Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik
Endliche Automaten und reguläre Ausdrücke

INSTITUT FÜR THEORETISCHE INFORMATIK

Organisatorisches

- Termine
- Homepage
- Übungsblätter
- Tutorien
- Klausur

Team

Vorlesung:
Prof. Dorothea Wagner

Übung:
Andrea Schumm (Andrea.Schumm@kit.edu),
Sprechstunde Dienstags 10:00-11:00 Uhr

Tanja Hartmann (Tanja.Hartmann@kit.edu),
Sprechstunde Freitags 9:00-10:00 Uhr

Tutorium:
Yvonne Braun, Joachim Priesner, Philipp Schneider, Steffen Kühner, Marcel Radermacher, Alexander Degitz, Steffen Kühner, Lukas Barth, Kai Wallisch, Tobias Maier, Sebastian Ulrich, Roland Gröll, Max Wagner, Stefan Altmayer, Sarah Lutteropp, Benno Evers, Philipp Loewner

Homepage

- `http://i11www.iti.kit.edu/`
- Aktuelle Informationen / Termine
- Alte Klausuren
- Skript
- Folien
- Übungsblätter
- Forum
  - Für Fragen an die Übungsleiter
  - Für Austausch untereinander
- Literaturempfehlungen

Literaturempfehlungen

- Ingo Wegener
  Theoretische Informatik
  B.G. Teubner Verlag Stuttgart, 1993
- Uwe Schöning
  Theoretische Informatik - kurzgefasst
  Hochschultaschenbuch, Spektrum Akademischer Verlag, 1997

Literaturempfehlungen

- R. Garey und D. S. Johnson
  Computers and Intractibility: A Guide to the Theory of NP-Completeness
  W. H. Freeman, New York, 1979
- T. Cormen, C. Leiserson, R. Rivest
  Introduction to Algorithms
  The MIT press, 1997, 2001.
- A. Asteroth, C. Baier
  Theoretische Informatik: eine Einführung in Berechenbarkeit, Komplexität und formale Sprachen mit 101 Beispielen
  Pearson Studium, 2002

Übungsblätter

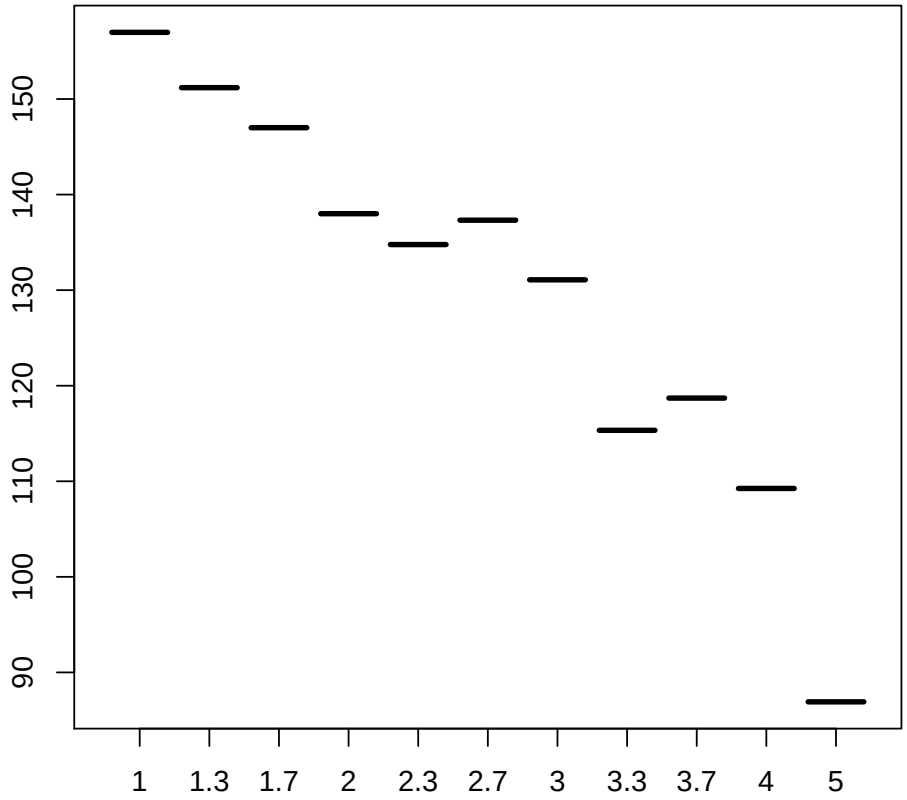
In der Regel:

- Ausgabe jeden zweiten Donnerstag
- Bearbeitungszeit: 2 Wochen
- Abgabe: Kasten im Untergeschoss des Informatik-Hauptgebäudes (50.34)
- Abgabe bis Donnerstag, 11.00 Uhr im Kasten oder bis zum Start der Übung im Hörsaal
- Rückgabe der korrigierten Blätter in den Tutorien
- Lösungen in der Übung
- Ausgabe von erstem Übungsblatt: Donnerstag 20.10.2011

Übungsblätter

- Doppelabgabe erlaubt, aber nur, wenn beide für das gleiche Tutorium eingeteilt sind
- Kopf des Blattes:
  - Name(n)
  - Vorname(n)
  - Matrikelnummer(n)
  - Tutoriumsnummer
  - Name des Tutors
- Keine losen Blätter, bitte zusammenheften
- Übungsblätter müssen handschriftlich sein, also nicht gedruckt oder kopiert
- Bei Abschreiben: Keine Punkte
- Ab 50% der erreichbaren Punkte gibt es einen Klausurbonus
- Der Klausurbonus wird nur auf bestandene Klausuren angerechnet

Statistik aus dem WS07/08



Tutorien

- Rückgabe der Übungsblätter
- Zusätzliche Aufgaben und Beispiele zum Stoff der Vorlesung
- Start: Nächste Woche (ab 27.10.2011)
- Einteilung über webinscribe, Homepage:
  `http://webinscribe.informatik.kit.edu/`
- Dort verlinkt: Merkblatt und Termine der Tutorien
- Anmeldebeginn: Dienstag, 18.10., 18:00 Uhr
- Anmeldeschluss: Donnerstag, 20.10., 18:00 Uhr!

Klausur

- Orientierung:
  Klausuren auf der Homepage
  aber: Im Vergleich zu Informatik III leicht veränderte Stoffwahl und 2-Stunden-Klausur
- Klausurbonus ab 50% der erreichbaren Übungsblattpunkte
- Klausurbonus wird nur auf bestandene Klausuren angerechnet
- Termine für Haupt- und Nachklausur werden noch bekannt gegeben

Ziele der Vorlesung TGI

Welche Fragestellungen werden in TGI behandelt?

- Theoretische Grundlagen zu Algorithmen- und Programmentwurf: Wie kann man allgemeingültige (rechner- und programmiersprachenunabhängige) Aussagen zu gegeben Problemstellungen machen?
- Gibt es Probleme die nicht von Computern gelöst werden können?
- Gibt es Probleme, für die Ausprobieren die beste Lösungsstrategie ist?
- Wie kann man konkrete Computer abstrakt betrachten oder modellieren?
- Wie kann man konkrete Programmiersprachen abstrakt betrachten oder modellieren?

Ziele der Vorlesung TGI

- Vertiefter Einblick in die Grundlagen der Theoretischen Informatik
- Beherrschen der Berechnungsmodelle und Beweistechniken der TI
- Verständnis für Grenzen und Möglichkeiten der Informatik in Bezug auf die Lösung von definierbaren aber nur bedingt berechenbaren Problemem
- Abstraktionsvermögen für grundlegende Aspekte der Informatik von konkreten Gegebenheiten wie konkreten Rechnern oder Programmiersprachen
- Anwendung der erlernten Beweistechniken bei der Spezifikation von Systemen der Informatik und für den systematischen Entwurf von Programmen und Algorithmen

Wiederholung aus Grundbegriffe der Informatik

- Wörter
- Formale Sprachen
- Reguläre Ausdrücke
- Endliche Automaten
- Kontextfreie Grammatiken

Wörter

- Ein *Alphabet* Σ ist eine endliche Menge von Zeichen.
  Beispiel: $\Sigma = \{0, 1\}$.
- Ein *Wort w* über einem Alphabet Σ ist eine (möglicherweise leere) Folge von Zeichen aus Σ.
  Beispiel: $\Sigma = \{0, 1\}$ und $w = 0010010$.
- Das *leere Wort* wird mit ε symbolisiert.
- Die *Menge aller Wörter* über einem Alphabet Σ wird mit Σ^* abgekürzt.
  Beispiel:
  $\Sigma = \{0, 1\}$ und $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

Wörter

- Die *Menge aller Wörter* mit Länge n über einem Alphabet Σ wird mit Σ^n abgekürzt. Beispiel $\Sigma = \{0, 1\}$:

$$\begin{aligned} \Sigma^0 &= \{\varepsilon\} \\ \Sigma^1 &= \{0, 1\} \\ \Sigma^2 &= \{00, 01, 10, 11\} \end{aligned}$$

- Die *Konkatenation* (Aneinanderhängen) zweier Wörter w_1 und w_2 wird mit $w_1 \cdot w_2$ oder $w_1 w_2$ abgekürzt.
  Beispiel: $w_1 = 001$ und $w_2 = 110$ dann $w_1 \cdot w_2 = 001110$.

Wörter

- Die *iterierte Konkatenation* eines Wortes w ist $w^k := \underbrace{w \cdot \ldots \cdot w}_{k\text{-mal}}$

  Beispiel: $w = 110$.

$$
\begin{aligned}
w^0 &= \epsilon \\
w^1 &= 110 \\
w^2 &= 110110 \\
w^3 &= 110110110
\end{aligned}
$$

Formale Sprachen

- Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge $L \subseteq \Sigma^*$.

  Beispiel: Sprache L' aller Wörter deren vorletztes Zeichen 0 ist

  $$L' = \{w0z \mid w \in \Sigma^*, z \in \Sigma\}$$

- Das Produkt $L_1 \cdot L_2$ der Sprachen L_1 und L_2 ist definiert als

  $$L_1 \cdot L_2 := \{w_1 w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$$

  Beispiel:

  $$L' =$$

Formale Sprachen

- Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge $L \subseteq \Sigma^*$.

  Beispiel: Sprache L' aller Wörter deren vorletztes Zeichen 0 ist

$$L' = \{w0z \mid w \in \Sigma^*, z \in \Sigma\}$$

- Das Produkt $L_1 \cdot L_2$ der Sprachen L_1 und L_2 ist definiert als

$$L_1 \cdot L_2 := \{w_1 w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$$

  Beispiel:

$$L' = \Sigma^* \cdot \{00, 01\}$$

Formale Sprachen

- Produkte einer Sprache L mit sich selbst werden abgekürzt als

$$L^k := \underbrace{L \cdot \ldots \cdot L}_{k\text{-mal}}$$

Beispiel $L = \{00, 1\}$:

$$\begin{aligned}
L^0 &= \{\varepsilon\} \\
L^1 &= \{00, 1\} \\
L^2 &= \{00, 1\} \cdot \{00, 1\} = \{0000, 001, 100, 11\} \\
L^3 &= \{0000, 001, 100, 11\} \cdot \{00, 1\} \\
&= \{000000, 00100, 10000, 1100, 00001, 0011, 1001, 111\}
\end{aligned}$$

Formale Sprachen

Läßt sich ein Wort w schreiben als $w = u \cdot v \cdot x$, wobei u, v, x beliebige Wörter sind, so heißt:

$$\left.\begin{array}{ll} u & \text{Präfix} \\ v & \text{Teilwort} \\ x & \text{Suffix} \end{array}\right\} \text{von } w$$

Beispiel w=TAL

$$\begin{array}{ll} \text{Präfixe} & P = \{\varepsilon, \text{T}, \text{TA}, \text{TAL}\} \\ \text{Suffixe} & S = \{\varepsilon, \text{L}, \text{AL}, \text{TAL}\} \\ \text{Teilworte} & \{A\} \cup P \cup S \end{array}$$

Formale Sprachen

Seien $L, L_1, L_2 \subseteq \Sigma^*$ Sprachen.

Produktsprache	$L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
k-faches Produkt	$L^k := \{w_1 \cdot w_2 \cdot \ldots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$ $L^0 := \{\epsilon\}$
Quotientensprache	$L_1 / L_2 := \{w \in \Sigma^* \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$
Kleene'scher Abschluss	$L^* := \bigcup_{i \geq 0} L^i$
Positiver Abschluss	$L^+ := \bigcup_{i \geq 1} L^i$
Komplementsprache	$L^c := \Sigma^* \setminus L$

Reguläre Sprachen

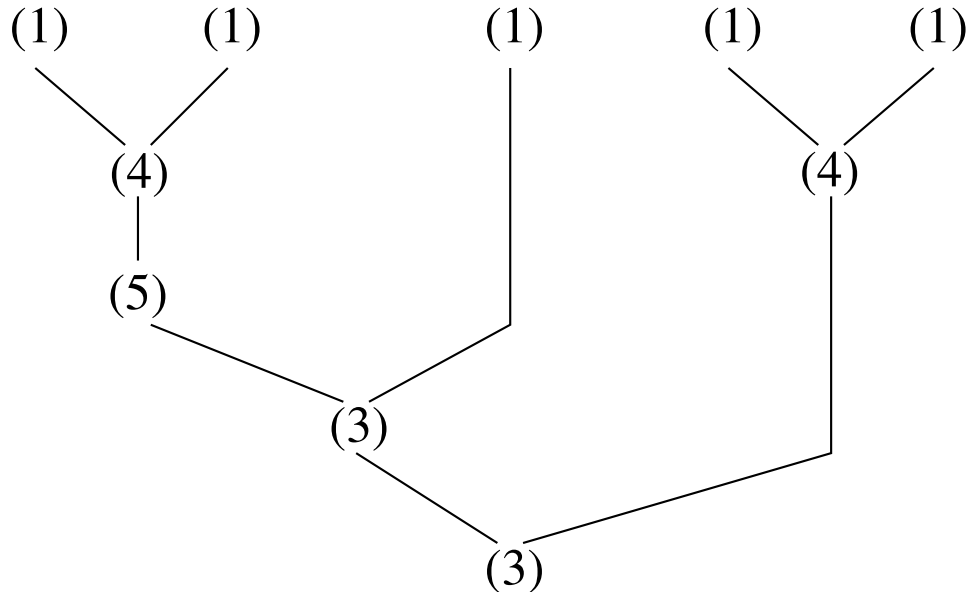
Eine Sprache $L \subseteq \Sigma^*$ heißt *regulär*, wenn für sie einer der folgenden Punkte gilt: (induktive Definition)

1. Verankerung:
   1. $L = \{a\}$ mit $a \in \Sigma$ oder
   2. $L = \emptyset$
2. Induktion: Seien L_1, L_2 reguläre Sprachen
   1. $L = L_1 \cdot L_2$ oder
   2. $L = L_1 \cup L_2$ oder
   3. $L = L_1^*$

Reguläre Sprachen

Beispiel: Die Sprache aller Wörter über $\{0, 1\}$, die als vorletztes Zeichen eine 0 haben

$$L := (\{0\} \cup \{1\})^* \cdot \{0\} \cdot (\{0\} \cup \{1\})$$



Reguläre Ausdrücke

- Wir benutzen eine leicht andere Schreibweise für reguläre Ausdrücke als in GBI.
- Sei Σ eine Alphabet. Eine reguläre Sprache über Σ kann durch einen *regulären Ausdruck* beschrieben werden.

Reguläre Ausdrücke

Dabei bezeichnet

- $\emptyset$ den regulären Ausdruck, der die leere Menge beschreibt.
- ε den regulären Ausdruck, der die Menge $\{\varepsilon\}$ beschreibt.
- a den regulären Ausdruck, der die Menge $\{a\}$ beschreibt.

Wenn α, β reguläre Ausdrücke sind, die die Sprachen $L(\alpha), L(\beta)$ beschreiben, so schreiben wir

- $(\alpha) \cup (\beta)$ für $L(\alpha) \cup L(\beta)$
- $(\alpha) \cdot (\beta)$ für $L(\alpha) \cdot L(\beta)$
- $(\alpha)^+$ für $L(\alpha)^+$
- $(\alpha)^*$ für $L(\alpha)^*$

Reguläre Ausdrücke

Notation

- Wir schreiben auch α statt $L(\alpha)$ und $w \in \alpha$ statt $w \in L(\alpha)$.
- * bindet stärker als $\cdot$ und $\cdot$ stärker als $\cup$
- Das heißt zum Beispiel:

$$a \cup b \cdot c = a \cup (b \cdot c)$$

- Wir lassen unnötige Klammern weg.

Reguläre Ausdrücke - Beispiele

- L ist der reguläre Ausdruck für die Sprache aller Wörter über $\{0,1\}$, die als vorletztes Zeichen eine 0 haben

$$L = (0 \cup 1)^* 0 (0 \cup 1)$$

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 10 als Teilwort}\}$

$$L = (0 \cup 1)^* 10 (0 \cup 1)^*$$

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 101 als Teilwort}\}$

$$L = (0 \cup 1)^* 101 (0 \cup 1)^*$$

Reguläre Ausdrücke - Beispiele

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 10 nicht als Teilwort}\} = 0^*1^*$,

  denn
  - $w \in 0^*1^* \Rightarrow w \in L$; also ist $0^*1^* \subseteq L$.
  - Sei $w \in L$. Dann kommen nach der ersten Eins keine Nullen mehr vor, d.h. $w = w'1\ldots1$ wobei w' keine 1 enthält. Also ist $w \in 0^*1^*$.

Endliche Automaten

- In GBI wurden Mealy- und Moore-Automaten behandelt.
- In TGI werden nur endliche Akzeptoren benötigt.

Endliche Automaten

Ein (deterministischer) endlicher Automat DEA $(Q, \Sigma, \delta, s, F)$ besteht aus:

- Q, einer endlichen Menge von *Zuständen*
- Σ, einer endlichen Menge von *Eingabesymbolen*
- $\delta\colon Q \times \Sigma \to Q$, einer *Übergangsfunktion*
- $s \in Q$, einem *Startzustand*;
- $F \subseteq Q$, einer Menge von *Endzuständen*.

Endliche Automaten

Der Automat heißt

- endlich, da die Zustandsmenge (vgl. mit Speicher, Gedächtnis) endlich ist.
- deterministisch, da δ eine Funktion ist und der Automat somit in jedem Schritt eindeutig arbeitet. Es gibt keine Zufälligkeiten oder Wahlmöglichkeiten.

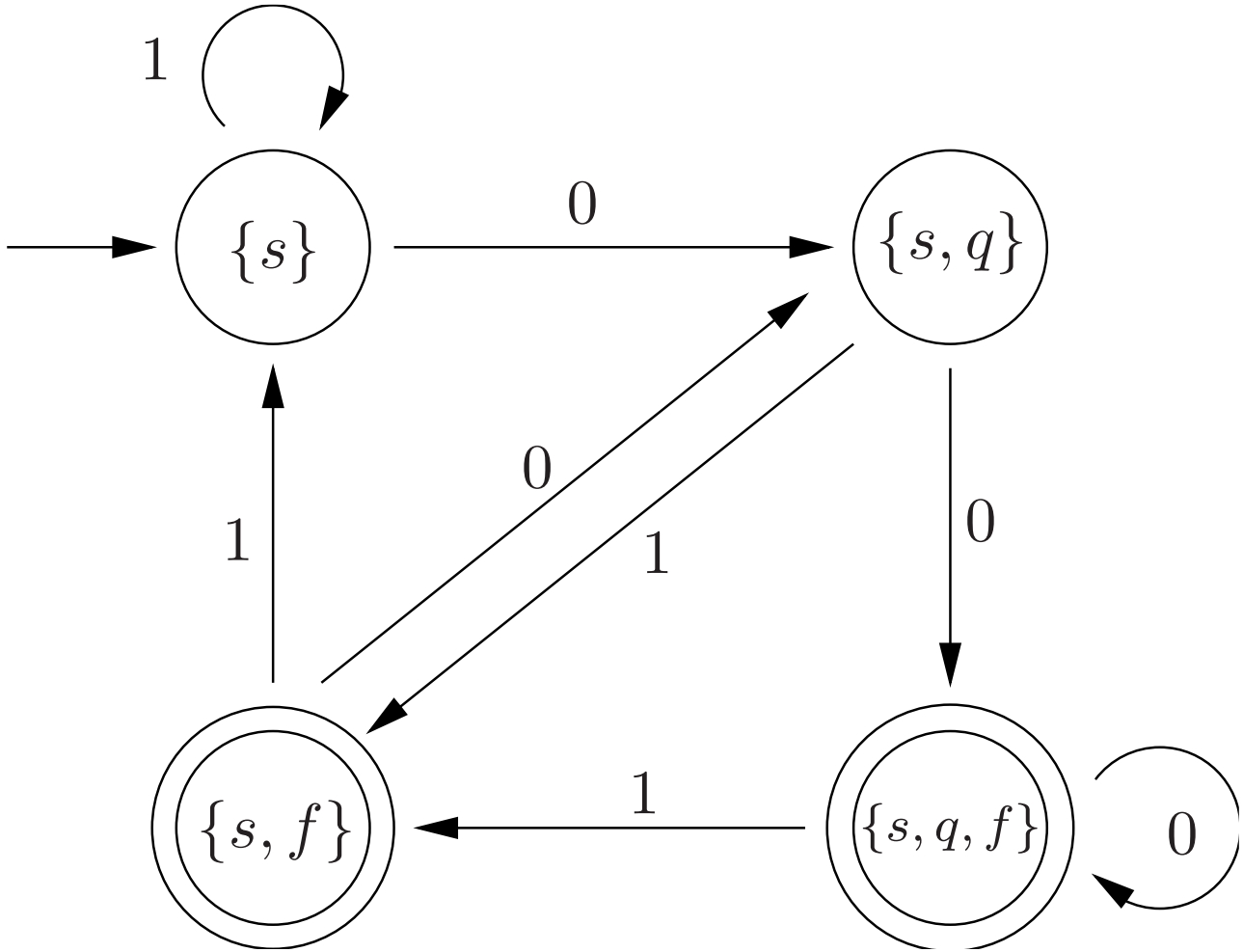
Was kann ein endlicher Automat?

- Gegeben ist eine Eingabe als endliche Folge von Eingabesymbolen. Der Automat entscheidet, ob die Eingabe zulässig ist oder nicht, indem er in einem Endzustand endet oder nicht.

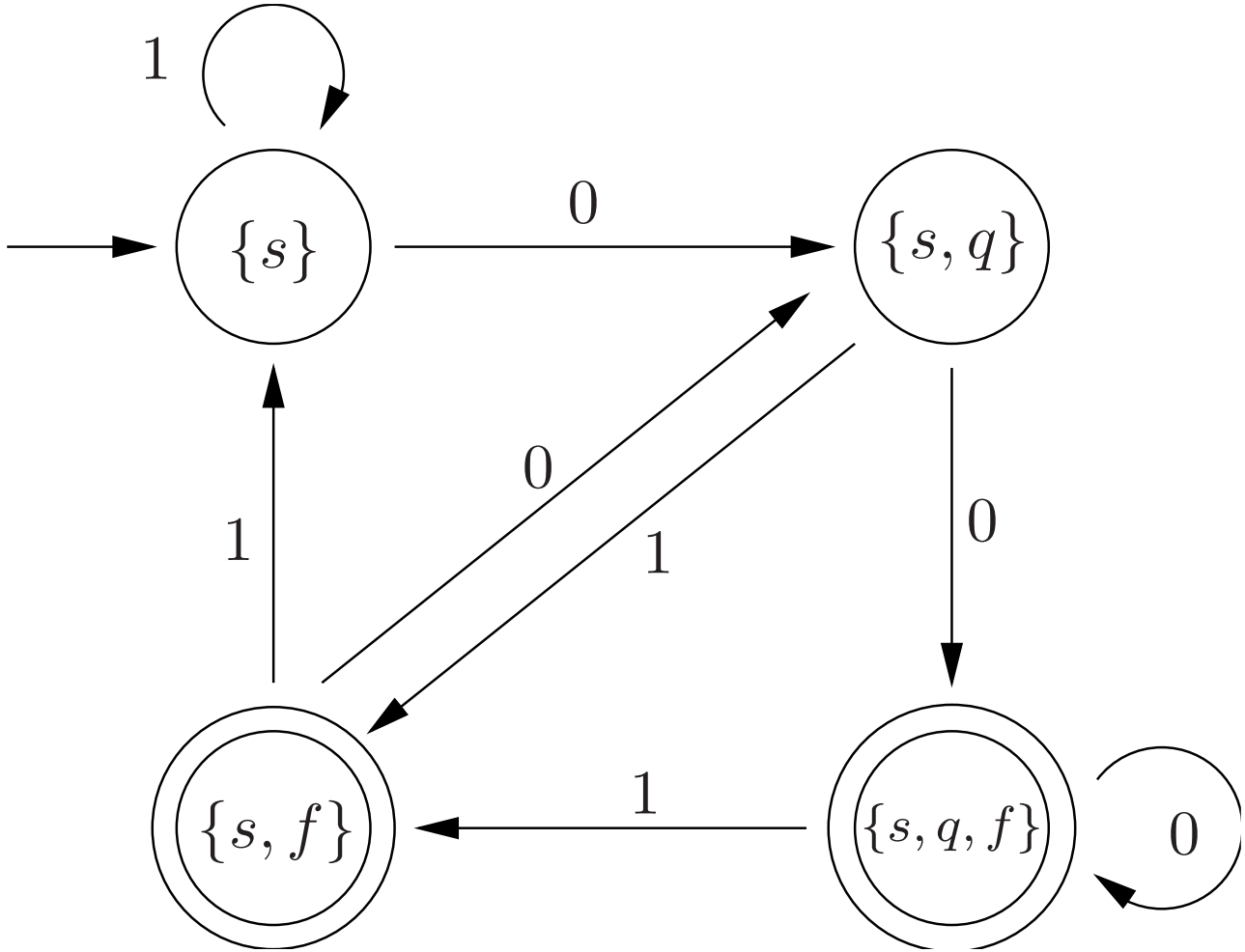
Endliche Automaten

- Ein endlicher Automat **erkennt** oder **akzeptiert** eine Sprache L, d.h. eine Menge von Wörtern über dem Alphabet des Automaten, wenn er nach Abarbeitung eines Wortes w genau dann in einem Endzustand ist, wenn das Wort w in der Sprache L ist ($w \in L$).
- Eine formale Sprache heißt *endliche Automatensprache*, wenn es einen endlichen Automaten gibt, der sie erkennt.

Endliche Automaten



Endliche Automaten



Sprache aller Wörter, deren vorletztes Symbol 0 ist: $L = (0 \cup 1)^* 0 (0 \cup 1)$

Kontextfreie Grammatiken

Eine kontextfreie Grammatik $G = (\Sigma, V, S, R)$ ist gegeben durch

- ein endliches Alphabet Σ (auch Terminalalphabet genannt)
- eine endlichen Menge V mit $V \cap \Sigma = \emptyset$ von Variablen (auch Nichtterminale genannt)
- ein Startsymbol $S \in V$
- eine endlichen Menge von Ableitungsregeln R, d.h. durch eine Menge von Tupeln $(l, r) \in V \times (\Sigma \cup V)^*$ (auch Produktionen genannt)

Wir schreiben Produktionen auch in der Form $l \to r$.

Kontextfreie Grammatiken

Gegeben ist eine Regel $l \to r$.

- Wenn in einem Wort w das Zeichen l vorhanden ist, darf l in w durch r ersetzt werden.
- Wir schreiben $w \to z$, wenn w durch Anwendung *einer* Ableitungsregel in z verwandelt wird.
- Wir schreiben $w \overset{*}{\to} z$, wenn w durch Anwendung *mehrerer* Ableitungsregel in z verwandelt wird.

Die von einer Grammatik $G = (\Sigma, V, S, R)$ erzeugte Sprache $L(G)$ ist die Menge aller Wörter $z \in \Sigma^*$ für die gilt $S \overset{*}{\to} z$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01 \mid 0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01 \mid 0S1 \to 01 \mid 0011 \mid 00S11 \\ &\to 01 \mid 0011 \mid 000111 \mid 000S111 \to \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01 \mid 0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01 \mid 0S1 \to 01 \mid 0011 \mid 00S11 \\ &\to 01 \mid 0011 \mid 000111 \mid 000S111 \to \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01 \mid 0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01 \mid 0S1 \to 01 \mid 0011 \mid 00S11 \\ &\to 01 \mid 0011 \mid 000111 \mid 000S111 \to \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01 \mid 0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01 \mid 0S1 \to 01 \mid 0011 \mid 00S11 \\ &\to 01 \mid 0011 \mid 000111 \mid 000S111 \to \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Welche Grammatik erzeugt die Sprache über dem Alphabet $\Sigma = \{0, 1\}$ deren vorletztes Zeichen 0 ist?

Beispiel Kontextfreie Grammatiken

Welche Grammatik erzeugt die Sprache über dem Alphabet $\Sigma = \{0, 1\}$ deren vorletztes Zeichen 0 ist?

Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S, A, B\} \\ R &:= \{S \to A0|A1, \quad A \to B0, \quad B \to \epsilon|B0|B1\} \end{aligned}$$

Ableitung:

$$S \to A0|A1 \to B00|B01 \to \ldots$$

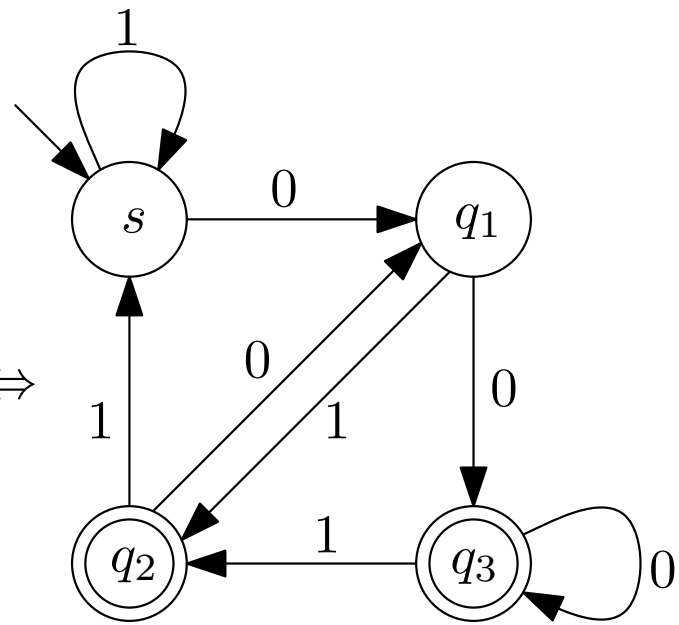
Kontextfreie Grammatiken

In TGI werden wir Grammatiken kennenlernen, deren Produktionen *mächtiger* sind, als die von kontextfreien Grammatiken.

Endliche Automaten und Reguläre Sprachen

- Beispiel: Sprache aller Wörter über $\{0, 1\}$, deren vorletztes Zeichen eine 0 ist:

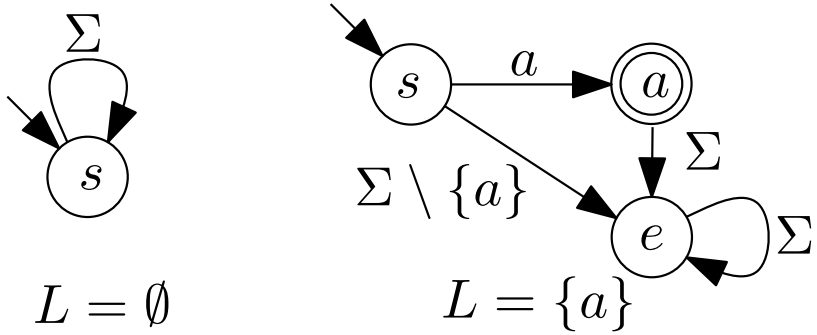
$$L := (0 \cup 1)^* 0 (0 \cup 1) \quad \Leftrightarrow$$



- Gibt es zu jeder regulären Sprache einen endlichen Automaten, der diese erkennt?
- Falls ja: Wie kann man diesen konstruieren?
- Welche Sprachen werden durch endliche Automaten erkannt?

Satz:
Jede reguläre Sprache wird von einem (deterministischen) endlichen Automaten (DEA) akzeptiert.

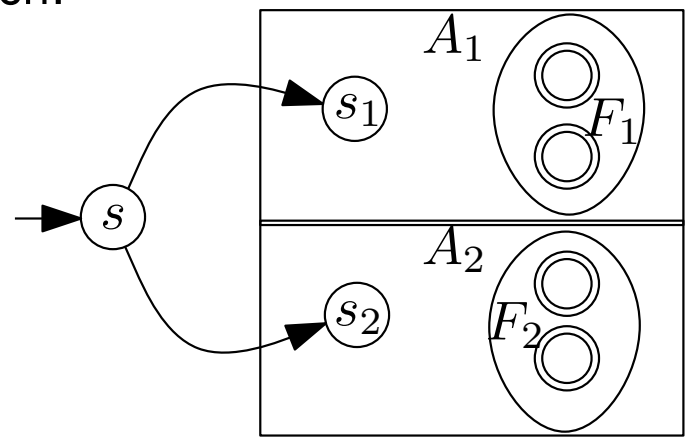
- $L := L(\alpha)$ reguläre Sprache über Σ, die durch α beschreibbar ist
- Induktion über n =Anzahl der $\cup$, $\cdot$ und *-Zeichen in α
- **Induktionsanfang** $n = 0$:



- **Induktionsannahme**: Für alle L, die durch reguläre Ausdrücke mit weniger als n Operationen beschreibbar sind, existiert akzeptierender DEA

Satz:
Jede reguläre Sprache wird von einem (deterministischen) endlichen Automaten (DEA) akzeptiert.

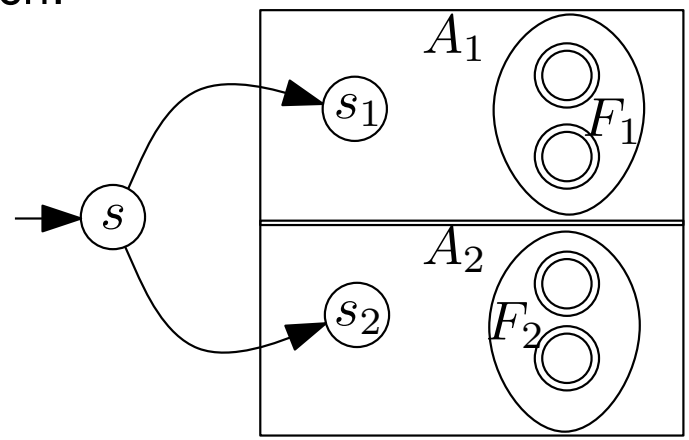
- **Induktionsschluss**:
  - $n > 0 \Rightarrow L = L_1 \cup L_2$ oder $L = L_1 \cdot L_2$ oder $L = L_1^*$
  - Fall 1: $L = L_1 \cup L_2$
  - Sei A_1 DEA, der L_1 akzeptiert und A_2 DEA, der L_2 akzeptiert
  - Idee: „Baue“ aus A_1 und A_2 Automaten, der L akzeptiert
  - 1. Versuch:



- Brauche Übergang ohne Lesen eines Zeichens mit Wahlmöglichkeit
- DEA’s erlauben das nicht
- Rest des Beweises: später!

Satz:
Jede reguläre Sprache wird von einem (deterministischen) endlichen Automaten (DEA) akzeptiert.

- **Induktionsschluss**:
  - $n > 0 \Rightarrow L = L_1 \cup L_2$ oder $L = L_1 \cdot L_2$ oder $L = L_1^*$
  - Fall 1: $L = L_1 \cup L_2$
  - Sei A_1 DEA, der L_1 akzeptiert und A_2 DEA, der L_2 akzeptiert
  - Idee: „Baue“ aus A_1 und A_2 Automaten, der L akzeptiert
  - 1. Versuch:



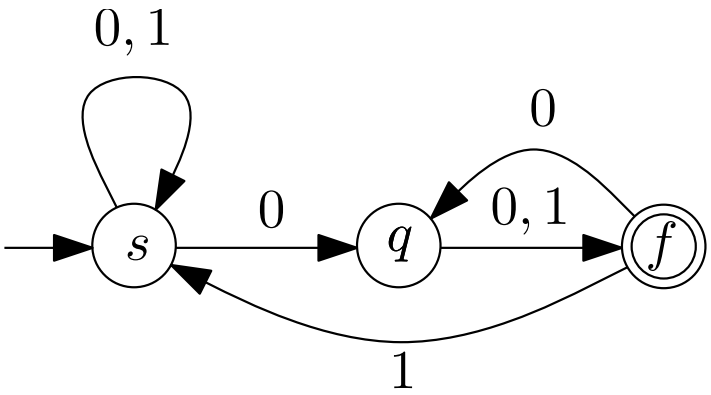
- Brauche Übergang ohne Lesen eines Zeichens mit Wahlmöglichkeit
- DEA’s erlauben das nicht
- Rest des Beweises: später!

Nichtdeterministische endliche Automaten

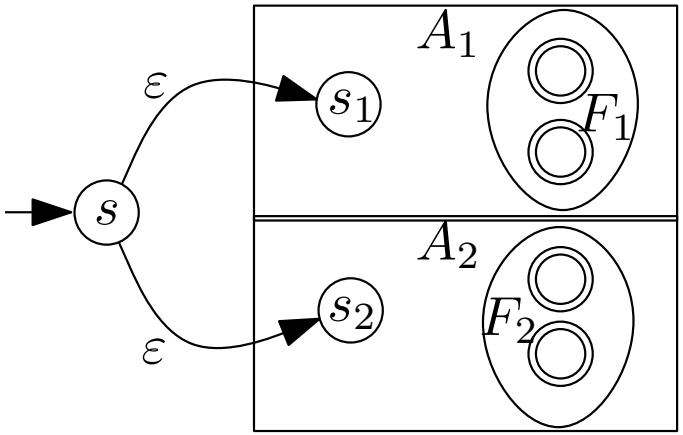
- Ein **nichtdeterministischer endlicher Automat** (NEA) besteht aus:
  - Q, einer endlichen Zustandsmenge;
  - Σ, einem endlichen Alphabet;
  - δ, einer Übergangsfunktion $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$
  - s, einem Startzustand;
  - F, einer Menge von Endzuständen;
- Interpretation von δ:
  - Bei Abarbeitung eines Symbols kann der Automat sich –nichtdeterministisch– aussuchen, in welchen Zustand einer Teilmenge aus Q er übergeht
  - Auch möglich: Springe spontan ohne Lesen eines Zeichens aus Σ in neuen Zustand (ε-Übergang)
- Ein nichtdeterministischer endlicher Automat **akzeptiert** eine Wort $w \in \Sigma^*$, wenn es eine Folge von Übergängen gibt, so dass er bei Eingabe von w in einen Endzustand gelangt, d.h. bei Eingabe von w ein Endzustand *erreichbar* ist.

Beispiele für NEA's

Beispiel 1:

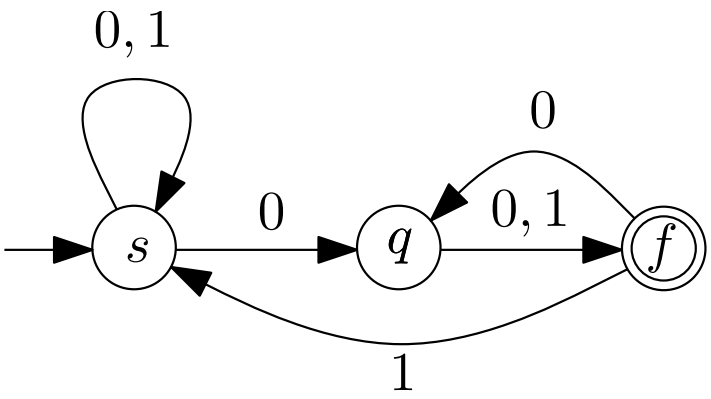


Beispiel 2:



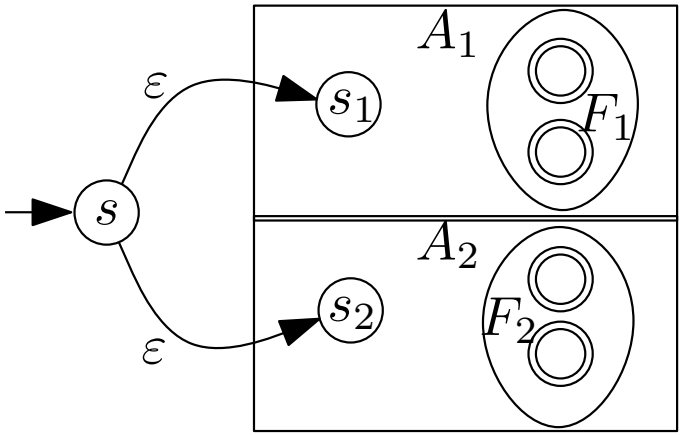
Beispiele für NEA's

Beispiel 1:



- $\delta(s, 0) = \{s, q\}$
- $w = 10$ wird nicht akzeptiert
- $w = 01$ wird akzeptiert
- Sprache aller Wörter über $\{0, 1\}$, deren vorletztes Zeichen 0 ist

Beispiel 2:



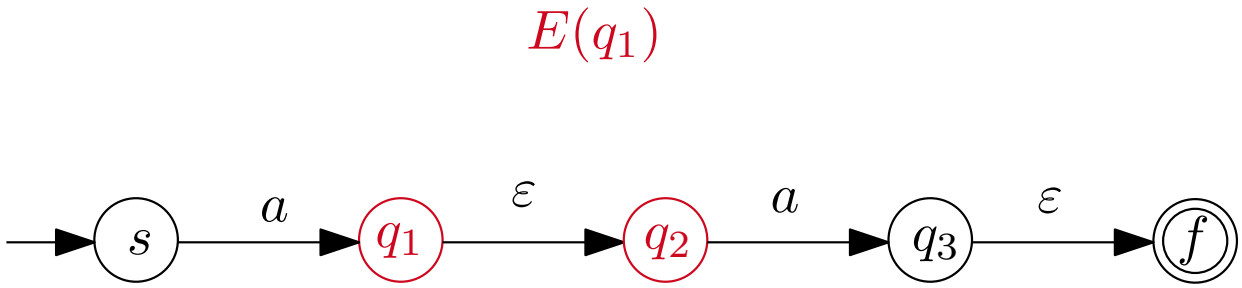
Erweiterung von δ

Definition:
Für einen Zustand $q \in Q$ ist der **ε–Abschluss** $E(q)$ wie folgt definiert:

$$E(q) := \{p \in Q \mid p \text{ ist von } q \text{ durch } \varepsilon\text{–Übergänge erreichbar}\}$$

Beachte, dass gilt:

- $E(q) \subseteq Q, \quad E(q) \in 2^Q$
- $q \in E(q)$, d.h. die Folge von ε–Übergängen darf auch leer sein



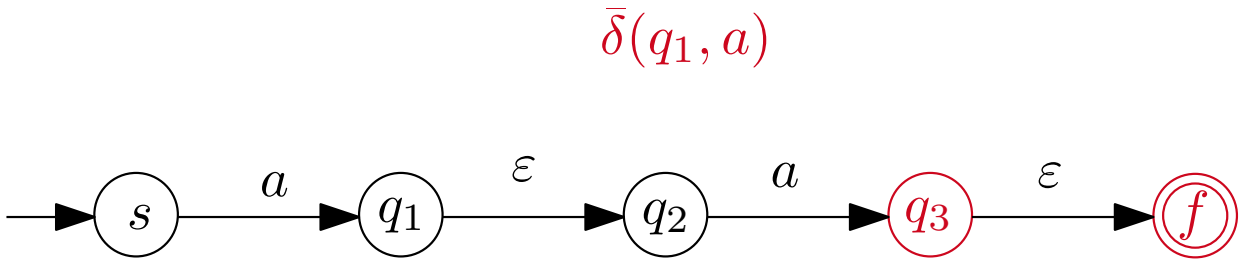
Erweiterung von δ

Idee: Berücksichtige ε-Übergänge bei der Übergangsfunktion

- Beim Lesen eines einzelnen Zeichens

$$\bar{\delta}\colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$$

$$\bar{\delta}(q,a) = \begin{cases} E(q) & \text{falls } a = \varepsilon \\ \bigcup_{p \in E(q)} \left(\bigcup_{r \in \delta(p,a)} E(r) \right) & \text{für } a \in \Sigma \end{cases}$$



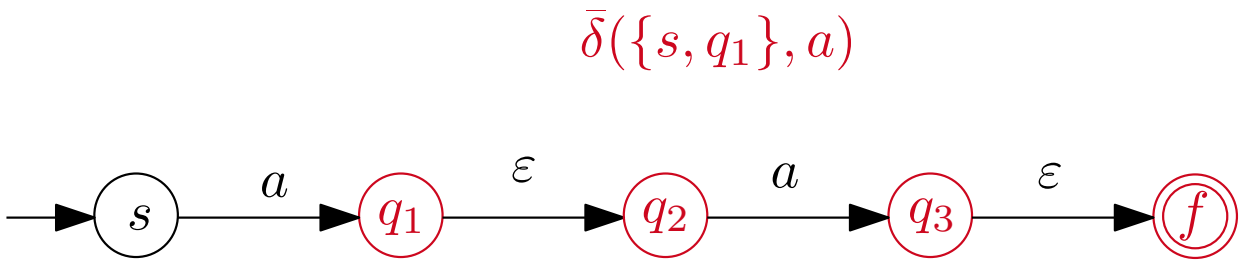
Erweiterung von δ

Idee: Berücksichtige ε-Übergänge bei der Übergangsfunktion

- Erweiterung auf Mengen von Zuständen :

$$\bar{\delta}\colon 2^Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$$

$$\bar{\delta}(P, a) = \begin{cases} \bigcup_{p \in P} E(p) & \text{falls } a = \varepsilon \\ \bigcup_{p \in P} \bar{\delta}(p, a) & \text{für } a \in \Sigma \end{cases}$$



Erweiterung von δ

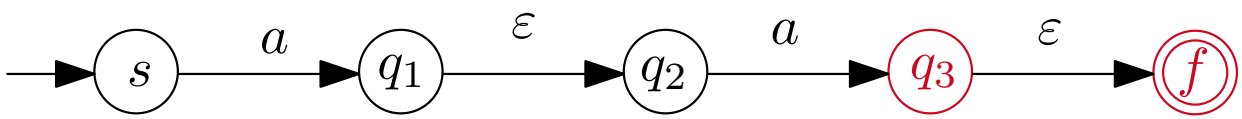
Idee: Berücksichtige ε-Übergänge bei der Übergangsfunktion

- Induktive Erweiterung auf Wörter:

$$\bar{\delta}\colon Q \times \Sigma^* \to 2^Q$$

$$\bar{\delta}(q, w) = \begin{cases} E(q) & \text{falls } w = \varepsilon \\ \bar{\delta}(q, w) & \text{falls } w = a \in \Sigma \\ \bigcup_{p \in \bar{\delta}(q,v)} \bar{\delta}(p, a) & \text{falls } w = va,\ a \in \Sigma,\ |v| > 0 \end{cases}$$

$$\bar{\delta}(s, w = aa)$$



- Es gilt: $w \in L(\mathcal{A}) \Leftrightarrow \bar{\delta}(s, w) \cap F \neq \emptyset$!

Erweiterung von δ

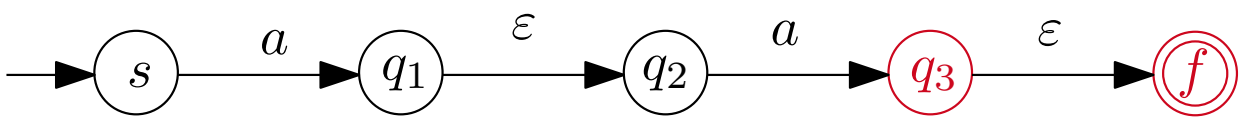
Idee: Berücksichtige ε-Übergänge bei der Übergangsfunktion

- Analog für Mengen von Zuständen:

$$\bar{\delta}\colon 2^Q \times \Sigma^* \to 2^Q$$

$$\bar{\delta}(P, w) = \bigcup_{p \in P} \bar{\delta}(p, w)$$

$$\bar{\delta}(\{s, q_1\}, w = aa)$$



Äquivalenz von NEA's und DEA's

Definition:
Zwei endliche Automaten, die dieselbe Sprache akzeptieren, heißen *äquivalent*.

Satz:
Zu jedem nichtdeterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

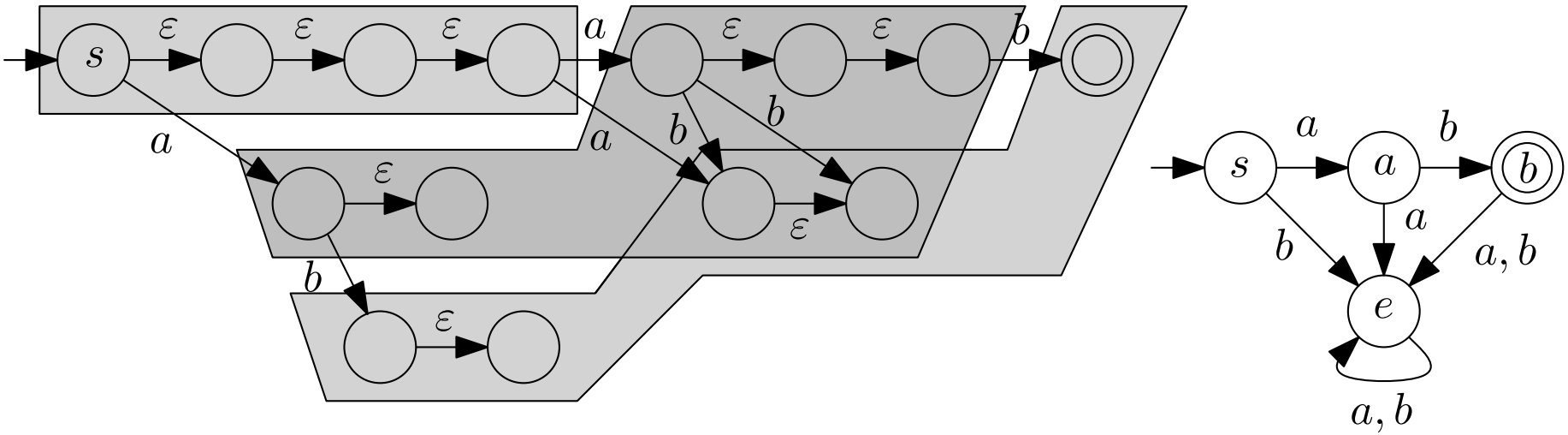
Satz:
Zu jedem nichtdeterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

Beweisidee:

- Sei $\mathcal{A}$ NEA
- Falls $\mathcal{A}$ w akzeptiert, so gibt es eine Abarbeitung, die in einem Endzustand endet
- Falls ein Zustand q bei der Eingabe von w erreichbar ist, so sind das auch alle Zustände in $E(q)$
- „Simuliere“ $\mathcal{A}$ durch einen DEA
- Zustände des DEA sind Mengen von Zuständen von $\mathcal{A}$
- Endzustände des DEA sind alle Mengen, die Endzustände von $\mathcal{A}$ enthalten

Satz:
Zu jedem nichtdeterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

Beispiel: Mögliche Abarbeitungen von $w = ab$ in einem NEA mit $ab \in L, a \notin L$ und äquivalenter DEA



Satz:
Zu jedem nichtdeterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

Potenzmengenkonstruktion: Gegeben sei ein NEA $\mathcal{A} := (Q, \Sigma, \delta, s, F)$. Wir konstruieren daraus einen DEA $\widetilde{\mathcal{A}} := (\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{s}, \widetilde{F})$:

- $\widetilde{Q} = 2^Q$, d.h. die Zustände des DEA sind Mengen von Zuständen des NEA.
- $\widetilde{\delta}\colon \widetilde{Q} \times \Sigma \to \widetilde{Q}$ mit $\widetilde{\delta}(\widetilde{q}, a) = \bar{\delta}(\widetilde{q}, a)$ für $a \in \Sigma$. Es ist also $\widetilde{q} \subseteq Q$ und jeder Zustand wird mit seinem ε–Abschluss im NEA identifiziert.
- $\widetilde{s} := E(s)$
- $\widetilde{F} := \{\widetilde{q} \in \widetilde{Q} \mid \widetilde{q} \cap F \neq \emptyset\}$

Satz:
Zu jedem nichtdeterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

- Zeige per Induktion über $n = |w|$, dass für alle $w \in \Sigma^*$ gilt:

$$\widetilde{\delta}(\widetilde{s}, w) = \bar{\delta}(s, w)$$

- Dann gilt:

$$\begin{aligned} w \in L(\widetilde{\mathcal{A}}) &\Leftrightarrow \widetilde{\delta}(\widetilde{s}, w) \in \widetilde{F} \Leftrightarrow \widetilde{\delta}(\widetilde{s}, w) \cap F \neq \emptyset \\ &\Leftrightarrow \bar{\delta}(s, w) \cap F \neq \emptyset \Leftrightarrow w \in L(\mathcal{A}) \end{aligned}$$

- **Induktionsanfang** $n = 0$, d.h. $w = \varepsilon$:

$$\widetilde{\delta}(\widetilde{s}, \varepsilon) = \bar{\delta}(E(s), \varepsilon) = \cup_{p \in E(s)} E(p) = E(s) = \bar{\delta}(s, \varepsilon)$$

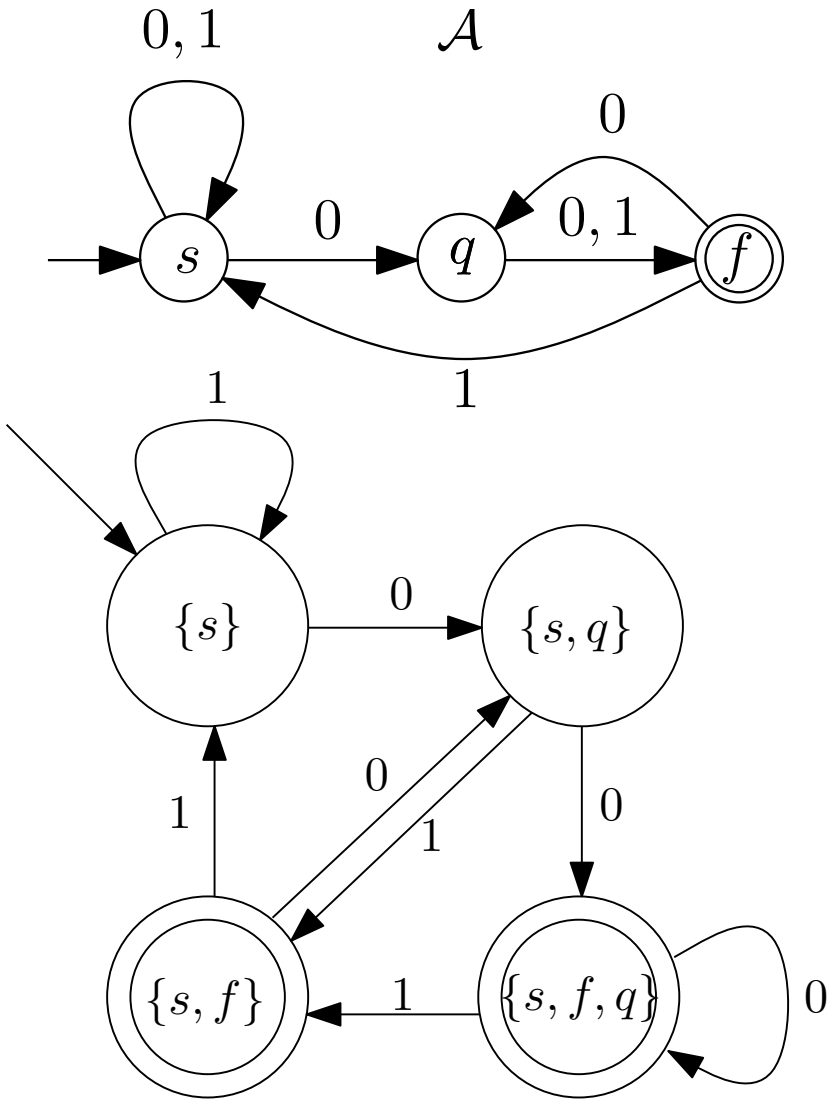
Satz:
Zu jedem nichtdeterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

- **Induktionsannahme**: Für alle Wörter w' mit $|w'| \leq n$ gilt:
  $\widetilde{\delta}(\widetilde{s}, w') = \bar{\delta}(s, w')$
- **Induktionsschluss** $|w| = n+1$:
  Sei $w = w'a$ mit $|w'| = n$ und $a \in \Sigma$. Dann gilt:

$$\begin{aligned}
\widetilde{\delta}(\widetilde{s}, w) &= \widetilde{\delta}(\widetilde{\delta}(\widetilde{s}, w'), a) \\
&\underset{(\mathrm{IV})}{=} \widetilde{\delta}(\bar{\delta}(s, w'), a) \\
&\underset{\text{Def } \widetilde{\delta}}{=} \bar{\delta}(\bar{\delta}(s, w'), a) = \bigcup_{p \in \bar{\delta}(s,w')} \bar{\delta}(p, a) = \bar{\delta}(s, w)
\end{aligned}$$

Beispiel Potenzmengenkonstruktion:

- $L = (0 \cup 1)^* 0 (0 \cup 1)$
- NEA $\mathcal{A}$ akzeptiert L
- Konstruiere DEA für L:
  - $\widetilde{s} = E(s) = \{s\}$
  - $\widetilde{\delta}(\{s\}, 0) = \{s, q\}$
  - $\widetilde{\delta}(\{s\}, 1) = \{s\}$
  - $\widetilde{\delta}(\{s, q\}, 0) = \{s, q, f\}$
  - $\widetilde{\delta}(\{s, q\}, 1) = \{s, f\}$
  - $\widetilde{\delta}(\{s, q, f\}, 0) = \{s, q, f\}$
  - $\widetilde{\delta}(\{s, q, f\}, 1) = \{s, f\}$
  - $\widetilde{\delta}(\{s, f\}, 0) = \{s, q\}$
  - $\widetilde{\delta}(\{s, f\}, 1) = \{s\}$
  - Alle anderen Zustände können gestrichen werden



Satz:
Jede reguläre Sprache wird von einem (deterministischen) endlichen Automaten (DEA) akzeptiert.

Beweis (Fortsetzung):

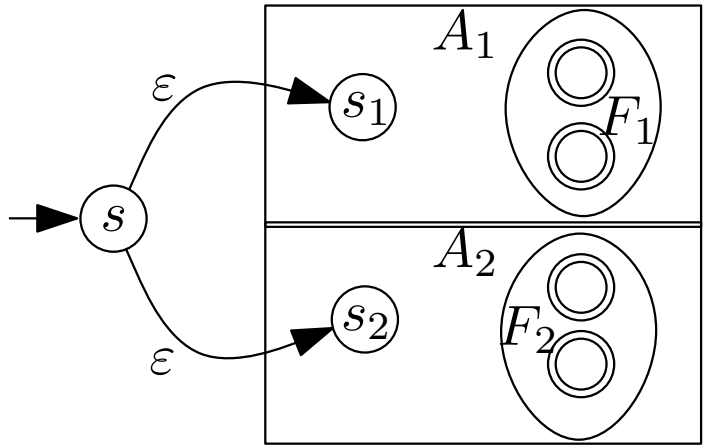
- Erinnerung:
  - $L := L(\alpha)$ reguläre Sprache über Σ, die durch α beschreibbar ist
  - Beweis per Induktion über $n =$Anzahl der $\cup$, $\cdot$ und *-Zeichen in α
  - Induktionsanfang bereits gezeigt
  - noch zu zeigen: Induktionsschritt für reguläre Sprachen $L = L_1 \cup L_2$, $L = L_1 \cdot L_2$ und $L = L_1^*$
- Seien L_1 und L_2 reguläre Sprachen, die von $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ erkannt werden
- Baue aus $\mathcal{A}_1$ und $\mathcal{A}_2$ NEA's, die $L_1 \cup L_2$, $L_1 \cdot L_2$ und L_1^* erkennen
- Aus diesen können äquivalente DEA's konstruiert werden

Satz:
Jede reguläre Sprache wird von einem (deterministischen) endlichen Automaten (DEA) akzeptiert.

Beweis (Fortsetzung):

- Fall 1: $L = L_1 \cup L_2$
- NEA $\mathcal{A} := (Q, \Sigma, \delta, s, F)$, der $L_1 \cup L_2$ erkennt:
  - $Q := Q_1 \cup Q_2 \cup \{s\}$ $(s \notin Q_i)$
  - $F = F_1 \cup F_2$
  - δ :

$$\delta(q,a) := \begin{cases} \{\delta_i(q,a)\} & , q \in Q_i,\ a \in \Sigma \\ \emptyset & , q \in Q \backslash \{s\},\ a = \varepsilon \\ \{s_1, s_2\} & , q = s,\ a = \varepsilon \\ \emptyset & , q = s,\ a \neq \varepsilon \end{cases}$$

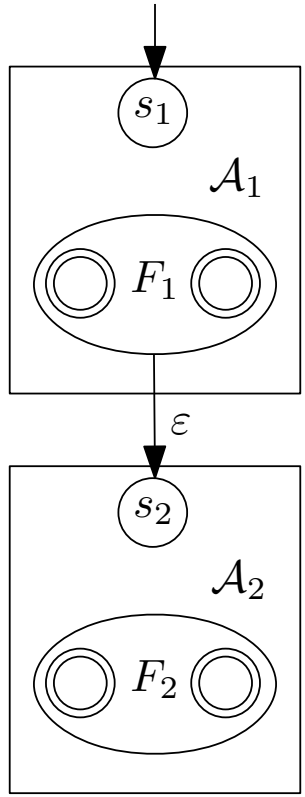


Satz:
Jede reguläre Sprache wird von einem (deterministischen) endlichen Automaten (DEA) akzeptiert.

Beweis (Fortsetzung):

- Fall 2: $L = L_1 \cdot L_2$
- NEA $\mathcal{A} := (Q, \Sigma, \delta, s, F)$, der $L_1 \cdot L_2$ erkennt:
  - $Q := Q_1 \cup Q_2$, $s := s_1$, $F := F_2$
  - $s := s_1$
  - $F := F_2$
  - δ:

$$\delta(q,a) := \begin{cases} \{\delta_i(q,a)\} & , q \in Q_i,\ a \in \Sigma \\ \emptyset & , q \in Q \backslash F_1,\ a = \varepsilon \\ \{s_2\} & , q \in F_1,\ a = \varepsilon \end{cases}$$

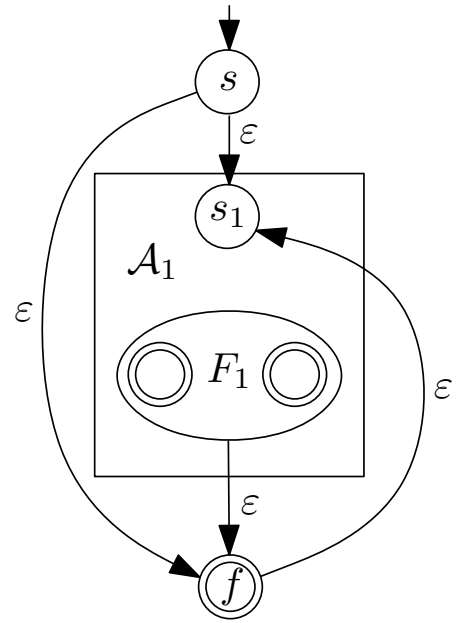


Satz:
Jede reguläre Sprache wird von einem (deterministischen) endlichen Automaten (DEA) akzeptiert.

Beweis (Fortsetzung):

- Fall 3: $L = L_1^*$
- NEA $\mathcal{A} := (Q, \Sigma, \delta, s, F)$, der L_1^* erkennt:
  - $Q := Q_1 \cup \{s, f\}$
  - s: neu
  - $F := \{f\}$ neuer Zustand
  - δ:

$$\delta(q,a) := \begin{cases} \{\delta_1(q,a)\} & , q \in Q_1,\ a \in \Sigma \\ \emptyset & , q \in Q_1 \setminus F_1,\ a = \varepsilon \\ \{f\} & , q \in F_1 \cup \{s\},\ a = \varepsilon \\ \emptyset & , q \in \{s, f\},\ a \neq \varepsilon \\ \{s_1\} & , q \in \{s, f\},\ a = \varepsilon \end{cases}$$



Entfernen von ϵ-Übergängen

Satz:
Zu jedem NEA $\mathcal{A}$ mit ε-Übergängen gibt es einen NEA $\widetilde{\mathcal{A}}$ ohne ε-Übergänge, der dieselbe Sprache akzeptiert und nicht mehr Zustände hat.

Entfernen von ϵ-Übergängen

Satz:
Zu jedem NEA $\mathcal{A}$ mit ε-Übergängen gibt es einen NEA $\widetilde{\mathcal{A}}$ ohne ε-Übergänge, der dieselbe Sprache akzeptiert und nicht mehr Zustände hat.

Beweis: Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ ein NEA mit ε–Übergängen.
Wir konstruieren $\widetilde{\mathcal{A}} := (\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{s}, \widetilde{F})$ wie folgt:

- $\widetilde{Q} := (Q \backslash F) \cup \widetilde{F}$
- $\widetilde{s} := s$
- $$\widetilde{\delta}(q, a) = \begin{cases} \{q\} & \text{falls } a = \varepsilon \\ \delta(E(q), a) & \text{sonst} \end{cases}$$
- $\widetilde{F} := \{q \in Q \mid E(q) \cap F \neq \emptyset\}$

Damit akzeptiert $\widetilde{A}$ dieselbe Sprache wie $\mathcal{A}$, und $|\widetilde{Q}| \leq |Q|$.

EA $\rightarrow$ Regularität

Satz:
Jede Sprache, die von einem endlichen Automaten erkannt wird, ist regulär.

Beweis: EA $\rightarrow$ Regularität

- Sei DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ gegeben.
- Es ist zu zeigen, dass $L(\mathcal{A})$ regulär ist.

Es gilt:

$L = \{w \in \Sigma^* \mid \mathcal{A}$ endet nach Abarbeitung von w in einem Zustand aus $F\}$

- Die Abarbeitung eines Wortes $w = a_1 \ldots a_k$ bewirkt das Durchlaufen einer Folge von Zuständen $s, q_1, \ldots, q_k$, wobei nicht notwendig $q_i \neq q_j$ für $i \neq j$ gilt.
- Wir suchen die Wörter, so dass der letzte Zustand in F ist.
- Betrachte für jeden Zustand $f \in F$ getrennt die Wörter, deren Abarbeitung in f endet.

Beweis: EA $\rightarrow$ Regularität

- Sei DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ gegeben.
- Es ist zu zeigen, dass $L(\mathcal{A})$ regulär ist.

Es gilt:

$$L = \{w \in \Sigma^* \mid \mathcal{A} \text{ endet nach Abarbeitung von } w \text{ in einem Zustand aus } F\}$$

Zu $f \in F$ definiere:

$$\begin{aligned} L_f &:= \{w \in \Sigma^* \mid \mathcal{A} \text{ endet nach Abarbeitung von } w \text{ in } f\} \\ &= \{w \in \Sigma^* \mid w \text{ überführt } s \text{ in } f \text{ (im Automaten } \mathcal{A})\} \end{aligned}$$

- Damit ist $L = \bigcup_{f \in F} L_f$.
- Wenn wir zeigen können, dass für alle $f \in F$ die Sprache L_f regulär ist, so ist auch L regulär.

Beweis: EA $\rightarrow$ Regularität

$$L_f := \{w \in \Sigma^* \mid w \text{ überführt } s \text{ in } f \text{ (im Automaten } \mathcal{A})\}$$

Ab jetzt sei $Q = \{q_1, \ldots, q_n\}$.
Wir definieren zu

$$q_r, q_t \in Q\text{: } L_{q_r,q_t} := \{w \in \Sigma^* \mid w \text{ überführt } q_r \text{ in } q_t\} \ .$$

Insbesondere gilt also: $L_f = L_{s,f}$. Unterteile L_{q_r,q_t}:

$$L_{q_r,i,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} \text{Abarbeitung von } w \text{ aus } q_r \text{ nach } q_t \text{ hat nur} \\ \text{Zwischenzustände } \{q_1, \ldots, q_i\} \end{array} \right\}$$

(also w bewirkt: $q_r \rightarrow \underbrace{\ldots\ldots\ldots}_{\in\{q_1,\ldots,q_i\}} \rightarrow q_t$.)

Damit gilt $L_{q_r,q_t} = L_{q_r,n,q_t}$.

Beweis: EA $\rightarrow$ Regularität

$$L_{q_r,i,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} \text{Abarbeitung von } w \text{ aus } q_r \text{ nach } q_t \text{ hat nur} \\ \text{Zwischenzustände } \{q_1, \ldots, q_i\} \end{array} \right\}$$

Wir zeigen, dass L_{q_r,i,q_t} für $q_r, q_t \in Q$ und $1 \leq i \leq n$ regulär sind:

- Zunächst betrachten wir direkte Überführungen, also $i = 0$:

$$L_{q_r,0,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} \text{Abarbeitung von } w \text{ führt von } q_r \text{ nach } q_t \\ \text{ohne Zwischenzustand} \end{array} \right\}$$

  Falls $r = t$ und somit $q_r = q_t$ ist, ist

$$L_{q_r,0,q_t} = \{a \in \Sigma \mid \delta(q_t, a) = q_t\} \cup \{\varepsilon\} .$$

  Andernfalls betrachten wir alle w mit $q_r \xrightarrow{w} q_t$, ohne Zwischenzustände, also

$$L_{q_r,0,q_t} = \{a \in \Sigma \mid \delta(q_r, a) = q_t\} .$$

  Diese Sprachen sind jeweils regulär.

Beweis: EA $\rightarrow$ Regularität

- Betrachte nun $i = 1$:

$$L_{q_r,1,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ überführt } q_r \text{ in } q_t \text{ entweder direkt oder} \\ \text{unter Benutzung nur von } q_1 \end{array} \right\}$$

  Es gilt dann:

$$L_{q_r,1,q_t} = L_{q_r,0,q_t} \cup \left(L_{q_r,0,q_1} \cdot L^*_{q_1,0,q_1} \cdot L_{q_1,0,q_t} \right)$$

  Also ist $L_{q_r,1,q_t}$ auch wieder regulär.

- Es gilt allgemein:

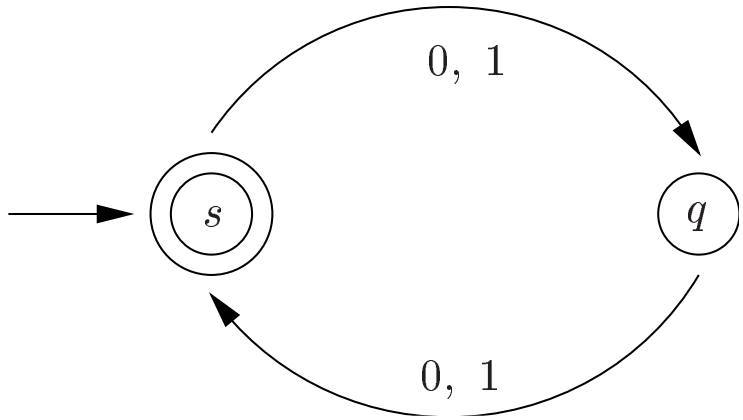
$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} \left(L_{q_{i+1},i,q_{i+1}} \right)^* L_{q_{i+1},i,q_t} \right)$$

Beweis: EA $\rightarrow$ Regularität

- Es wurden für $L_{\cdot,i+1,\cdot}$ nur die Sprachen $L_{\cdot,i,\cdot}$ und $\cup, \cdot, *$ verwendet.
- Damit ist gezeigt (per Induktion), dass $L_{\cdot,i+1,\cdot}$ regulär ist für beliebiges i $(1 \leq i+1 \leq n)$ und alle Zustandspaare aus Q^2.
- Damit ist gezeigt, dass insbesondere $L_f = L_{s,n,f}$ regulär ist für jedes $f \in F$.

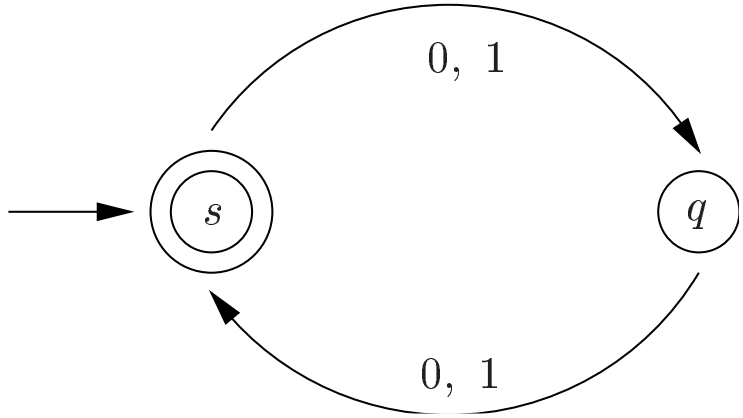
Beispiel

Sei $(Q, \Sigma, \delta, s, F)$ mit $Q := \{q_1 := s, q_2 := q\}, \Sigma := \{0, 1\}, F := \{s\}$



Gesucht: $L(Q, \Sigma, \delta, s, F)$. Es gilt $L = L_{q_1,2,q_1}$.

Beispiel



Gesucht: $L(Q, \Sigma, \delta, s, F)$. Es gilt $L = L_{q_1,2,q_1}$.

Dann ist

$$L_{q_i,0,q_i} = \varepsilon$$

$$L_{q_i,0,q_j} = (0 \cup 1) \text{ für } i, j \in \{1, 2\}, i \neq j$$

$$L_{q_1,1,q_1} = L_{q_1,0,q_1} \cup L_{q_1,0,q_1} \left(L_{q_1,0,q_1}\right)^* L_{q_1,0,q_1} = \varepsilon$$

$$L_{q_1,1,q_2} = L_{q_1,0,q_2} \cup L_{q_1,0,q_1} \left(L_{q_1,0,q_1}\right)^* L_{q_1,0,q_2} = (0 \cup 1) \cup \varepsilon\varepsilon^*(0 \cup 1) = 0 \cup 1$$

$$L_{q_2,1,q_1} = (0 \cup 1) \cup (0 \cup 1)\varepsilon^*\varepsilon = 0 \cup 1$$

$$L_{q_2,1,q_2} = \varepsilon \cup (0 \cup 1)\varepsilon^*(0 \cup 1) = \varepsilon \cup (0 \cup 1)(0 \cup 1)$$

$$L = L_{q_1,2,q_1} = L_{q_1,1,q_1} \cup \left(L_{q_1,1,q_2}\left(L_{q_2,1,q_2}\right)^* L_{q_2,1,q_1}\right)$$

$$= \varepsilon \cup (0 \cup 1)\left((0 \cup 1)(0 \cup 1)\right)^*(0 \cup 1) = \left((0 \cup 1)(0 \cup 1)\right)^*$$

Satz von Kleene

- Wir haben gezeigt, dass die von endlichen Automaten akzeptierten Sprachen genau die regulären Sprachen sind.
- Dies wird auch als der **Satz von Kleene** bezeichnet.

Satz (Satz von Kleene):
Die von endlichen Automaten akzeptierten Sprachen sind genau die regulären Sprachen.

Frage: Was können endliche Automaten nicht?

Frage: Was können endliche Automaten nicht?

Beispiel:
Die Sprache L der korrekten Klammerausdrücke über $\Sigma = \{(,)\}$.
Etwa

$$\Big(()()\Big), \Big(()()\Big(()\Big)\Big) \in L \qquad\qquad ((()), (()))()(\notin L$$

Frage: Was können endliche Automaten nicht?

Beispiel:
Die Sprache L der korrekten Klammerausdrücke über $\Sigma = \{(,)\}$.
Etwa

$$\Big(()()\Big), \Big(()()\Big(()\Big)\Big) \in L \qquad\qquad ((()), (()))()(\notin L$$

- Die Klammerung ist genau dann korrekt, wenn w gleich viele öffnende wie schließende Klammern enthält, und wenn man w von links nach rechts liest, so gibt es nie mehr „)“ als „(“ bis dahin.
- Ein Automat, der L erkennen kann, muss in der Lage sein, sich für ein beliebiges Wort $w \in L$ die Anzahl von (gegenüber) zu merken.
- Dies kann aber beliebig groß werden, und der Automat müsste über unendliche viele Zustände verfügen.
- Die Sprache der Klammerausdrücke ist also zwar simpel, aber wohl nicht regulär.

Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Beweis:

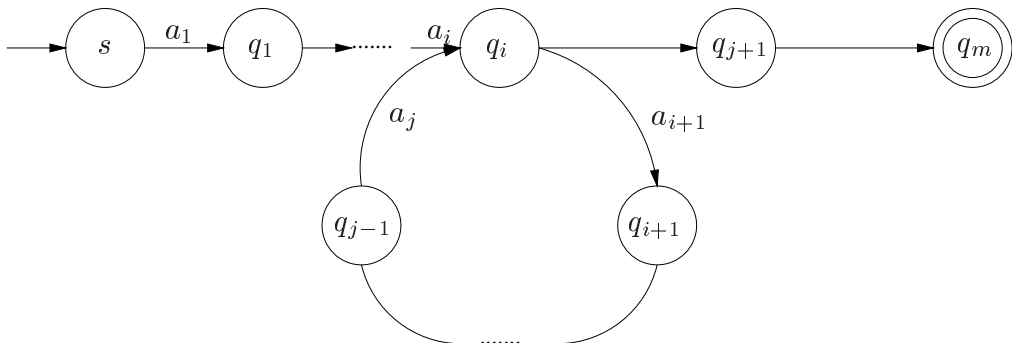
- Sei L eine reguläre Sprache.
- Dann existiert ein endlicher Automat, der L akzeptiert.
- Sei Q dessen Zustandsmenge und $n := |Q|$.
- Sei $w \in L$ mit $|w| > n$, etwa $w = a_1 \ldots a_n \ldots a_m$ mit $m > n$.

Pumping-Lemma für reguläre Sprachen

Beweis:

- Sei L eine reguläre Sprache.
- Dann existiert ein endlicher Automat, der L akzeptiert.
- Sei Q dessen Zustandsmenge und $n := |Q|$.
- Sei $w \in L$ mit $|w| > n$, etwa $w = a_1 \dots a_n \dots a_m$ mit $m > n$.

Bei der Abarbeitung von w werden dann die Zustände $q_0, \dots, q_m$ durchlaufen mit $q_m \in F$.
Dann gibt es i, j mit $0 \leq i, j \leq n$ und $i \neq j$, so dass $q_i = q_j$. Œ gelte $i < j$.

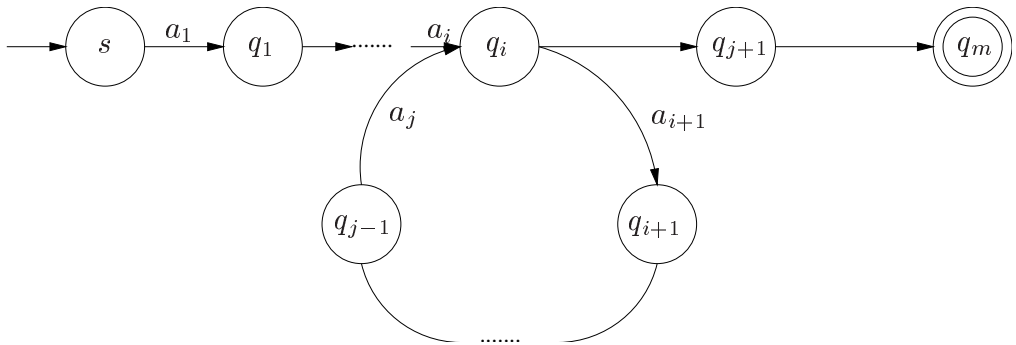


Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.



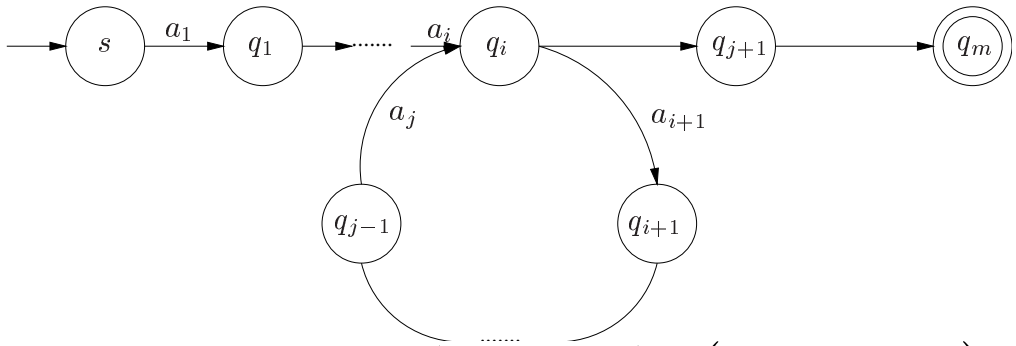
Dann kann der Zykel $q_i, q_{i+1}, \ldots, q_j = q_i$ auch gar nicht oder beliebig oft bei der Abarbeitung eines Wortes aus L durchlaufen werden so dass der Zustand $q_m \in F$ erreicht wird.

Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.



Also gibt es eine Zerlegung $w = \underbrace{(a_1 \ldots a_i)}_{u} \cdot \underbrace{(a_{i+1} \ldots a_j)}_{v} \cdot \underbrace{(a_{j+1} \ldots a_m)}_{x}$

mit $|uv| \leq n$ und $v \neq \varepsilon$, so dass auch $uv^i x \in L$ für alle $i \in \mathbb{N}_0$.

Bemerkung

- Das Pumping-Lemma liefert nur eine notwendige, aber nicht hinreichende Bedingung für die Regularität von Sprachen.

Beispiel (1) zum PL

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Gegeben sei

- $\Sigma = \{0, 1\}$
- $L = \{w \in \Sigma^* \mid w \text{ enthält 10 nicht als Teilwort } \} = 0^*1^*$

Betrachte

- $n = 1, \quad w = uvx, \quad u = \varepsilon$

Dann

- entspricht v also dem ersten Buchstaben von w
- kann $uv^i x$ auch 10 nicht als Teilwort besitzen.

Beispiel (2) zum PL

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Sei $\Sigma = \{0, 1\}$ und $L = \{0^i 1^i \mid i \geq 0\}$. Wir zeigen: L ist nicht regulär.

- Für ein n wähle $w = 0^n 1^n$
- Dann ist $|w| > n$
- Für jede Darstellung $w = uvx$ mit $|uv| \leq n$ und $v \neq \varepsilon$ ist aber

$$uv^0 x = 0^l 1^n \notin L \qquad (l < n)$$

Beispiel (3) zum PL

Sei $\Sigma = \{0, 1\}$ und

$$L = \left\{ w \in \Sigma^* \;\middle|\; w = 1^k \ (k > 0) \text{ oder } w = 0^j 1^{k^2} \ (j \geq 1,\ k \geq 0) \right\}.$$

Dann erfüllt L die Darstellung des PL:

- Sei $n = 1$ und $w \in L$ mit $|w| > 1$.
- w habe eine Darstellung $w = uvx$ mit $|uv| \leq n$ und $v \neq \varepsilon$.

Setze $u = \varepsilon$ und $|v| = 1$ das erste Symbol von w.

- Falls $w = 1^k$, so ist auch $uv^i x$ vom Typ $1^\ell \in L$.
- Falls $w = 0^j 1^{k^2}$, so ist auch $uv^0 x \in L$ (für $j = 1$ ist $uv^0 x = x = 1^{k^2}$). Für $i \geq 1$ gilt $uv^i x = 0^{j+i} 1^{k^2} \in L$.

Trotzdem ist L nicht regular. Dies lässt sich mit dem verallgemeinertem Pumping Lemma zeigen.

Verallgemeinertes PL für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| \geq n$ und jede Darstellung $w = tyx$ mit $|y| = n$ gilt:
Für das Teilwort y existiert eine Darstellung $y = uvz$ mit $v \neq \varepsilon$ bei der auch $tuv^i zx \in L$ ist für alle $i \in \mathbb{N}_0$.

Verallgemeinertes PL für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| \geq n$ und jede Darstellung $w = tyx$ mit $|y| = n$ gilt:
Für das Teilwort y existiert eine Darstellung $y = uvz$ mit $v \neq \varepsilon$ bei der auch $tuv^i zx \in L$ ist für alle $i \in \mathbb{N}_0$.

Beweis:

- Sei L eine reguläre Sprache.
- Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ der deterministische endliche Automat, der L erkennt.
- Setze $n := |Q| + 1$.
- Sei $tyx \in L$ mit $|y| = n$.
- Sei $q_0, \ldots, q_n$ die Folge der Zustände, die bei der Abarbeitung von y durchlaufen werden.

Verallgemeinertes PL für reguläre Sprachen

Beweis:

- Sei L eine reguläre Sprache.
- Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ der deterministische endliche Automat, der L erkennt.
- Setze $n := |Q| + 1$.
- Sei $tyx \in L$ mit $|y| = n$.
- Sei $q_0, \ldots, q_n$ die Folge der Zustände, die bei der Abarbeitung von y durchlaufen werden.
- Es enthält $q_0, \ldots, q_n$ mindestens einen Zykel
- Es gibt Zerlegung $y = uvz$ so dass v der Buchstabenfolge entspricht, die beim Durchlaufen des Zykels abgearbeitet wird.
- Insbesondere ist v nicht leer.
- Dieser Zykel kann dann beliebig oft durchlaufen werden.
- Also ist auch $tuv^i zx$ ist ein gültiges Wort, das der Automat erkennt.

Verallgemeinertes PL für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| \geq n$ und jede Darstellung $w = tyx$ mit $|y| = n$ gilt:
Für das Teilwort y existiert eine Darstellung $y = uvz$ mit $v \neq \varepsilon$ bei der auch $tuv^i zx \in L$ ist für alle $i \in \mathbb{N}_0$.

- Es enthält $q_0, \ldots, q_n$ mindestens einen Zykel
- Es gibt Zerlegung $y = uvz$ so dass v der Buchstabenfolge entspricht, die beim Durchlaufen des Zykels abgearbeitet wird.
- Insbesondere ist v nicht leer.
- Dieser Zykel kann dann beliebig oft durchlaufen werden.
- Also ist auch $tuv^i zx$ ist ein gültiges Wort, das der Automat erkennt.

Kapitel

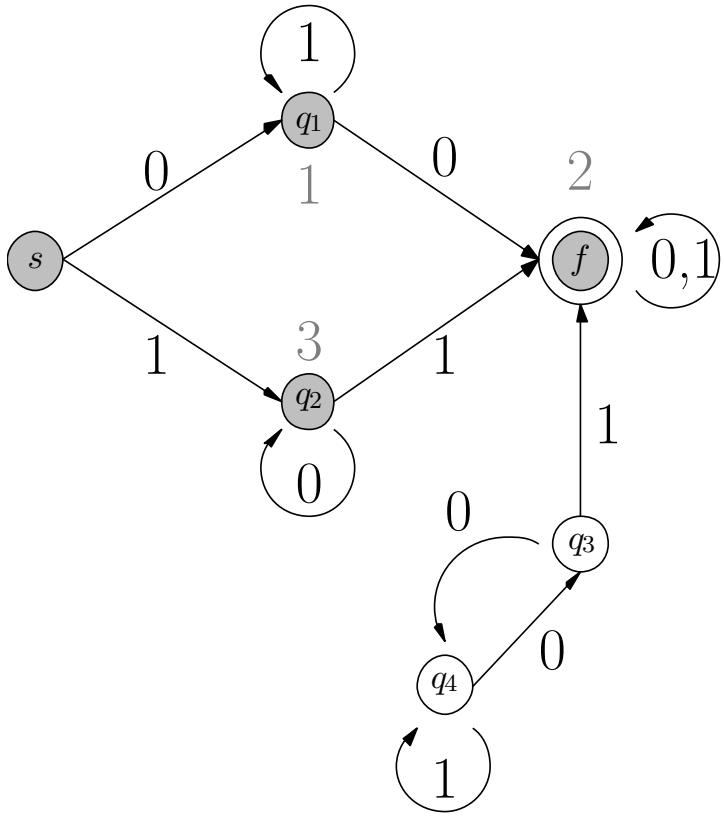
- **Minimierung von Automaten**
- **Äquivalenzklassenautomat**

Frage: Kann man konstruktiv die Anzahl der Zustände eines deterministischen endlichen Automatens erheblich verringern?

Frage: Kann man konstruktiv die Anzahl der Zustände eines deterministischen endlichen Automatens erheblich verringern?

Definition:
Zustände eines (deterministischen) endlichen Automatens, die vom Anfangszustand aus nicht erreichbar sind, heißen **überflüssig**.

Beispiel



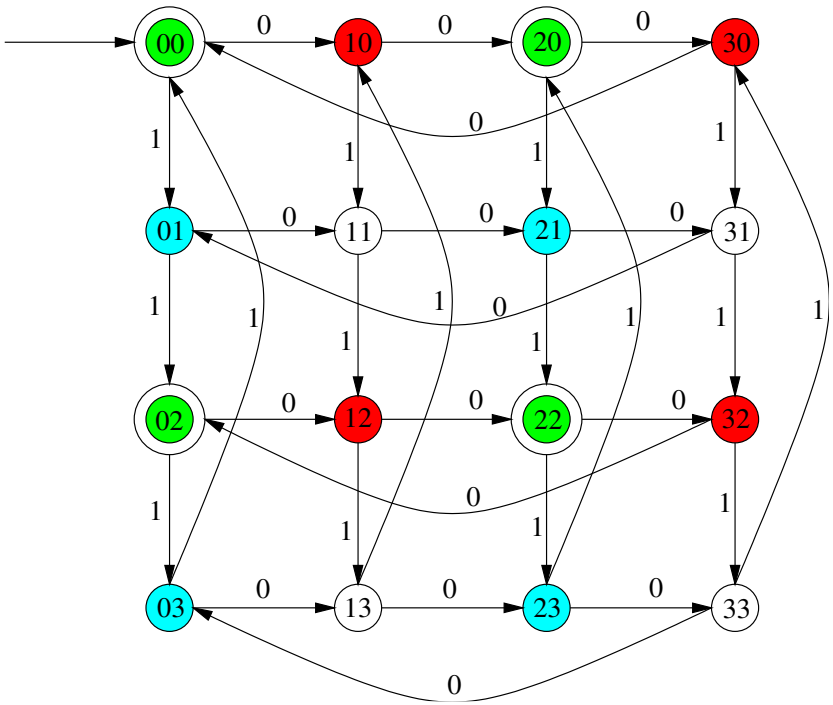
- Wir können endliche Automaten als gerichtete Graphen auffassen.
- Die überflüssigen Zustände entsprechen dann den Knoten, zu denen es vom Anfangsknoten aus keinen gerichteten Weg gibt.
- Eine Tiefensuche (**D**epth–**F**irst **S**earch, DFS) in dem Graphen liefert damit alle nicht überflüssigen Zustände.

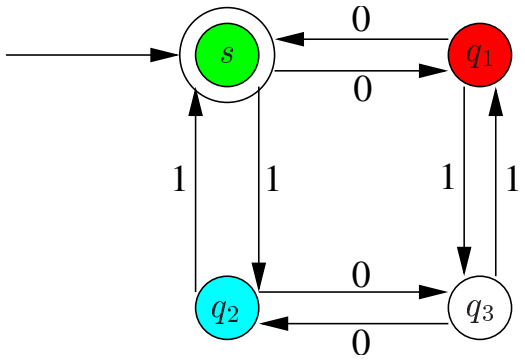
Satz:
Die Menge aller überflüssigen Zustände eines (deterministischen) endlichen Automaten kann in der Zeit $\mathcal{O}(|Q| \cdot |\Sigma|)$ berechnet werden.

Beweis: Wende DFS ab dem Startzustand an. Dies erfordert einen Aufwand proportional zu der Anzahl der Kanten in dem Graphen.

- Ein deterministischer endlicher Automat ohne überflüssige Zustände muss jedoch noch nicht minimal sein.

Beispiel





Beide Automaten akzeptieren die Sprache

$$L = \{w \in \{0,1\}^* \mid (|w|_0 \mod 2) = (|w|_1 \mod 2) = 0\}$$

mit $|w|_a$ = Anzahl der Vorkommen des Zeichens $a \in \Sigma$ in w

Äquivalenz

- Zwei Zustände haben dasselbe Akzeptanzverhalten, wenn es für das Erreichen eines Endzustandes durch Abarbeiten eines Wortes w unerheblich ist, aus welchem der beiden Zustände wir starten.
- Reduktion der Anzahl der Zustände durch Zusammlegen der Zustände mit gleichem Akzeptanzverhalten
- Letzten Beispiel: Färbung der Zustände mit gleichem Verhalten durch gleiche Farben

Definition (Äquivalenz):
Zwei Zustände p und q eines deterministischen endlichen Automaten heißen **äquivalent** $(p \equiv q)$, wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Offensichtlich ist $\equiv$ eine Äquivalenzrelation. Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Äquivalenz

- Zwei Zustände haben dasselbe Akzeptanzverhalten, wenn es für das Erreichen eines Endzustandes durch Abarbeiten eines Wortes w unerheblich ist, aus welchem der beiden Zustände wir starten.
- Reduktion der Anzahl der Zustände durch Zusammlegen der Zustände mit gleichem Akzeptanzverhalten
- Letzten Beispiel: Färbung der Zustände mit gleichem Verhalten durch gleiche Farben

Definition (Äquivalenz):
Zwei Zustände p und q eines deterministischen endlichen Automaten heißen **äquivalent** $(p \equiv q)$, wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(q, w) \in F.$$

Offensichtlich ist $\equiv$ eine Äquivalenzrelation. Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Der Äquivalenzklassenautomat

Definition (Äquivalenzklassenautomat):
Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

- $Q^{\equiv} := \{[q] \mid q \in Q\}$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f] \mid f \in F\}$

Der Äquivalenzklassenautomat

Definition (Äquivalenzklassenautomat):
Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

- $Q^{\equiv} := \{[q] \mid q \in Q\}$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f] \mid f \in F\}$

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

Beweis: Wir müssen zeigen, dass $F^{\equiv}$ und $\delta^{\equiv}$ wohldefiniert sind, der Rest ist klar. Dazu zeigen wir:

- ein Endzustand kann nur zu einem Endzustand äquivalent sein,
- δ führt äquivalente Zustände beim Lesen desselben Symbols wieder in äquivalente Zustände über.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

- ein Endzustand kann nur zu einem Endzustand äquivalent sein,

Für ε gilt:

$$\delta(p, \varepsilon) \in F \Leftrightarrow \delta(q, \varepsilon) \in F \ .$$

Es ist

$$\delta(p, \varepsilon), \delta(q, \varepsilon) \in F \text{ genau für } p, q \in F \ .$$

Also:

$$\text{Falls } p \equiv q, \text{ dann gilt } p, q \in F \text{ oder } p, q \notin F.$$

Also ist $F^{\equiv}$ wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

- δ führt äquivalente Zustände beim Lesen desselben Symbols wieder in äquivalente Zustände über.

Sei $p \equiv q$. Dann gilt für alle $w \in \Sigma^*$

$$\delta(q, w) \in F \Leftrightarrow \delta(p, w) \in F$$

Somit gilt nach Definition von $\equiv$ auch für alle $a \in \Sigma$:

$$\delta(\delta(q, a), w) = \delta(q, aw) \in F \Leftrightarrow \delta(p, aw) = \delta(\delta(p, a), w) \in F.$$

Damit folgt $\delta(q, a) \equiv \delta(p, a)$, also ist auch $\delta^{\equiv}$ wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu $\mathcal{A}$ akzeptiert dieselbe Sprache wie $\mathcal{A}$.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu $\mathcal{A}$ akzeptiert dieselbe Sprache wie $\mathcal{A}$.

Beweis:

- Sei $w \in \Sigma^*$, $q_0 := s, q_1, \ldots, q_n$ die Folge der Zustände, die von $\mathcal{A}$ bei der Abarbeitung von w durchlaufen werden.
- Bei Abarbeitung von w in $\mathcal{A}^{\equiv}$ werden dann die Zustände $[q_0], [q_1], \ldots, [q_n]$ durchlaufen.
- $\mathcal{A}$ akzeptiert w genau dann, wenn $q_n \in F$ gilt. $\mathcal{A}^{\equiv}$ akzeptiert w genau dann, wenn $[q_n] \in F^{\equiv}$ gilt.
- Nach Definition von $\mathcal{A}^{\equiv}$ ist $q_n \in F$ genau dann, wenn $[q_n] \in F^{\equiv}$ gilt.

Frage

Frage:
Wie konstruiert man $\mathcal{A}^{\equiv}$ zu $\mathcal{A}$? D.h. wie berechnet man alle Äquivalenzklassen zu den Zuständen von $\mathcal{A}$?

Beweis der Äquivalenz von zwei Zuständen p scheint aufwendig:
Nach Definition muss nachgewiesen werden, dass für alle $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

- Es gibt jedoch unendlich viele $w \in \Sigma^*$.
- Es ist einfacher für p und q zu zeigen, dass p nicht äquivalent zu q ist.
- Dafür benötigen wir *nur ein* Wort $w \in \Sigma^*$ mit
  $\delta(p, w) \in F$ aber $\delta(q, w) \notin F$, oder
  $\delta(p, w) \notin F$ aber $\delta(q, w) \in F$.

Idee

Notation:
Wir bezeichnen ein solches Wort w als **Zeuge** für die Nichtäquivalenz von p und q und sagen w trennt p und q.

Idee: Teste systematisch Zustandspaare auf Nichtäquivalenz

- Betrachte alle Worte aus Σ^* in aufsteigender Länge.
- Überprüfe für jedes Wort, ob es Zeuge für Nichtäquivalenz von zwei Zuständen ist.

Frage

Frage
Wann kann dieses Verfahren abgebrochen werden?

- Sei $w = aw'$ ein *kürzester* Zeuge für $p \not\equiv q$.
- Dann ist w' Zeuge für $p' := \delta(p, a) \not\equiv \delta(q, a) =: q'$.
- Wenn es für $p' \not\equiv q'$ einen kürzeren Zeugen w'' gäbe, so wäre aw'' ein kürzerer Zeuge für $p \not\equiv q$ als w.
- Dies ist aber ein Widerspruch dazu, dass w ein kürzester Zeuge ist.

- **Fazit:** Wenn wir alle Wörter aus Σ^* in der Reihenfolge ihrer Länge darauf testen, ob sie Zeuge sind, und für eine bestimmte Länge kein Zeuge mehr für eine Nichtäquivalenz auftritt, so kann das Verfahren abgebrochen werden.

Vorgehensweise

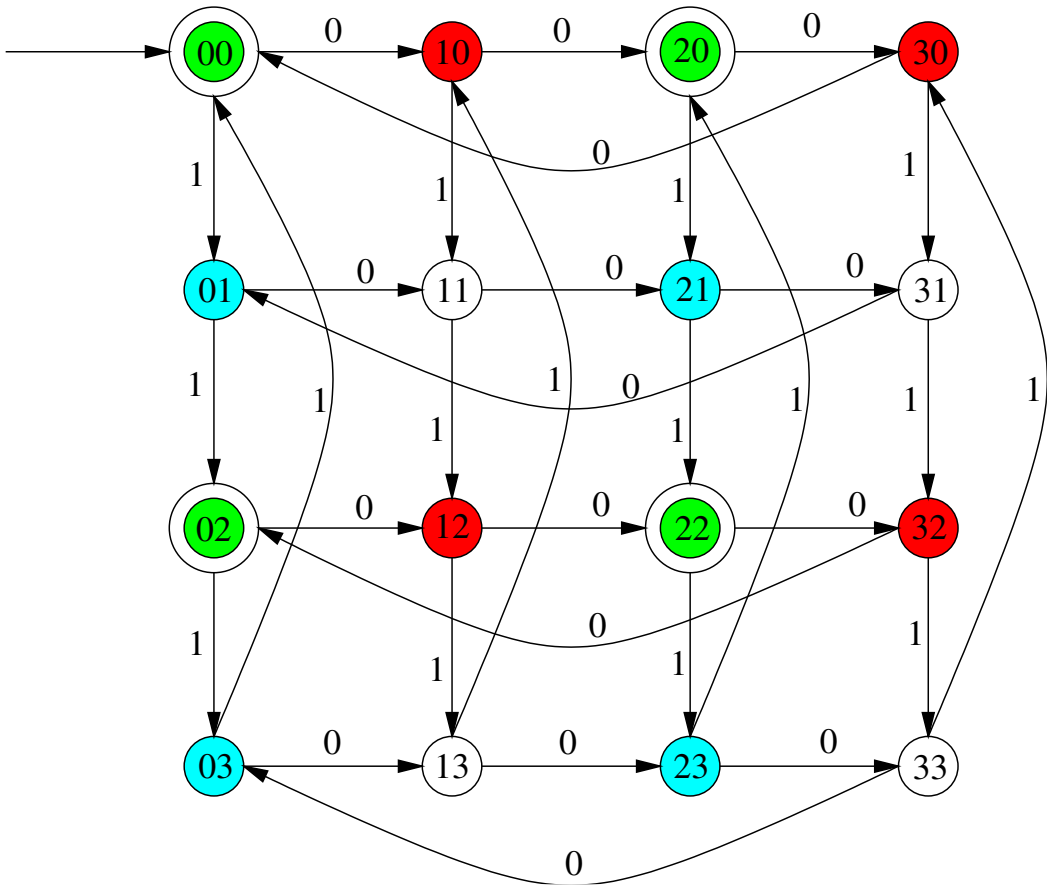
Vorgehensweise für die Konstruktion von $\mathcal{A}^{\equiv}$ aus $\mathcal{A}$

- Betrachte alle Zustandspaare und zunächst ε,
- dann alle Elemente aus Σ,
- dann alle Wörter der Länge 2 aus Σ^*,
- u.s.w.

Zunächst betrachte alle Zustände als eine Klasse.

- Dann trennt ε die Zustände aus F von denen aus $Q \setminus F$.
- Danach testen wir nur noch Paare von Zuständen aus F beziehungsweise $Q \setminus F$.
- Durch mindestens ein Wort der Länge 1 wird entweder F oder $Q \setminus F$ weiter getrennt, oder das Verfahren ist beendet.
- Dies wird iterativ so weitergeführt mit Wörtern wachsender Länge.

Beispiel zur Vorgehensweise



Vorläufige Äquivalenzklassen

vorher

$\{00, 01, 02, 03, 10, 11, 12, 13, 20, 21, 22, 23, 30, 31, 32, 33\}$

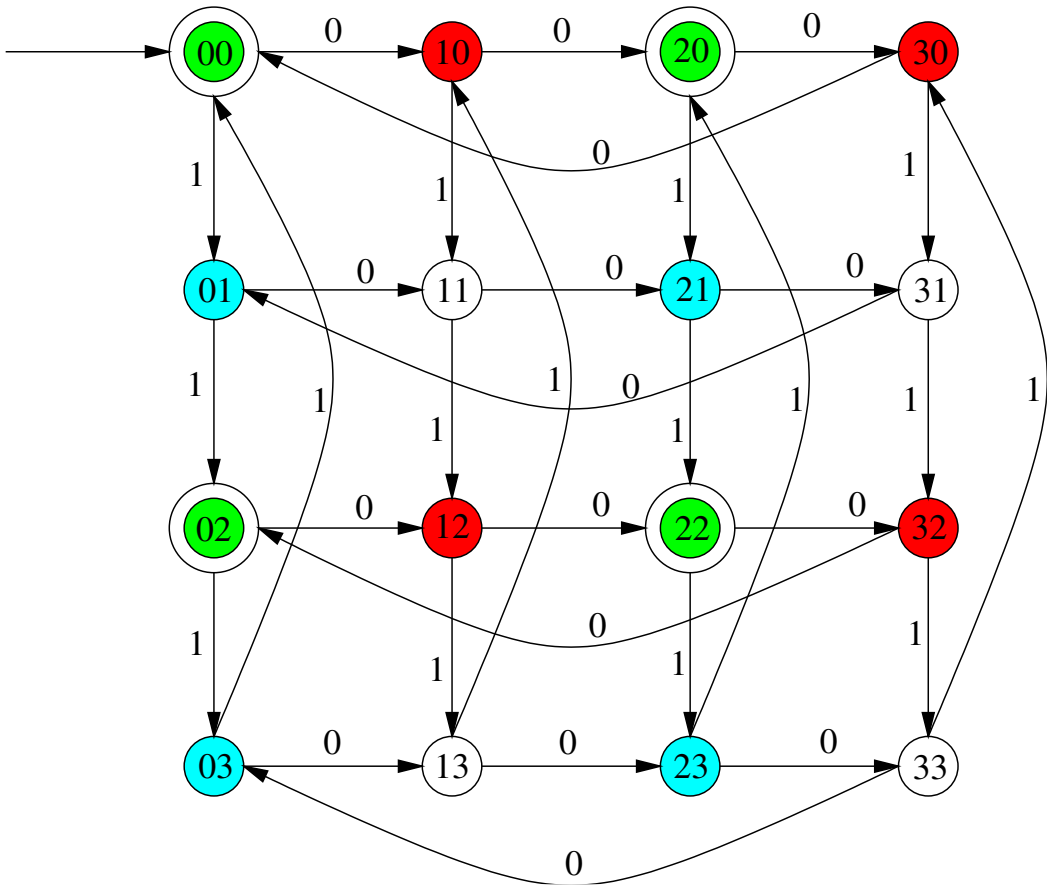
nachher

$\{00, 02, 20, 22\}$

$\{01, 03, 10, 11, 12, 13, 21, 23, 30, 31, 32, 33\}$

ε trennt $\underbrace{\{00, 02, 20, 22\}}_{\text{grün}}$ von $\{01, 03, 10, 11, 12, 13, 21, 23, 30, 31, 32, 33\}$

Beispiel zur Vorgehensweise



Vorläufige Äquivalenzklassen

vorher

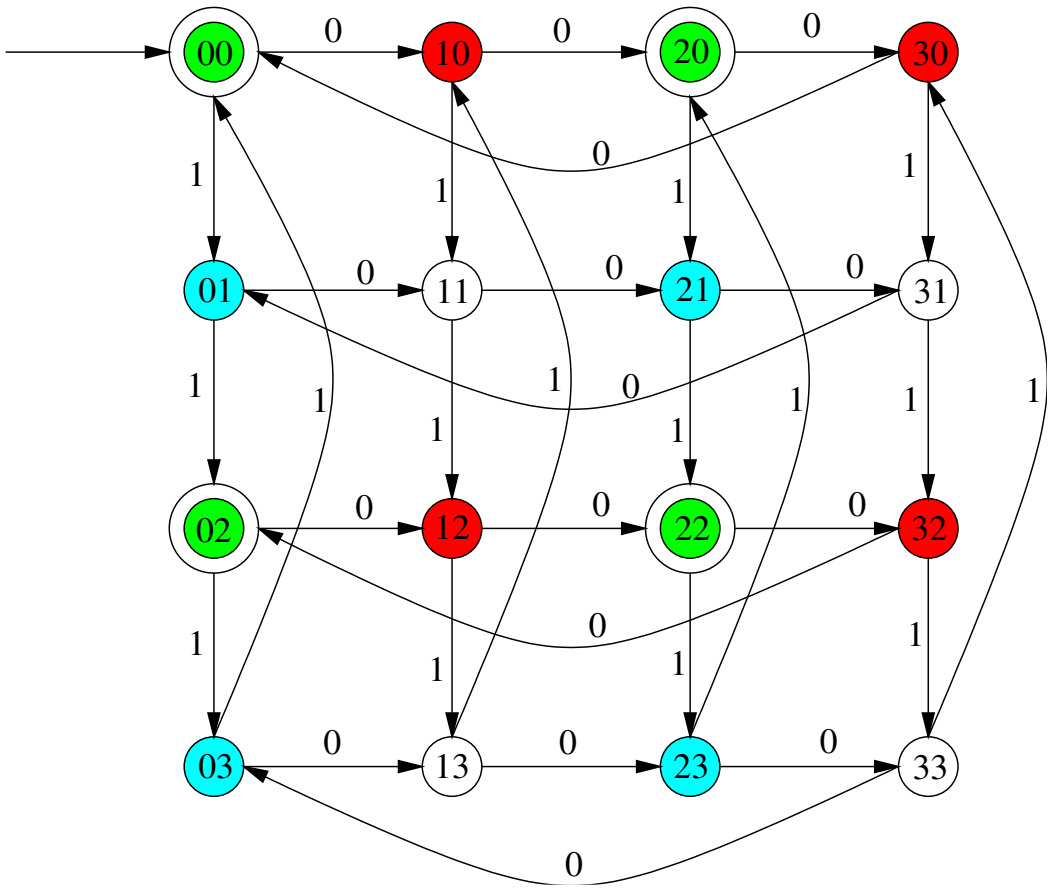
$\{00, 02, 20, 22\}$
$\{01, 03, 10, 11, 12, 13, 21, 23, 30, 31, 32, 33\}$

nachher

$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 11, 13, 21, 23, 31, 33\}$

0 trennt $\underbrace{\{10, 30, 12, 32\}}_{\text{rot}}$ von $\{01, 03, 11, 13, 21, 23, 31, 33\}$

Beispiel zur Vorgehensweise



Vorläufige Äquivalenzklassen

vorher

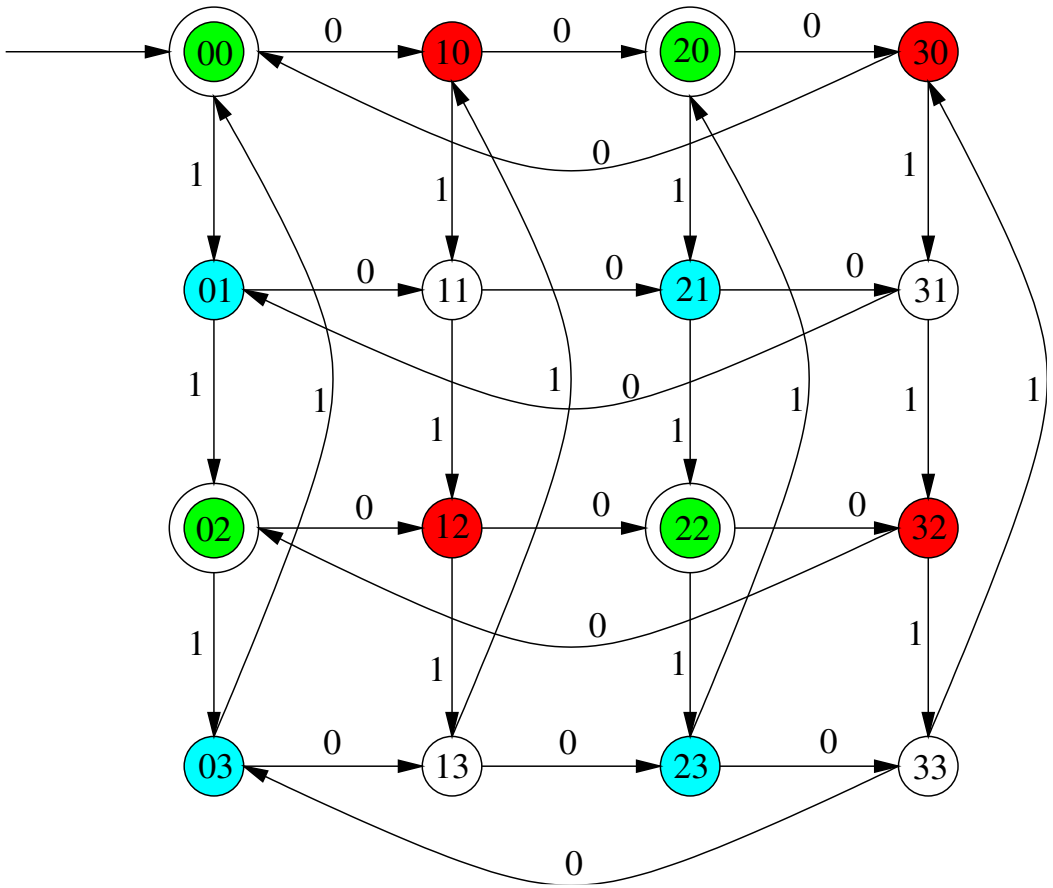
$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 11, 13, 21, 23, 31, 33\}$

nachher

$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 21, 23\}$
$\{11, 13, 31, 33\}$

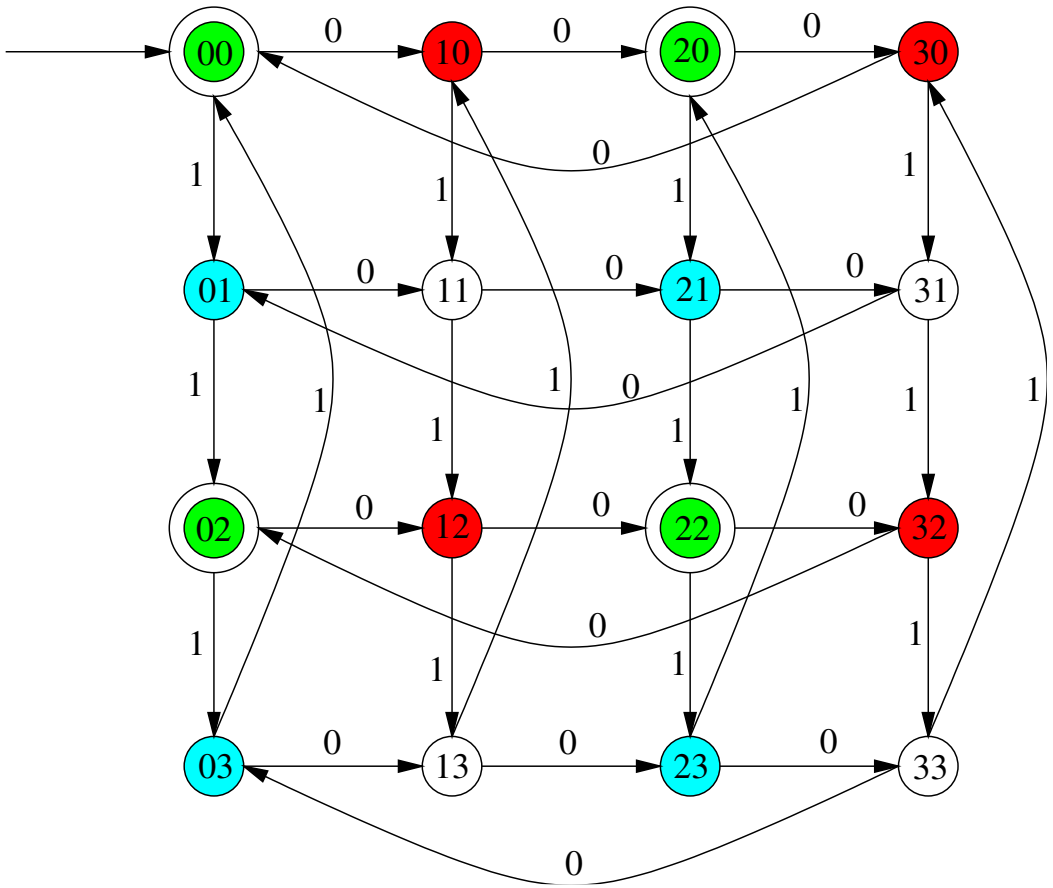
1 trennt $\underbrace{\{01, 03, 21, 23\}}_{\text{blau}}$ von $\underbrace{\{11, 13, 31, 33\}}_{\text{weiß}}$

Beispiel zur Vorgehensweise



Die Wörter $00, 01, 10, 11$ trennen keine Zustandspaare mehr.

Beispiel zur Vorgehensweise



Äquivalenzklassen

$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 21, 23\}$
$\{11, 13, 31, 33\}$

Fazit: Die Äquivalenzklassen der Zustände sind:
$s = [00], q_1 = [01], q_2 = [10]$ und $q_3 = [11]$.

Frage

Frage:
Ist der Äquivalenzklassenautomat zu einem deterministischen endlichen Automaten schon der äquivalente Automat mit der minimalen Anzahl von Zuständen?

Frage

Frage:
Ist der Äquivalenzklassenautomat zu einem deterministischen endlichen Automaten schon der äquivalente Automat mit der minimalen Anzahl von Zuständen?

Antwort:
Ja, wir zeigen dies wie folgt:

- Zuerst konstruieren wir den minimalen Automaten zur Sprache L (Automat der Nerode-Relation)
- Anschließend zeigen wir, dass $\mathcal{A}^{\equiv}$ höchstens soviele Zustände hat wie der Automat der Nerode-Relation.

Definitionen: Rechtsinvarianz und Index

Definition (Rechtsinvarianz und Index):
Eine Äquivalenzrelation R über Σ^* heißt **rechtsinvariant**, wenn

für alle $x, y \in \Sigma^*$ gilt: falls $x \; R \; y$ so gilt auch $xz \; R \; yz$ für alle $z \in \Sigma^*$.

Den **Index** von R bezeichnen wir mit **ind(R)**; er ist die Anzahl der Äquivalenzklassen von Σ^* bezüglich R.

Nerode-Relation

Definition (Nerode-Relationen):
Für eine Sprache $L \subseteq \Sigma^*$ ist die **Nerode–Relation** R_L definiert durch: für $x, y \in \Sigma^*$ ist $x \; R_L \; y$ genau dann wenn $(xz \in L \Leftrightarrow yz \in L)$ für alle $z \in \Sigma^*$ gilt.

Die Nerode–Relation R_L zu einer Sprache $L \subseteq \Sigma^*$ ist eine rechtsinvariante Äquivalenzrelation. Es gilt:

$$\begin{aligned} x \; R_L \; y &\Rightarrow (xw \in L \Leftrightarrow yw \in L) \text{ für alle } w \in \Sigma^* \\ &\Rightarrow (xzw \in L \Leftrightarrow yzw \in L) \text{ für alle } w, \; z \in \Sigma^* \\ &\Rightarrow (xz \; R_L \; yz) \text{ für alle } z \in \Sigma^*. \end{aligned}$$

Satz von Nerode

Satz (von Nerode):
Die folgenden Aussagen sind äquivalent:

1. $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.
2. L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit endlichem Index.
3. Die Nerode–Relation hat endlichen Index.

Beweis zu Satz von Nerode: (1) $\rightarrow$ (2)

(1) $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.

(2) L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit endlichem Index.

Beweis: Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ der deterministische endliche Automat, der L akzeptiert, und $R_{\mathcal{A}}$ wie folgt definiert:

$$\forall\, x, y \in \Sigma^* : x\ R_{\mathcal{A}}\ y \iff \delta(s, x) = \delta(s, y).$$

- $R_{\mathcal{A}}$ ist eine rechtsinvariante Äquivalenzrelation.
- Der Index von $R_{\mathcal{A}}$ ist die Anzahl der nicht überflüssigen Zustände von $\mathcal{A}$, also endlich.
- Also ist L die Vereinigung der Äquivalenzklassen von $R_{\mathcal{A}}$, die zu den Endzuständen von $\mathcal{A}$ gehören.

Beweis zu Satz von Nerode: (2) $\rightarrow$ (3)

(2) L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation R mit endlichem Index.

(3) Die Nerode–Relation hat endlichen Index.

Beweis:

- Wir zeigen $x\ R\ y$ impliziert $x\ R_L\ y$ (R_L eine Vergröberung von R)
- Dann gilt $\text{ind}(R_L) \leq \text{ind}(R) < \infty$.

Sei also $x\ R\ y$.

- Da R rechtsinvariant ist, gilt für alle $z \in \Sigma^*$: $xz\ R\ yz$.
- Voraussetzung: Jede Äquivalenzklasse von R gehört entweder ganz oder gar nicht zu L
- Also: $xz, yz \in L$ oder $xz, yz \notin L$.
- Damit folgt $x\ R_L\ y$.

Beweis zu Satz von Nerode: (3) $\rightarrow$ (1)

(3) Die Nerode–Relation hat endlichen Index.

(1) $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L. Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

Beweis zu Satz von Nerode: (3) $\rightarrow$ (1)

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L. Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

δ ist wohldefiniert:

- Falls $[w]_{R_L} = [w']_{R_L}$ dann gilt $w \; R_L \; w'$ und wegen Rechtsinvarianz von R_L auch $wa \; R_L \; w'a$.
- Also ist $[wa]_{R_L} = [w'a]_{R_L}$.

Beweis zu Satz von Nerode: (3) $\rightarrow$ (1)

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L. Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

Es bleibt zu zeigen, dass $\mathcal{A}$ genau L akzeptiert.

- Nach Konstruktion ist $\delta(s, w) = \delta([\varepsilon], w) = [\varepsilon w]_{R_L} = [w]_{R_L}$.
- Also wird w von $\mathcal{A}$ akzeptiert genau dann, wenn $[w] \in F$ gilt, d.h. wenn $w \in L$.

Korollar

Korollar
Der im dritten Beweisteil zum Satz von Nerode konstruierte Automat $\mathcal{A}$ zu R_L — der **Automat der Nerode–Relation** — ist minimal.

Korollar

Korollar
Der im dritten Beweisteil zum Satz von Nerode konstruierte Automat $\mathcal{A}$ zu R_L — der **Automat der Nerode–Relation** — ist minimal.

Beweis: Sei $\mathcal{A}' := (Q', \Sigma, \delta', s', F')$ ein deterministischer endlicher Automat, der L akzeptiert.

- Aus 1 $\Rightarrow$ 2 folgt, dass eine rechtsinvariante Äquivalenzrelation $R_{\mathcal{A}'}$ mit $\text{ind}(R_{\mathcal{A}'}) \leq |Q'|$ existiert.
- Wegen 2 $\Rightarrow$ 3 gilt: $\text{ind}(R_L) \leq \text{ind}(R_{\mathcal{A}'})$.
- Mit 3 $\Rightarrow$ 1 folgt

$$|Q| = \text{ind}(R_L) \leq \text{ind}(R_{\mathcal{A}'}) \leq |Q'|,$$

für den Nerode–Automat $\mathcal{A} = (Q, \Sigma, \delta, s, F)$.

Minimalität des Äquivalenzklassenautomats

Satz (Minimalität des Äquivalenzklassenautomats):
Der Äquivalenzklassenautomat $A^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ohne überflüssige Zustände ist minimal.

Minimalität des Äquivalenzklassenautomats

Satz (Minimalität des Äquivalenzklassenautomats):
Der Äquivalenzklassenautomat $A^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ohne überflüssige Zustände ist minimal.

Beweis: Sei L die vom Automaten $\mathcal{A}$ bzw. $\mathcal{A}^{\equiv}$ akzeptierte Sprache.

- $A^{\equiv}$ hat keine überflüssigen Zustände.
- Letzter Korollar: Es genügt zu zeigen, dass $|Q^{\equiv}| = \text{ind}(R_L)$.
- Es bleibt zu zeigen, dass für alle $x, y \in \Sigma^*$ gilt:
  $x \; R_L \; y \Rightarrow \delta(s, x) \equiv \delta(s, y)$.

$$
\begin{aligned}
x \; R_L \; y &\Rightarrow \forall\, z \in \Sigma^* : \; (xz \in L \Leftrightarrow yz \in L) \\
&\Rightarrow \forall\, z \in \Sigma^* : \; (\delta(s, xz) \in F \Leftrightarrow \delta(s, yz) \in F) \\
&\Rightarrow \forall\, z \in \Sigma^* : \; (\delta(\delta(s, x), z) \in F \Leftrightarrow \delta(\delta(s, y), z) \in F) \\
&\Rightarrow \delta(s, x) \equiv \delta(s, y)
\end{aligned}
$$

Zusammenfassung

- Ein DEA ist ein Modell für einen sehr einfachen Computer
- Folgende Mengen sind gleich
  - Die Menge der regulären Sprachen
  - Die Menge aller Sprachen, die von einem DEA erkannt werden.
  - Die Menge aller Sprachen, die von einem NEA erkannt werden.
- Mit Potenzmengenkonstruktion kann ein zu einem NEA äquivalenter DEA konstruiert werden.
- Das Pumping Lemma für reguläre Sprachen und das Verallgemeinerte Pumping-Lemma für reguläre Sprachen sind Hilfsmittel, mit denen für manche Sprachen gezeigt werden kann, dass sie nicht regulär sind.
- Der Äquivalenzklassenautomat zu einem DEA ohne überflüssige Zustände akzeptiert die gleiche Sprache und ist zustandsminimal.
- Der Automat der Nerode-Relation zu einem DEA akzeptiert die gleiche Sprache und ist zustandsminimal