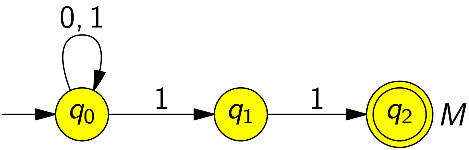
Konstruiere $M_1 \times M_2$!


A
B
C
M_1
0
1
2
M_2
A0
B0
C0
A1
B1
C1
A2
B2
C2
$M_1 \times M_2$

Einige Vorteile endlicher deterministischer Automaten:

- durch Computer *schnell* simulierbar
- wenig Speicher benötigt:
  Tabelle für δ (read-only), aktueller Zustand
- Eingabe kann vergessen werden, nur von links nach rechts lesen
- Sie können schön visualisiert werden
- Sie können automatisch generiert werden (z.B. lex, egrep)

Nichtdeterministische endliche Automaten (NFAs)



Dies ist kein DFA!

1. Zwei Transitionen mit 1 aus q_0
2. Keine Transition mit 0 aus q_1

Welche Sprache soll M erkennen?

Nichtdeterministische endliche Automaten (NFAs)

Definition

Ein NFA ist ein 5-Tupel $M = (Q, \Sigma, \delta, q_0, F)$ mit

- Q Menge der *Zustände*
- Σ *Eingabealphabet*
- $\delta\colon Q \times \Sigma \to 2^Q$ *Übergangsfunktion*
- $q_0 \in Q$ *Startzustand*
- $F \subseteq Q$ *Endzustände*

Definition

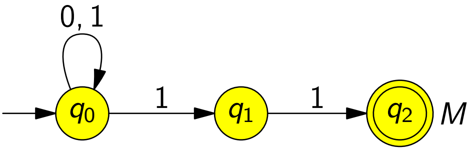
Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein NFA.

$\hat{\delta}\colon Q \times \Sigma^* \to 2^Q$ definiert durch

- $\hat{\delta}(q, \epsilon) = \{q\}$
- $\hat{\delta}(q, wa) = \{p \mid \text{es gibt } r \in \hat{\delta}(q, w) \text{ und } p \in \delta(r, a)\}$

$L(M) := \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$

Beispiel



- $\delta(q_0, 0) = \{q_0\}$
- $\delta(q_0, 1) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 010110101101) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 11111) = \{q_0, q_1, q_2\}$

$L(M) = (0 + 1)^*11$

Der Potenzautomat

Definition

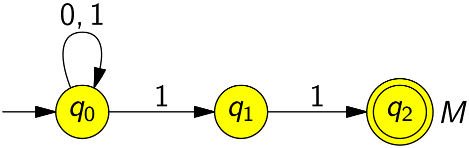
Sei M ein NFA, $M = (Q, \Sigma, \delta, q_0, F)$

Der zugehörige *Potenzautomat* M' ist so aufgebaut:

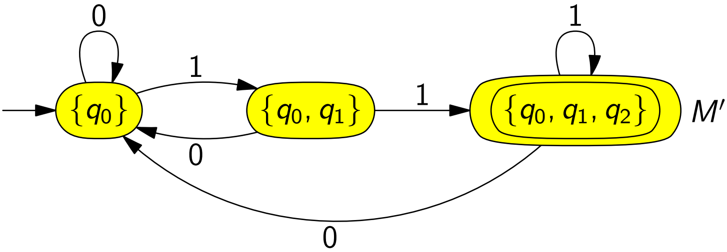
- $M' = (2^Q, \Sigma, \delta', \{q_0\}, F')$ mit
- $\delta' : 2^Q \times \Sigma \to 2^Q, (S, a) \mapsto \bigcup_{q \in S} \delta(q, a)$
- $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$

Der Potenzautomat ist ein DFA!

Beispiel



Der Potenzautomat hat die Zustände $\emptyset$, $\{q_0\}$, $\{q_1\}$, $\{q_2\}$, $\{q_0, q_1\}$, $\{q_0, q_2\}$, $\{q_1, q_2\}$ und $\{q_0, q_1, q_2\}$ und sieht so aus:



Nichterreichbare Zustände weggelassen!

Theorem

Zu jedem NFA M gibt es einen DFA M' mit $L(M) = L(M')$

Beweis.

$L(M) = L(M')$ für den Potenzautomaten M':

- $M = (Q, \Sigma, \delta, q_0, F)$
- $M' = (2^Q, \Sigma, \delta', \{q_0\}, F')$ mit
- $\delta' : 2^Q \times \Sigma \to 2^Q, (S, a) \mapsto \bigcup_{q \in S} \delta(q, a)$
- $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$

Induktion über $|w|$: $\hat{\delta}'(\{q_0\}, w) = \hat{\delta}(q_0, w)$

Daher:

$$\hat{\delta}'(\{q_0\}, w) \in F' \iff \hat{\delta}(q_0, w) \cap F \neq \emptyset \qquad \square$$

Vergleich: DFA und NFA

Vorteile eines DFA:

- Effizient simulierbar

Vorteile eines NFA:

- Oft kleiner als DFA
- Einfacher zu entwerfen
- Halbwegs effizient simulierbar