# Tutorium
# Automaten und Formale Sprachen
## Teil II.

Christopher Blöcker, B. Sc.
inf9900@fh-wedel.de

SS 2012

## Definition

Zwei Automaten $\mathcal{A}$ und $\mathcal{A}'$ sind äquivalent „$\simeq$“, wenn sie dieselbe Sprache akzeptieren.

$$\mathcal{A} \simeq \mathcal{A}' \Leftrightarrow \mathcal{L}_{\mathcal{A}} = \mathcal{L}_{\mathcal{A}'},$$

also

$$\bigwedge_{w \in \Sigma^*} w \in \mathcal{L}_{\mathcal{A}} \Leftrightarrow w \in \mathcal{L}_{\mathcal{A}'}.$$

## Konstruktion

Ein zu einem **NEA** $\mathcal{A}$ äquivalenter **DEA** $\mathcal{A}'$ kann durch die *Potenzmengenkonstruktion* bestimmt werden.

## Potenzmengenkonstruktion

Sei $\mathcal{A}$ ein **NEA** mit

$$\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F}).$$

Der **DEA** $\mathcal{A}'$ mit

$$\mathcal{A}' \simeq \mathcal{A}$$

und

$$\mathcal{A}' = (\mathcal{Q}', \Sigma', \delta', q_0', \mathcal{F}')$$

wird bestimmt durch:

1. $\mathcal{Q}' \leftarrow \mathcal{P}(\mathcal{Q})$
2. $\Sigma' \leftarrow \Sigma \setminus \{\epsilon\}$
3. $\delta' \leftarrow \{(q, a, \epsilon(r)) \mid (q \in \mathcal{Q}') \wedge (a \in \Sigma') \wedge (r \leftarrow \bigcup_{s \in q} \delta(s, a))\}$
4. $q_0' \leftarrow \epsilon(\{q_0\})$
5. $\mathcal{F}' \leftarrow \{q \in \mathcal{Q}' \mid q \cap \mathcal{F} \neq \emptyset\}$

Dabei bezeichne $\epsilon(q)$ die Epsilon-Hülle von $q$.

## Problem

Die Überführungsfunktion $\delta$ eines **DEA** ist nicht surjektiv.
(Es gibt Zustände, die nicht erreicht werden können.)

## Eliminierung unnützer Zustände

Die Markierungen der Transitionen sind nicht relevant, der Graph des **DEA** wird als einfacher Digraph betrachtet.

Die Menge der erreichbaren Zustände lässt sich durch eine Fixpunktiteration bestimmen.

1. $\mathcal{Q}_0 \leftarrow \{q_0\}$
2. $\mathcal{Q}_{i+1} \leftarrow \mathcal{Q}_i \cup \mathcal{N}(\mathcal{Q}_i)$

Dabei bezeichne $\mathcal{N}(\mathcal{Q}_i)$ die von $\mathcal{Q}_i$ aus erreichbaren Zustände (die Nachbarn von $\mathcal{Q}_i$).

Die Iteration wird fortgesetzt bis $\mathcal{Q}_{i+1} = \mathcal{Q}_i$.

## Assoziativität

- $(\mathcal{R}_1 \circ \mathcal{R}_2) \circ \mathcal{R}_3 = \mathcal{R}_1 \circ (\mathcal{R}_2 \circ \mathcal{R}_3)$
- $(\mathcal{R}_1 + \mathcal{R}_2) + \mathcal{R}_3 = \mathcal{R}_1 + (\mathcal{R}_2 + \mathcal{R}_3)$

## Kommutativität

- $\mathcal{R}_1 + \mathcal{R}_2 = \mathcal{R}_2 + \mathcal{R}_1$

## Distributivität

- $(\mathcal{R}_1 + \mathcal{R}_2) \circ \mathcal{R}_3 = \mathcal{R}_1 \circ \mathcal{R}_3 + \mathcal{R}_2 \circ \mathcal{R}_3$

## Neutrale Elemente

- $\epsilon \circ \mathcal{R} = \mathcal{R} = \mathcal{R} \circ \epsilon$
- $\emptyset + \mathcal{R} = \mathcal{R} = \mathcal{R} + \emptyset$

## Sonderfall: Leere Menge

- $\mathcal{R} \circ \emptyset = \emptyset = \emptyset \circ \mathcal{R}$

## Idempotenz

1. $(\mathcal{R}^*)^* = \mathcal{R}^*$
2. $(\mathcal{R}^+)^* = \mathcal{R}^*$
3. $(\mathcal{R}^*)^+ = \mathcal{R}^*$
4. $\epsilon^* = \epsilon^+ = \epsilon$
5. $\epsilon \circ \mathcal{R}^* = \mathcal{R}^* = \mathcal{R}^* \circ \epsilon$
6. $\mathcal{R}^* \circ \mathcal{R}^* = \mathcal{R}^*$
7. $\mathcal{R}^* + \mathcal{R}^* = \mathcal{R}^*$
8. $\mathcal{R}^+ \circ \mathcal{R}^+ = \mathcal{R} \circ \mathcal{R} \circ \mathcal{R}^*$
9. $\mathcal{R}^+ + \mathcal{R}^+ = \mathcal{R}^+$
10. $\mathcal{R}^* \circ \mathcal{R}^+ = \mathcal{R}^+ = \mathcal{R}^+ \circ \mathcal{R}^*$

## Die Chomsky-Hierarchie

Noam Chomsky hat den Begriff der Chomsky-Hierarchie geprägt und die folgenden 4 Sprachklassen definiert:

- Typ-3 : Reguläre Sprachen $\mathcal{L}_3$
- Typ-2 : Kontext-Freie Sprachen $\mathcal{L}_2$
- Typ-1 : Kontext-Sensitive Sprachen $\mathcal{L}_1$
- Typ-0 : Rekursiv Aufzählbare Sprachen $\mathcal{L}_0$

Es gilt:

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$$

Jede dieser Sprachklassen lässt sich durch Grammatiken des jeweiligen Types erzeugen. Dabei unterliegen die Produktionen gewissen Restriktionen.

## Anmerkung

Es gibt (überabzählbar viele) Sprachen, die **nicht einmal** rekursiv aufzählbar sind!

## Frage

Wie lässt sich feststellen, ob

$$\mathcal{L} \in \mathcal{L}_3?$$

## Antwort

Eine Sprache ist vom Typ-3, wenn sie von einem endlichen Automaten akzeptiert wird.

Da Endliche Automaten, Reguläre Ausdrücke und Typ-3-Grammatiken äquivalent sind (sie lassen sich ineinander überführen) ist für den Nachweis $\mathcal{L} \in \mathcal{L}_3$ ein Endlicher Automat $\mathcal{A}$, ein Regulärer Ausdruck $\mathcal{R}$ oder eine Typ-3-Grammatik $\mathcal{G}$ mit

$$\mathcal{L}_{\mathcal{A}} = \mathcal{L} \text{ bzw. } \mathcal{L}_{\mathcal{R}} = \mathcal{L} \text{ bzw. } \mathcal{L}_{\mathcal{G}} = \mathcal{L}$$

anzugeben.

## Frage

Wie lässt sich feststellen, ob

$$\mathcal{L} \notin \mathcal{L}_3?$$

## Antwort

Mit Hilfe des Pumping-Lemmas.

Das Pumping-Lemma gilt für jede reguläre Sprache.

$$\mathcal{L} \in \mathcal{L}_3 \rightarrow \mathrm{PL}\left(\mathcal{L}\right).$$

Wenn die Bedingungen des Pumping-Lemmas nicht erfüllt sind, so kann die betroffene Sprache nicht regulär sein.

Durch Kontraposition erhält man

$$\overline{\mathrm{PL}(\mathcal{L})} \rightarrow \mathcal{L} \notin \mathcal{L}_3.$$

## Das Pumping-Lemma

Sei $\mathcal{L} \subseteq \Sigma^*$ eine reguläre Sprache.
Dann gibt es eine natürliche Zahl $p \in \mathbb{N}$ derart, dass sich jedes Wort $x \in \mathcal{L}$ mit $|x| \geq p$ zerlegen lässt in drei Teilworte

$$x = uvw$$

mit

1. $|v| > 0$
2. $|uv| \leq p$
3. $\bigwedge_{i \in \mathbb{N}} uv^i w \in \mathcal{L}$

## Vorsicht!

Es gilt

$$\mathcal{L} \in \mathcal{L}_3 \rightarrow \mathrm{PL}(\mathcal{L}).$$

Die Umkehrung gilt jedoch **nicht**! Die folgende Aussage ist also **falsch**:

$$\lightning\ \mathrm{PL}(\mathcal{L}) \rightarrow \mathcal{L} \in \mathcal{L}_3.\ \lightning$$

## Problem

Es gibt Sprachen mit $\mathcal{L} \notin \mathcal{L}_3$, für die aber das Pumping-Lemma erfüllt ist.

## Nerode-Relation

Seien $x, y \in \Sigma^*, \mathcal{L} \subseteq \Sigma^*, \mathcal{R} \subseteq \Sigma^* \times \Sigma^*$.

$$\mathcal{R}_{\mathcal{N}} : x \sim y \Leftrightarrow \bigwedge_{z \in \Sigma^*} (xz \in \mathcal{L}) \Leftrightarrow (yz \in \mathcal{L})$$

## Satz von Myhill und Nerode

Wenn der Index der Nerode-Relation endlich ist, also wenn es nur endlich viele Äquivalenzklassen bezüglich $\mathcal{R}_{\mathcal{N}}$ gibt, so gilt $\mathcal{L} \in \mathcal{L}_3$.

## Problem

Die Darstellung eines Endlichen Automaten ist nicht eindeutig, es gibt (abzählbar) unendlich viele Automaten, die alle dieselbe Sprache akzeptieren.

Außerdem kann ein Automat Zustände enthalten, die nutzlos sind oder aber welche, die äquivalent zueinander sind.

## Automaten-Minimierung

Bei der Minimierung sollen unnütze Zustände entfernt und äquivalente Zustände zu Äquivalenzklassen zusammengefasst werden.

Ausgehend von einem Automaten $\mathcal{A}$ entsteht der sogenannte Äquivalenzklassenautomat $\mathcal{A}_{\sim}$ mit $\mathcal{A} \simeq \mathcal{A}_{\sim}$.

## Ablauf der Minimierung

Sei $\mathcal{A}$ ein DEA.

1. Entfernen unerreichbarer Zustände
2. Bilden der Äquivalenzklassen

## Definition der Äquivalenzklassen

$$\mathcal{R} \subseteq \mathcal{Q} \times \mathcal{Q}$$

mit

$$\mathcal{R} : q_i \sim q_j \Leftrightarrow \bigwedge_{w \in \Sigma^*} \left[\delta^* (q_i, w) \in \mathcal{F}\right] \Leftrightarrow \left[\delta^* (q_j, w) \in \mathcal{F}\right].$$

## Anmerkung

Die entstehenden Äquivalenzklassen stimmen mit denen der NERODE-Relation überein.

## Nerode-Relation

Seien $x, y \in \Sigma^*, \mathcal{L} \subseteq \Sigma^*, \mathcal{R} \subseteq \Sigma^* \times \Sigma^*$.

$$\mathcal{R}_{\mathcal{N}} : x \sim y \Leftrightarrow \bigwedge_{z \in \Sigma^*} (xz \in \mathcal{L}) \Leftrightarrow (yz \in \mathcal{L})$$

## Definition der Äquivalenzklassen

$$\mathcal{R} \subseteq \mathcal{Q} \times \mathcal{Q}$$

mit

$$\mathcal{R} : q_i \sim q_j \Leftrightarrow \bigwedge_{w \in \Sigma^*} \left[\delta^* (q_i, w) \in \mathcal{F}\right] \Leftrightarrow \left[\delta^* (q_j, w) \in \mathcal{F}\right].$$

## Definition der Äquivalenzklassen

$$\mathcal{R} : q_i \sim q_j \Leftrightarrow \bigwedge_{w \in \Sigma^*} \left[\delta^* (q_i, w) \in \mathcal{F}\right] \Leftrightarrow \left[\delta^* (q_j, w) \in \mathcal{F}\right].$$

## Problem

Es muss für alle Wörter aus $\Sigma^*$ geprüft werden, ob die Bedingung erfüllt ist. Das ist aber nicht möglich.

Man verschafft sich dadurch Abhilfe, dass man die negierte Aussage verwendet

$$\mathcal{R} : q_i \nsim q_j \Leftrightarrow \bigvee_{w \in \Sigma^*} \left[\delta^* (q_i, w) \in \mathcal{F}\right] \wedge \left[\delta^* (q_j, w) \notin \mathcal{F}\right].$$

Ein solches $w$ nennen wir einen Zeugen. Das Auffinden von Zeugen geschieht durch eine Fixpunktiteration mit einem Table-Filling-Verfahren.

## Auffinden von Zeugen

$$\mathcal{R} : q_i \nsim q_j \Leftrightarrow \bigvee_{w \in \Sigma^*} \left[\delta^* (q_i, w) \in \mathcal{F}\right] \wedge \left[\delta^* (q_j, w) \notin \mathcal{F}\right].$$

- Erzeuge leere Matrix
- Markiere End- und Nichtendzustände als nicht äquivalent
- Prüfe jedes (bisher nicht markierte) Zustandspaar mit jedem Eingabe**zeichen** auf Äquivalenz bis keine Paare mehr markiert werden
- Fasse die Zustände der nicht markierten Paare zusammen

Das Überprüfen ganzer Wörter ist aufgrund der Transitivität der Äquivalenzrelation nicht notwendig und wird durch die Fixpunktiteration „simuliert“.