Grundbegriffe der Theoretischen Informatik

Sommersemester 2017 - Beate Bollig

Die Folien basieren auf den Materialien von Thomas Schwentick.

Teil A: Reguläre Sprachen

4: Minimierung von Automaten

Der Borussia-Newsticker-Automat (1/3)

- Ich habe einen Bekannten, der ein ziemlich großer Fan von Borussia ist
- Er sammelt auch Fanartikel, die mit Borussia zu tun haben und verfolgt einen Newsticker, der ihn über Auktionen informiert
- Dabei möchte er auch Auktionen finden, in deren Beschreibung der Name „Borussia“ falsch geschrieben wurde (borussia, borusssia, borusia, borrussia, brussia, borissia)
- Können wir ihm dabei helfen?

Definition: MultiSearch

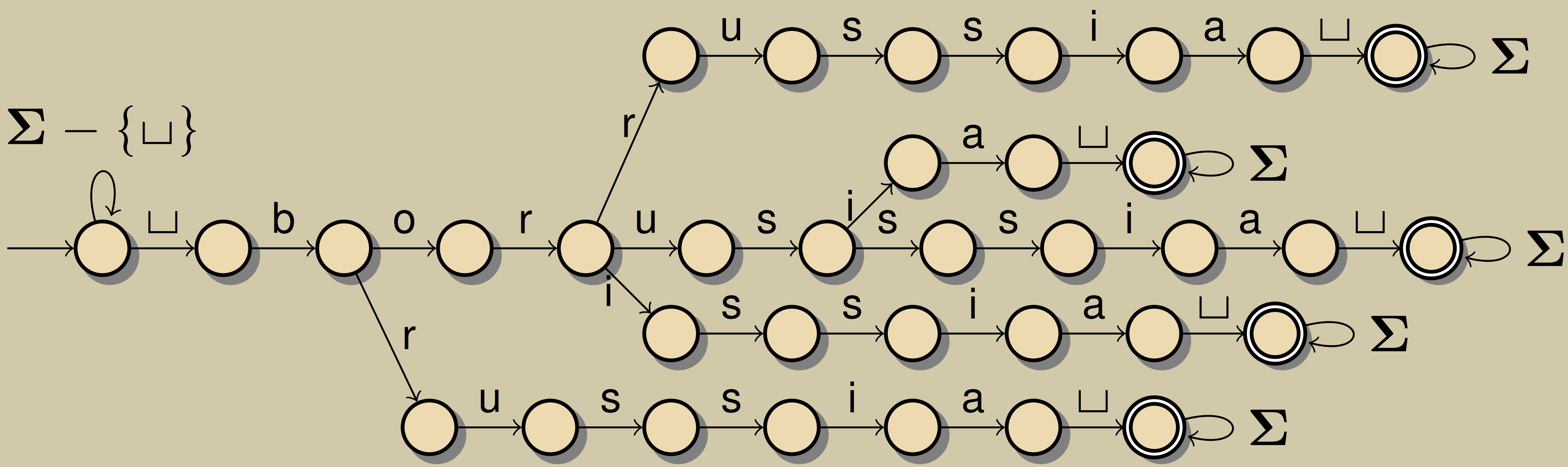
Gegeben: Menge $M = \{w_1, \ldots, w_n\}$ von Zeichenketten, String v

Frage: Kommt einer der Strings $w_1, \ldots, w_n$ in v vor?

- v entspricht also dem Newsticker
- $w_1, \ldots, w_n$ entsprechen den möglichen (richtigen und falschen) Schreibweisen von „Borussia“

Der Borussia-Newsticker-Automat (2/3)

Beispiel: Umsetzung als DFA



- $\boldsymbol{\Sigma} = \{a, \ldots, z, \sqcup\}$ ☞ $\sqcup$ steht für das Leerzeichen

- Jeder Zustand hat ausgehende Transitionen für alle Symbole aus $\boldsymbol{\Sigma}$
  - Nicht angezeigte Transitionen, bei denen Blanks gelesen werden, führen in den zweiten Zustand
  - Alle anderen nicht angezeigten Transitionen führen in den Startzustand

- **Ist diese Lösung optimal?**
- Oder gibt es einen kleineren Automaten für die „Borussia-Sprache"?

Idee: Wim Martens

Minimierung endlicher Automaten: Fragen & Antworten

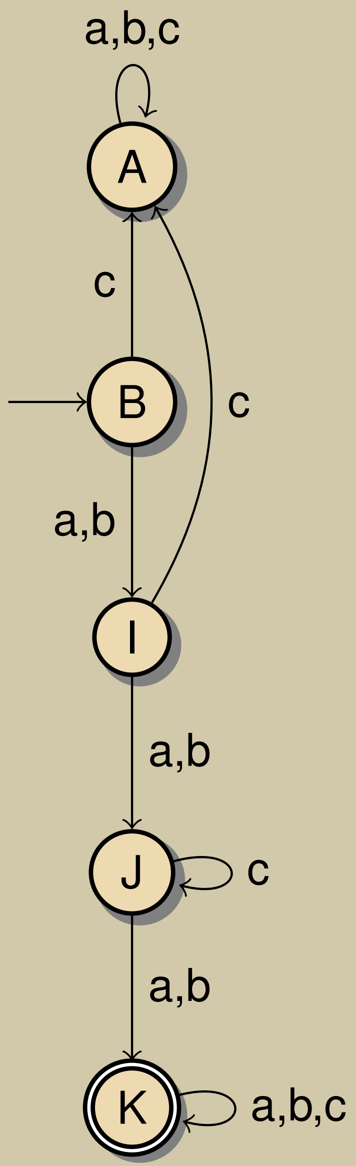
- Gibt es zu jedem DFA einen kleinsten äquivalenten DFA? **Natürlich!**

- Wieviele verschiedene kleinste äquivalente DFAs kann es geben? **Nur einen, bis auf Isomorphie**
- Kann man zu jedem DFA effizient den kleinsten äquivalenten DFA konstruieren? **Ja, das ist sogar ziemlich einfach**

- Zur genaueren Beantwortung dieser Fragen müssen wir die Struktur regulärer Sprachen etwas besser verstehen
- Dabei hilft uns eine etwas „mathematischere“ Sicht auf reguläre Sprachen

Minimierung: Grundidee

Beispiel



- p, q **F-äquivalent**: $p \in F \iff q \in F$

Beispiel

- Unerreichbare Zustände können gelöscht werden: G und H
- F-äquivalente Zustände, für die alle Übergänge in dieselben Zustände führen, können verschmolzen werden: E und K
- Senkenzustände können verschmolzen werden: L und A
- Allgemein: F-äquivalente Zustände, von denen aus das Akzeptierverhalten für alle nachfolgenden Eingabesequenzen gleich ist, können verschmolzen werden:
  - D, F, und J
  - C und I

- Was soll „das Akzeptierverhalten für alle nachfolgenden Eingabesequenzen ist gleich“ genau bedeuten?
- Wie lassen sich die verschmelzbaren Zustände berechnen?

Inhalt

Nerode-Relation: Definition und Beispiel

- Die folgende Definition präzisiert den vagen Begriff „das Akzeptierverhalten für alle möglichen nachfolgenden Eingabesequenzen ist gleich“ durch eine Äquivalenzrelation für Strings

Definition

- Sei $L \subseteq \Sigma^*$ eine Sprache
- Die **Nerode-Relation** $\sim_L$ auf Σ^* ist definiert durch:
  - $\underline{x \sim_L y} \overset{\text{def}}{\Leftrightarrow}$ für alle $z \in \Sigma^*$ gilt: $$xz \in L \iff yz \in L$$

- Wir werden sehen: ein Automat ist minimal für L, wenn jeweils alle Strings, die bezüglich $\sim_L$ äquivalent sind, den Automaten in den selben Zustand bringen

Beispiel

- Sei $L = L_g$ (gerade vielen Einsen und Nullen)
- Wann sind zwei Strings x und y bezüglich $\sim_L$ äquivalent?
  - Gilt $011 \sim_L 01$?
    - ∗ Nein: denn die Wahl von $z = 0$ ergibt:
      - · $(xz =)\ 0110 \in L$ aber
      - · $(yz =)\ 010 \notin L$
  - Gilt $10 \sim_L 010$?
    - ∗ Nein: denn die Wahl von $z = 1$ ergibt:
      - · $(xz =)\ 101 \notin L$ aber
      - · $(yz =)\ 0101 \in L$
  - Gilt $100 \sim_L 10110$?
    - ∗ Ja: Beide haben ungerade viele Einsen und gerade viele Nullen und erreichen L durch Anhängen von Strings mit ungerade vielen Einsen und gerade vielen Nullen
- Beobachtung: $\sim_L$ hat vier Äquivalenzklassen

Exkurs: Äquivalenzrelationen (1/3)

- Sei A eine Menge
- $\underline{A^n} \stackrel{\text{def}}{=}$ Menge der n-Tupel mit Einträgen aus A
- Eine Menge $R \subseteq A^n$ heißt **n-stellige Relation über A**

Beispiel

- **Gleiches-Semester-Relation**:
  - A: Menge aller Studierenden
  - $(x, y) \in R$, falls x und y im selben Semester sind

- **Gleicher-Rest-Relation modulo k**:
  - A : Menge $\mathbb{N}$ der natürlichen Zahlen
  - $(x, y) \in R$ falls x und y bei Division durch k den selben Rest haben
    - Schreibweise: $x \equiv_k y$

Exkurs: Äquivalenzrelationen (2/3)

Definition

- Eine 2-stellige (binäre) Relation R über A heißt
  - **reflexiv** $\overset{\text{def}}{\Leftrightarrow}$ für alle $x \in A$ gilt: $(x, x) \in R$
  - **symmetrisch** $\overset{\text{def}}{\Leftrightarrow}$ für alle $x, y \in A$ gilt: $(x, y) \in R \Rightarrow (y, x) \in R$
  - **transitiv** $\overset{\text{def}}{\Leftrightarrow}$ für alle $x, y, z \in A$ gilt: $(x, y) \in R, (y, z) \in R \Rightarrow (x, z) \in R$

- Eine 2-stellige reflexive, symmetrische, transitive Relation heißt **Äquivalenzrelation**

Beobachtung

- $\sim_L$ ist eine **Äquivalenzrelation**

- Äquivalenzrelationen werden oft mit $\sim$ statt R bezeichnet
- Infix-Notation: $x \sim y$ statt $(x, y) \in \sim$
- Beispiele: Gleiches-Semester-Relation, Gleicher-Rest-Relation

- **Äquivalenzklasse:** maximale Menge K von Elementen, so dass für alle $x, y \in K$: $x \sim y$
- $[x] \overset{\text{def}}{=}$ Äquivalenzklasse von x, also die Menge aller y mit $x \sim y$
- Wird eine Äquivalenzklasse in der Form $[x]$ benannt, so wird x oft als **Repräsentant** dieser Klasse bezeichnet

- Es gilt: $y \in [x] \iff [y] = [x]$

Beispiel

- Bezüglich der $\equiv_6$-Relation gilt:
  $$[2] = \{2, 8, 14, 20, \ldots\}$$

Satz von Myhill und Nerode (1/4)

- Zur Erinnerung:
  $x \sim_L y \overset{\text{def}}{\Leftrightarrow}$ für alle $z \in \Sigma^*$ gilt
  $$(xz \in L \iff yz \in L)$$

Satz 4.1

- Eine Sprache L ist genau dann regulär, wenn $\sim_L$ endlich viele Äquivalenzklassen hat

Beweis

- Wir zeigen zuerst:
  $\sim_L$ **hat endlich viele Klassen**
  $$\Rightarrow L \text{ ist regulär}$$
  - Wir definieren den **Äquivalenzklassenautomaten** $\mathcal{A}_L \overset{\text{def}}{=} (Q, \Sigma, \delta, s, F)$ für L wie folgt:
    * Q ist die Menge der Äquivalenzklassen von $\sim_L$
    * $s \overset{\text{def}}{=} [\epsilon]$
    * $F \overset{\text{def}}{=} \{[x] \mid x \in L\}$
    * für alle $x \in \Sigma^*, \sigma \in \Sigma$:
      $$\delta([x], \sigma) \overset{\text{def}}{=} [x\sigma]$$

Beweis (Forts.)

- Vorsicht: δ ist mit Hilfe von Repräsentanten der Klassen definiert
  - → wir müssen zeigen, dass die Definition des Funktionswertes **nicht** von der Wahl des Repräsentanten **abhängt**
  - Also: wenn wir zwei verschiedene Strings x, y aus einer Äquivalenzklasse von $\sim_L$ für die Definition von δ verwenden, erhalten wir jeweils das selbe Ergebnis

- Behauptungen:
  (1) **δ ist wohldefiniert**:
  $$x \sim_L y \Rightarrow x\sigma \sim_L y\sigma$$
  (2) **F ist sinnvoll definiert**:
  $$[x] \in F \iff x \in L$$
  (3) $L(\mathcal{A}_L) = L$

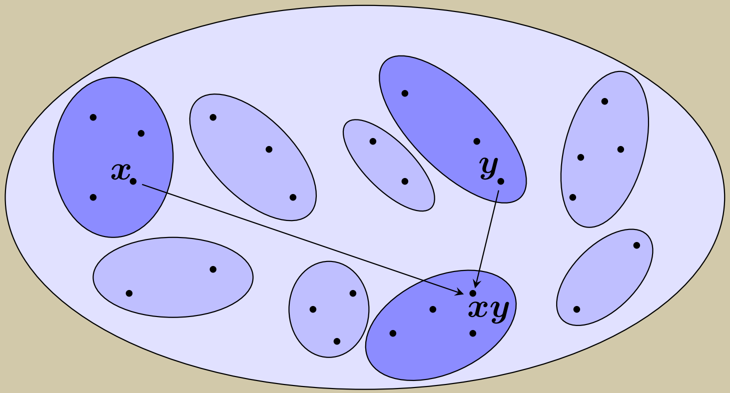
Exkurs: Wohldefiniertheit

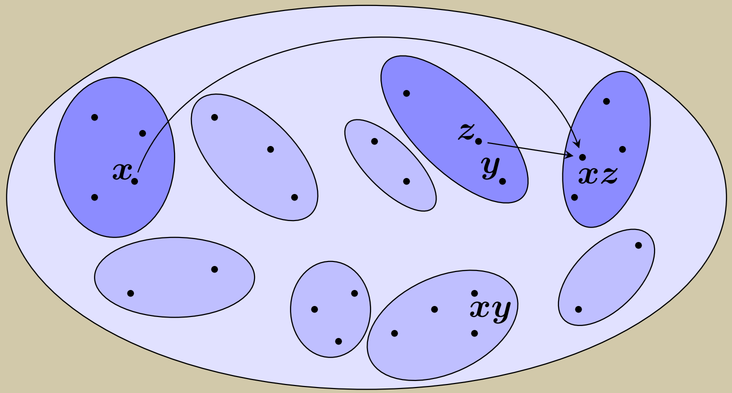
Eine Verknüpfung auf den Klassen einer Äquivalenzrelation, die auf Repräsentanten dieser Äquivalenzklasse definiert ist, heißt **wohldefiniert**, wenn die Definition unabhängig von der Wahl der jeweiligen Repräsentanten ist.

Sei $[x] \cdot [y] = [x \cdot y]$. Das Ergebnis der multiplikativen Verknüpfung der Äquivalenzklassen, in denen x und y liegen, soll die Äquivalenzklasse sein, in der das Produkt $x \cdot y = xy$ liegt, wobei diese Multiplikation über die Elemente x und y definiert ist.

Beispiel

Veranschaulichung einer Operation, die nicht wohldefiniert ist:





Satz von Myhill und Nerode: Beweis (2/4)

Beweis (Forts.)

(1) Zu zeigen: falls $x \sim_L y$, so gilt für alle $\sigma \in \Sigma$:
  - $x\sigma \sim_L y\sigma$

  - Sei also $x \sim_L y$ und $\sigma \in \Sigma$
  - Sei $z \in \Sigma^*$ beliebig:

$$\begin{aligned}(x\sigma)z \in L &\iff x(\sigma z) \in L \\ &\iff y(\sigma z) \in L \quad \text{☞ wegen } x \sim_L y \\ &\iff (y\sigma)z \in L\end{aligned}$$

(2) Zu zeigen: $[x] \in F \iff x \in L$
  - $[x] \in F \Rightarrow$ es gibt y mit $x \sim_L y$ und $y \in L$
  - ➨ Mit $z = \epsilon$ ergibt sich $x \in L \iff y \in L$, also: $x \in L$

  - Umgekehrt folgt aus $x \in L$ auch $[x] \in F$ nach Definition von F

Satz von Myhill und Nerode: Beweis (3/4)

Beweis (Forts.)

(3) Wir zeigen zunächst durch Induktion, dass für alle $w \in \Sigma^*$ gilt:

$$\delta^*(s, w) = [w] \qquad (\#)$$

- $w = \epsilon$ ✓

- $w = u\sigma$:

$$\begin{aligned} \delta^*(s, u\sigma) &= \delta(\delta^*(s, u), \sigma) && \text{☞ Def. } \delta^* \\ &= \delta([u], \sigma) && \text{☞ Ind.} \\ &= [u\sigma] && \text{☞ Def. } \delta \end{aligned}$$

- Also:

$$\begin{aligned} w \in L(\mathcal{A}_L) &\iff \delta^*(s, w) \in F && \text{☞ Def } L(\mathcal{A}_L) \\ &\iff [w] \in F && \text{☞ } (\#) \\ &\iff w \in L && \text{☞ (2)} \end{aligned}$$

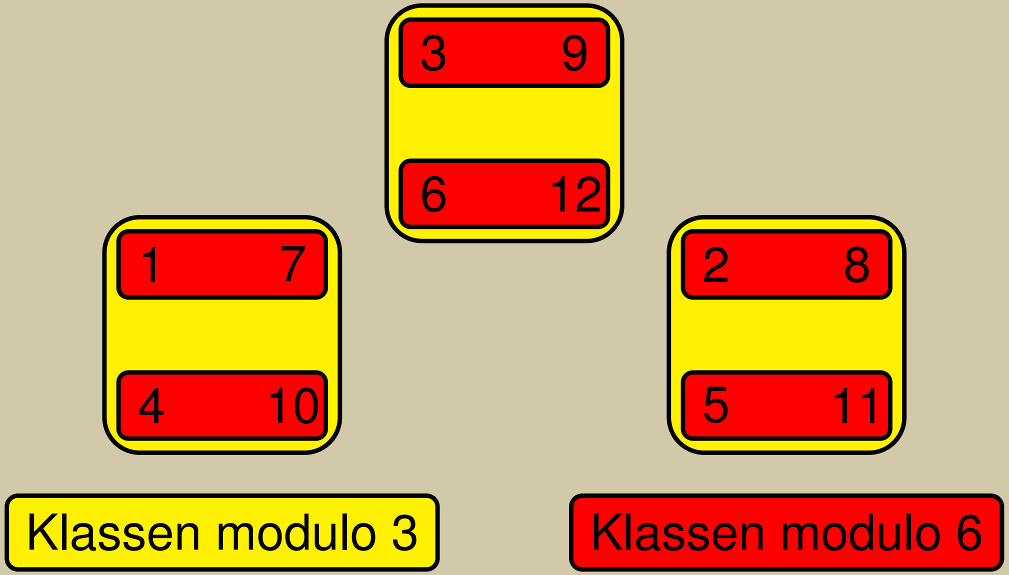
- Aus (1)-(3) folgt, dass $\mathcal{A}_L$ ein Automat für L ist

➨ L ist regulär

Exkurs: Äquivalenzrelationen (3/3)

- Für die „Rückrichtung“ des Beweises benötigen wir den Begriff der **Verfeinerung einer Äquivalenzrelation**

Beispiel



- Die Äquivalenzrelation $\equiv_6$ ist eine Verfeinerung der Äquivalenzrelation $\equiv_3$

Definition

- Seien $\sim_1, \sim_2$ Äquivalenzrelationen über derselben Grundmenge
- $\sim_1$ heißt **Verfeinerung von** $\sim_2$, wenn für alle x, y gilt: $x \sim_1 y \Rightarrow x \sim_2 y$

- Falls $\sim_1$ Verfeinerung von $\sim_2$ ist, gilt: Anzahl Klassen von $\sim_1 \geqslant$ Anzahl Klassen von $\sim_2$

- Weiteres Beispiel: die Gleiches-Semester-und-gleicher-Studiengang-Relation ist eine Verfeinerung der Gleiches-Semester-Relation

Satz von Myhill und Nerode: Beweis (4/4)

Beweis (Forts.)

- Jetzt zeigen wir:
  L regulär $\Rightarrow$
  $\sim_L$ hat endlich viele Klassen

- Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein DFA für L

- Wir definieren eine Äquivalenzrelation $\sim_{\mathcal{A}}$ mit $|Q|$ Klassen und zeigen:
  $\sim_{\mathcal{A}}$ ist eine Verfeinerung von $\sim_L$

- Dann folgt:
  - Anzahl Klassen von $\sim_L$
    $\leqslant$ Anzahl Klassen von $\sim_{\mathcal{A}}$
    $= |Q| < \infty$

Beweis (Forts.)

- Wir definieren $\sim_{\mathcal{A}}$ durch:
  $$\underline{x \sim_{\mathcal{A}} y} \overset{\text{def}}{\Leftrightarrow} \delta^*(s, x) = \delta^*(s, y)$$

- **Behauptung:** $\sim_{\mathcal{A}}$ ist eine Verfeinerung von $\sim_L$, also:
  für alle x, y gilt: $x \sim_{\mathcal{A}} y \Rightarrow x \sim_L y$

- Seien also $x, y \in \Sigma^*$ mit $x \sim_{\mathcal{A}} y$

➡ $\delta^*(s, x) = \delta^*(s, y)$

➡ für alle $z \in \Sigma^*$ gilt:
$$\delta^*(s, xz) = \delta^*(s, yz)$$

➡ für alle $z \in \Sigma^*$ gilt:
$$xz \in L \iff yz \in L$$

➡ $x \sim_L y$

- Damit ist der Beweis des Satzes von Myhill und Nerode vollständig

Satz von Myhill und Nerode: Anwendung (1/2)

- Mit dem Satz von Myhill und Nerode lässt sich also herausfinden, ob eine gegebene Sprache regulär ist:
  - Wir haben gesehen, dass die Relation $\sim_{L_g}$ vier Klassen hat
  - L_g ist also regulär
  - Der in Kapitel 2 konstruierte Automat $\mathcal{A}_g$ ist sogar im Wesentlichen der Äquivalenzklassen-automat zu L_g

  ✎ (bis auf Isomorphie, siehe später)

- Der Satz ist aber auch für den Nachweis, dass eine gegebene Sprache **nicht** regulär ist, nützlich

Satz von Myhill und Nerode: Anwendung (2/2)

Beispiel

- Wir berechnen die Äquivalenzklassen von $L_{ab} = \{a^n b^n \mid n \geqslant 0\}$
- Es gilt z.B.: $a^4 b \sim_{L_{ab}} a^5 b^2 \sim_{L_{ab}} a^6 b^3 \ldots$
- $\sim_{L_{ab}}$ hat die Klassen:
  - $B_k \stackrel{\text{def}}{=} \{a^{i+k} b^i \mid i \geqslant 1\}$, für jedes $k \geqslant 0$,
  - $A_k \stackrel{\text{def}}{=} \{a^k\}$, für jedes $k \geqslant 0$,
  - $C \stackrel{\text{def}}{=} \{a^i b^j \mid i < j\} \cup \overline{L(a^* b^*)}$, die Klasse aller Strings, für die es überhaupt keine Verlängerung gibt, die in L_{ab} liegt

- Notation:
  - Sei L eine Sprache über Σ und $v \in \Sigma^*$
  - $\underline{L/v} \stackrel{\text{def}}{=} \{z \in \Sigma^* \mid vz \in L\}$

Beispiel (Forts.)

- Um nachzuweisen, dass dies die Äquivalenzklassen von $\sim_{L_{ab}}$ sind, ist zu zeigen:
  (1) Jeder String kommt in einer Klasse vor ✓
  (2) Für alle Strings u, v in derselben Klasse gilt: $u \sim_{L_{ab}} v$
  (3) Für Strings u, v aus verschiedenen Klassen gilt: $u \not\sim_{L_{ab}} v$

- Dazu genügt es zu zeigen, dass
  (2') für alle Strings v einer Klasse die Menge L_{ab}/v gleich ist
  (3') für verschiedene Klassen die Mengen L_{ab}/v verschieden sind

- Für jedes k gilt:
  - Für $v \in B_k$ ist $L_{ab}/v = \{b^k\}$
  - Für $v \in A_k$ ist $L_{ab}/v = \{a^i b^{i+k} \mid i \geqslant 0\}$
- Für $v \in C$ ist $L_{ab}/v = \varnothing$

➡ Die Äquivalenzklassen sind korrekt angegeben

➡ unendlich viele Klassen $\Rightarrow$ L_{ab} nicht regulär

Minimaler Automat: Eindeutigkeit (1/3)

- Was bringt uns der Satz von Myhill und Nerode für die Minimierung von Automaten?
- Aus dem Beweis können wir direkt schließen:

Lemma 4.2

- Ist L eine reguläre Sprache und $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein DFA für L, dann gilt:

$$|Q| \geqslant \text{Anzahl Klassen von } \sim_L$$

- Also: Jeder Automat für L hat mindestens so viele Zustände wie der Äquivalenzklassenautomat $\mathcal{A}_L$
- $\mathcal{A}_L$ ist also **ein** minimaler Automat für L

- Wir zeigen gleich:
  - In einem gewissen Sinne ist $\mathcal{A}_L$ sogar in jedem Automaten für L enthalten
  - Und: falls $|Q|$ gleich der Anzahl der Klassen von $\sim_L$ ist, sind $\mathcal{A}$ und $\mathcal{A}_L$ „praktisch identisch“
- $\mathcal{A}_L$ ist also **der** minimale Automat für L

✎ Wir betrachten zuerst die Formalisierung von „praktisch identisch“: Isomorphie von DFAs

Minimaler Automat: Eindeutigkeit (2/3)

Definition

- Seien $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ DFAs mit dem selben Eingabealphabet

- **$\mathcal{A}_1$ und $\mathcal{A}_2$ sind isomorph**, falls es eine Bijektion $\pi : Q_1 \to Q_2$ gibt mit:
  (1) $\pi(s_1) = s_2$,
  (2) für alle $q \in Q_1$ gilt:
  $$q \in F_1 \iff \pi(q) \in F_2, \text{ und}$$
  (3) für alle $q \in Q_1$ und $\sigma \in \Sigma$ gilt:
  $$\pi(\delta_1(q, \sigma)) = \delta_2(\pi(q), \sigma)$$
- Notation: $\mathcal{A}_1 \cong \mathcal{A}_2$

- Informell bedeutet $\mathcal{A}_1 \cong \mathcal{A}_2$:
  - Die DFAs unterscheiden sich nur hinsichtlich der Namen der Zustände:
    * Wenn in $\mathcal{A}_1$ die Zustände gemäß π umbenannt werden, ergibt sich $\mathcal{A}_2$

Minimaler Automat: Eindeutigkeit (3/3)

Lemma 4.3

- Ist $\mathcal{A}$ ein Automat für eine Sprache L, der die selbe Anzahl von Zuständen wie $\mathcal{A}_L$ hat, so gilt: $\mathcal{A} \cong \mathcal{A}_L$

- Der Beweis findet sich im Anhang

Minimalautomat

- Insgesamt haben wir bisher gezeigt:

Satz 4.4

- Für jede reguläre Sprache L ist $\mathcal{A}_L$ der bis auf Isomorphie eindeutig bestimmte minimale Automat für L

Beweisskizze

- Nach Lemma 4.2 hat jeder Automat für L mindestens soviele Zustände wie $\mathcal{A}_L$
- Nach 4.3 ist jeder Automat für L, der genau so viele Zustände wie $\mathcal{A}_L$ hat, isomorph zu $\mathcal{A}_L$

- Wir betrachten jetzt, wie sich $\mathcal{A}_L$ aus einem gegebenen Automaten für L berechnen lässt

Inhalt

Minimaler Automat: Berechnung

- **Wie lässt sich $\mathcal{A}_L$ konstruieren?**

- Auch hierfür liefert Satz 4.1 einen Hinweis:
  - Ist $\mathcal{A}$ ein Automat für L, so ist $\sim_{\mathcal{A}}$ eine Verfeinerung von $\sim_L$

➡ $\mathcal{A}_L$ kann durch Zusammenlegen von Zuständen (Äquivalenzklassen) aus $\mathcal{A}$ erzeugt werden

  - Genauer: zwei Zustände p, q von $\mathcal{A}$ können zusammengelegt werden, wenn sie im folgenden Sinne **äquivalent** sind:

$(**)$ für alle $z \in \Sigma^*$ gilt:

$$\delta^*(p, z) \in F \iff \delta^*(q, z) \in F$$

➡ Um den minimalen Automaten zu konstruieren, genügt es also, zu berechnen, welche Zustände zusammengelegt werden können

- Es ist allerdings algorithmisch einfacher, zunächst zu berechnen, welche Zustände **nicht** zusammengelegt werden können

- Deshalb betrachten wir jetzt einen Algorithmus, der die Menge $N(\mathcal{A})$ der **nicht äquivalenten Paare** berechnet:

$$\underline{N(\mathcal{A})} \stackrel{\text{def}}{=} \{(p, q) \mid p, q \in Q, \exists w : \delta^*(p, w) \in F \not\Leftrightarrow \delta^*(q, w) \in F\}$$

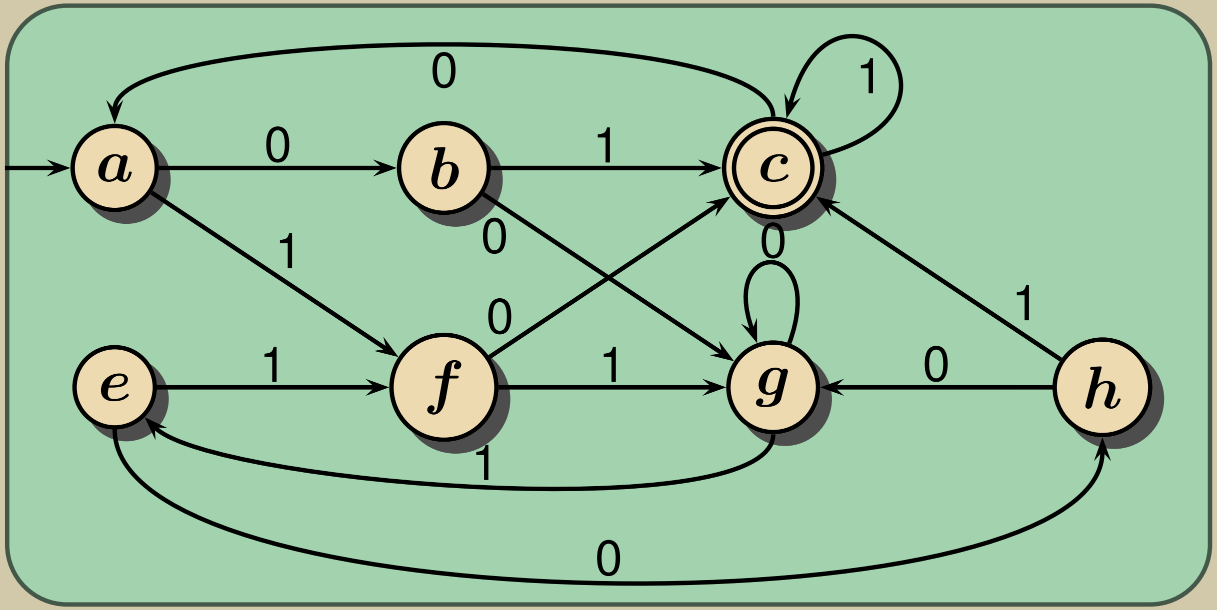
Der Markierungsalgorithmus

Markierungsalgorithmus

- Eingabe: $\mathcal{A} = (Q, \Sigma, \delta, s, F)$
- Ausgabe: Relation $N(\mathcal{A})$

1. $M := \{(p, q), (q, p) \mid p \in F, q \notin F\}$
2. $M' := \{(p, q) \notin M \mid \exists \sigma \in \Sigma : (\delta(p, \sigma), \delta(q, \sigma)) \in M\}$
3. $M := M \cup M'$
4. Falls $M' \neq \emptyset$, weiter mit 2.
5. Ausgabe M

Beispiel



	a	b	c	e	f	g	h
a		X^1	X^0		X^1	X^2	X^1
b			X^0	X^1	X^1	X^1	
c				X^0	X^0	X^0	X^0
e					X^1	X^2	X^1
f						X^1	X^1
g							X^1
h							

Nicht markiert: $(a, e), (b, h)$

Markierungsalgorithmus: Korrektheit (1/2)

Lemma 4.5

- Der Markierungsalgorithmus berechnet $N(\mathcal{A})$

Beweisskizze

- Zur Erinnerung:
  $N(\mathcal{A}) = \{(p,q) \mid p,q \in Q,$
  $\exists w : \delta^*(p,w) \in F \not\Leftrightarrow \delta^*(q,w) \in F\}$

- Wir zeigen:
  (a) Wenn (p,q) im k-ten Durchlauf (von 2.) markiert wird,
  dann gibt es einen String w der Länge k mit
  $$\delta^*(p,w) \in F \not\Leftrightarrow \delta^*(q,w) \in F$$
  (b) Wenn (p,q) durch den Algorithmus nicht markiert wird,
  dann gilt für alle Strings $w \in \Sigma^*$:
  $$\delta^*(p,w) \in F \iff \delta^*(q,w) \in F$$

Beweisskizze für (a)

- Beweis durch Induktion nach k
  - $k = 0$ ✓
  - Von $k-1$ zu k:
  - Zu jedem Paar (p,q), das im k-ten Durchlauf markiert wird, gibt es ein Paar (p',q'), das im $(k-1)$-ten Durchlauf markiert wird mit
    $$\delta(p,\sigma) = p', \delta(q,\sigma) = q'$$
  - Nach Induktion gibt es also einen String v der Länge $k-1$ mit
    $$\delta^*(p',v) \in F \not\Leftrightarrow \delta^*(q',v) \in F$$

➡ $\delta^*(p,\sigma v) \in F \not\Leftrightarrow \delta^*(q,\sigma v) \in F$

Markierungsalgorithmus: Korrektheit (2/2)

Lemma 4.5

- Der Markierungsalgorithmus berechnet $N(\mathcal{A})$

Beweisskizze

- Zur Erinnerung:
  $N(\mathcal{A}) = \{(p,q) \mid p,q \in Q, \exists w : \delta^*(p,w) \in F \not\Leftrightarrow \delta^*(q,w) \in F\}$

- Wir zeigen:
  (a) Wenn (p,q) im k-ten Durchlauf (von 2.) markiert wird, dann gibt es einen String w der Länge k mit
  $$\delta^*(p,w) \in F \not\Leftrightarrow \delta^*(q,w) \in F$$
  (b) Wenn (p,q) durch den Algorithmus nicht markiert wird, dann gilt für alle Strings $w \in \Sigma^*$:
  $$\delta^*(p,w) \in F \Longleftrightarrow \delta^*(q,w) \in F$$

Beweisskizze für (b)

- Beweis durch Widerspruch:
  - Angenommen, es gibt ein **Gegenbeispiel** (p,q,w), so dass
    - $*$ der Algorithmus (p,q) nicht markiert, aber
    - $*$ $\delta^*(p,w) \in F \not\Leftrightarrow \delta^*(q,w) \in F$

    - $*$ Sei (p,q,w) ein Gegenbeispiel mit dem kürzest möglichen w
    - $*$ Klar: $w \neq \epsilon$ wegen Schritt (1) des Algorithmus
    - $*$ Seien $v \in \Sigma^*$, $\sigma \in \Sigma$ mit $w = \sigma v$
    - $*$ Da (p,q) unmarkiert ist, ist auch $(\delta(p,\sigma), \delta(q,\sigma))$ unmarkiert
    - ➡ $(\delta(p,\sigma), \delta(q,\sigma), v)$ ist auch ein Gegenbeispiel, aber v ist kürzer als w
    - $*$ **Widerspruch zur Wahl von** (p,q,w)

Minimierungsalgorithmus

Minimierungsalgorithmus für DFA

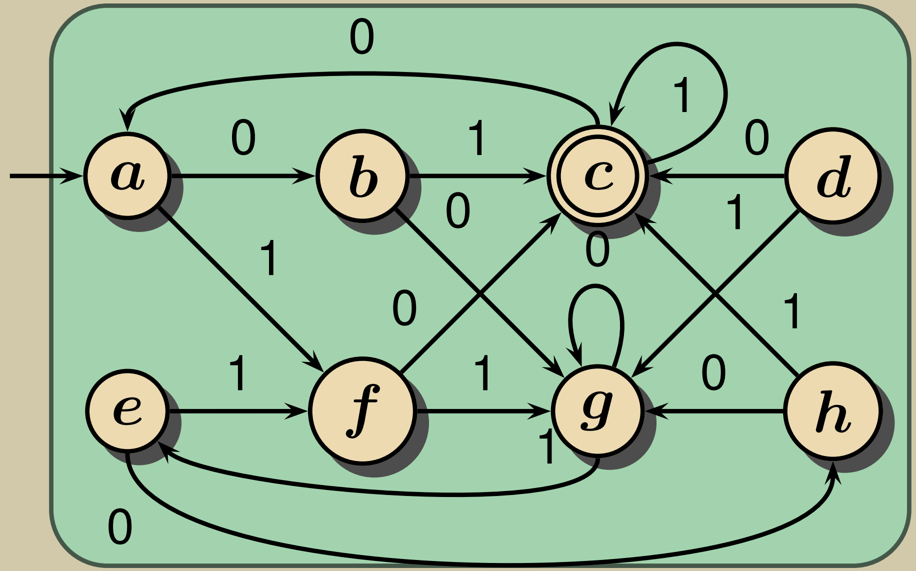
- Eingabe: $\mathcal{A} = (Q, \Sigma, \delta, s, F)$
- Ausgabe: minimaler Automat $\mathcal{A}'$ mit $L(\mathcal{A}') = L(\mathcal{A})$

1. Entferne alle Zustände von $\mathcal{A}$, die von s aus nicht erreichbar sind.
2. Berechne die Relation $N(\mathcal{A})$ mit dem Markierungsalgorithmus.
3. Verschmelze sukzessive alle nicht markierten Zustandspaare zu jeweils einem Zustand.

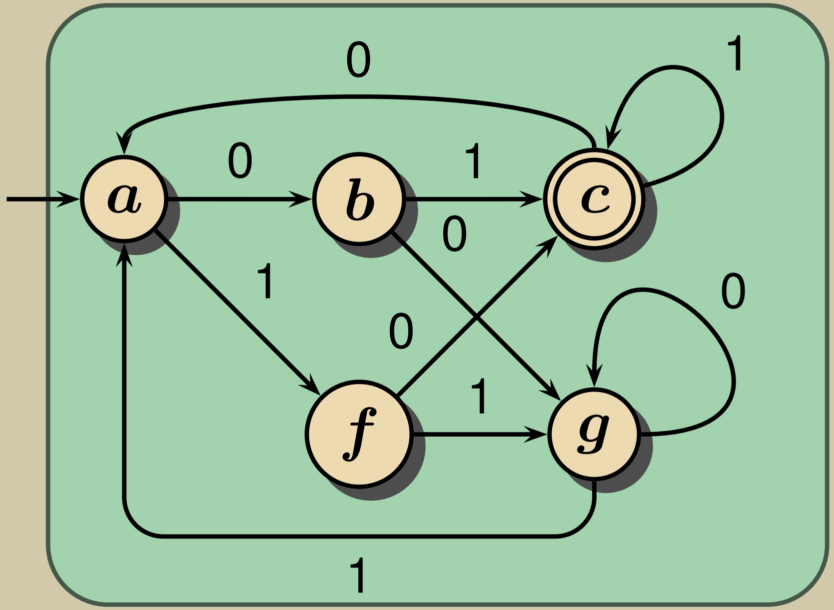
- Laufzeit des Minimierungsalgorithmus:
  1. $\mathcal{O}(|\delta|) = \mathcal{O}(|Q|^2|\Sigma|)$
     ☞ nächstes Kapitel
  2. $\mathcal{O}(|Q|^2|\Sigma|)$ bei geschickter Implementierung
  3. $\mathcal{O}(|Q|^2|\Sigma|)$

  Zusammen: $\mathcal{O}(|Q|^2|\Sigma|)$

Beispiel



Minimaler Automat:



Vom RE zum DFA: vollständig

- Damit kennen wir nun alle Teilschritte von der Spezifikation einer regulären Sprache bis zur Berechnung eines möglichst kleinen endlichen Automaten

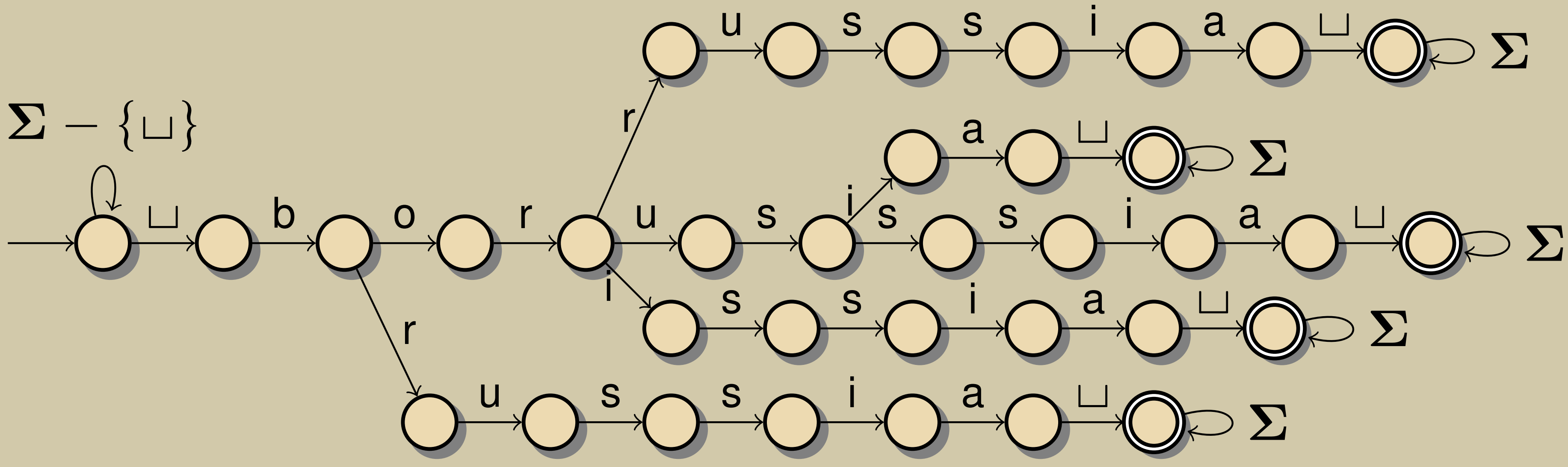
1. Spezifiziere die Sprache durch einen regulären Ausdruck α
2. Wandle α in einen ϵ-NFA $\mathcal{A}_1$ um
3. Wandle $\mathcal{A}_1$ in einen DFA $\mathcal{A}_2$ um
4. Wandle $\mathcal{A}_2$ in einen minimalen DFA $\mathcal{A}_3$ um

✎ Der e-Mail-Adressen-DFA ist übrigens schon minimal...

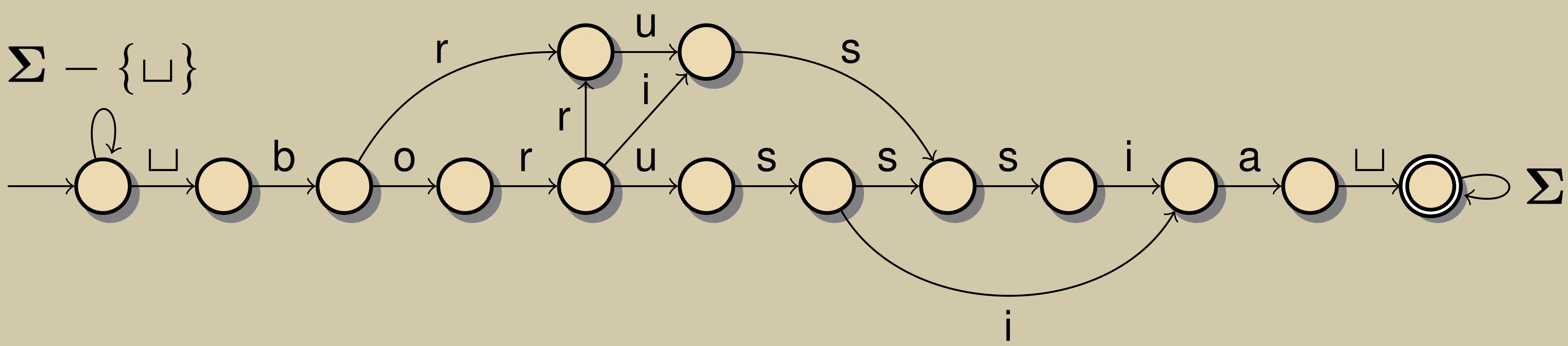
Der Borussia-Newsticker-Automat (3/3)

Beispiel

- Ist der Borussia-Automat minimal?



- Nein, dies ist der minimale DFA:

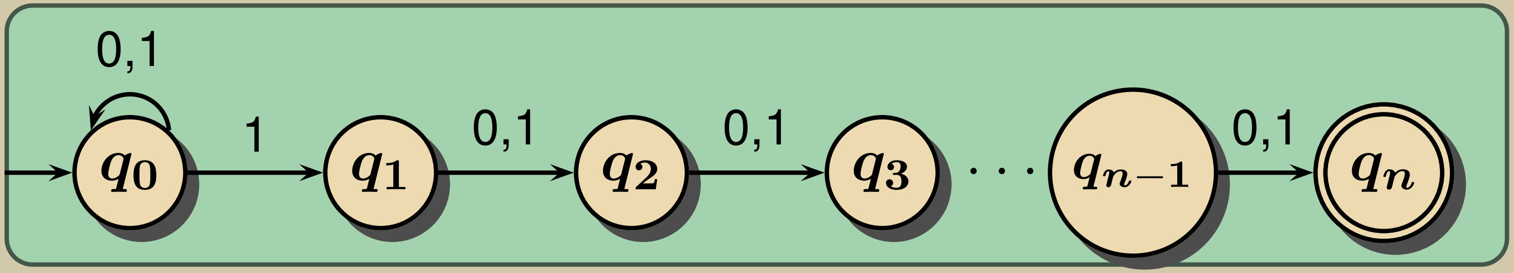


Satz von Myhill und Nerode: weitere Anwendung

- Der Satz von Myhill und Nerode liefert auch eine Methode um die Größe des Minimalautomaten für eine reguläre Sprache zu berechnen:
  - Zähle die Klassen von $\sim_L$

Beispiel

- Wir betrachten wieder die Sprache L_n aller 0-1-Strings, deren n-tes Zeichen von rechts eine 1 ist:



- Es ist leicht zu zeigen, dass zwei Strings x, y genau dann in derselben Äquivalenzklasse von $\sim_{L_n}$ sind, wenn sie dasselbe Suffix der Länge n haben

✎ Dabei werden bei Strings der Länge $< n$ führende Nullen „hinzugedacht“

➡ Es gibt soviele Klassen in $\sim_L$ wie es 0-1-Strings der Länge n gibt

➡ $\sim_{L_n}$ hat 2^n Klassen

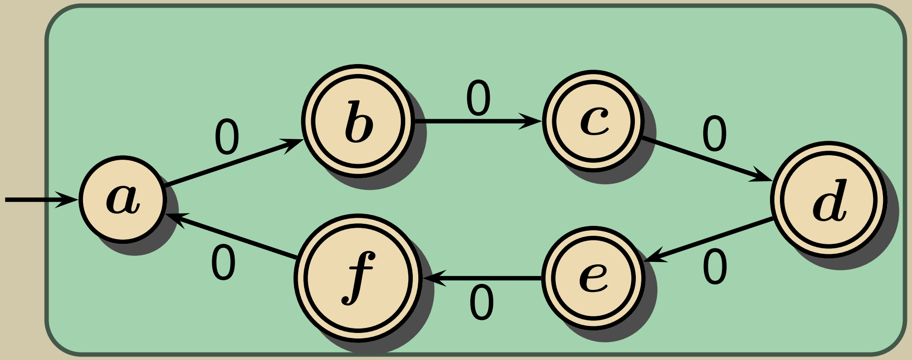
➡ Jeder Automat für L_n hat mindestens 2^n Zustände

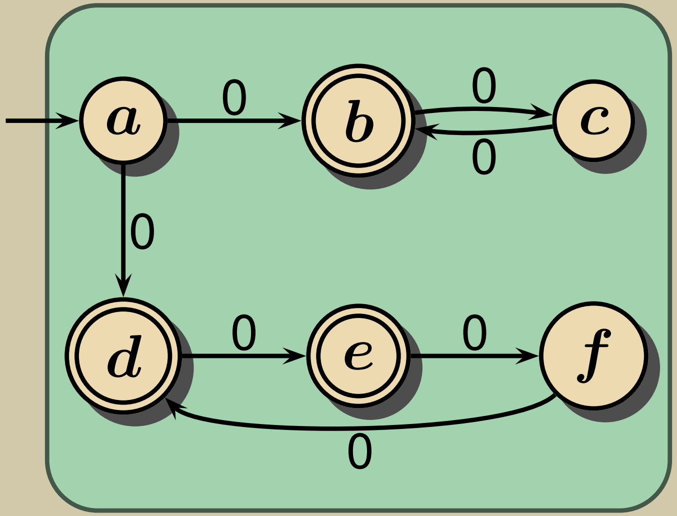
Minimale NFAs

- Es gibt zwar auch zu jedem NFA $\mathcal{A}$ einen kleinsten NFA $\mathcal{A}'$ mit
$$L(\mathcal{A}') = L(\mathcal{A})$$
- Aber der kleinste NFA ist im Allgemeinen nicht bis auf Isomorphie eindeutig

Beispiel

- Die Menge aller Strings der Form 0^n, für die 6 kein Teiler von n ist, hat zwei kleinste NFAs:





- Idee: 6 ist genau dann kein Teiler von n, wenn 2 oder 3 kein Teiler von n ist

Zusammenfassung

- Zu einem gegebenen DFA ist der minimale äquivalente DFA bis auf Isomorphie eindeutig bestimmt und kann mit Hilfe des Markierungsalgorithmus in Zeit $\mathcal{O}(|Q|^2|\Sigma|)$ berechnet werden

- **Literatur:**
  - John R. Myhill. Finite automata and the representation of events. Technical Report WADC TR-57-624, Wright-Paterson Air Force Base, 1957
  - A. Nerode. Linear automaton transformations. *Proc. Amer. Math. Soc.*, 9:541–544, 1958

Minimaler Automat: Eindeutigkeit (3/3)

Lemma 4.3

- Ist $\mathcal{A}$ ein Automat für eine Sprache L, der die selbe Anzahl von Zuständen wie $\mathcal{A}_L$ hat, so gilt: $\mathcal{A} \cong \mathcal{A}_L$

Beweisidee

- Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein solcher Automat
- $\mathcal{A}$ minimal $\Rightarrow$ in $\mathcal{A}$ sind alle Zustände erreichbar

➡ für jeden Zustand q von $\mathcal{A}$ gibt es einen String w_q mit $\delta^*(s, w_q) = q$

- Wir definieren eine Abbildung π durch: $\pi(q) \stackrel{\text{def}}{=} [w_q]$
- **Behauptung:** π ist ein Isomorphismus von $\mathcal{A}$ auf $\mathcal{A}_L$

Beweisdetails

(1) $\pi(s) = [\epsilon]$ ✓

(2) $q \in F \iff w_q \in L \iff$ $\pi(q) = [w_q]$ ist akzeptierender Zustand von $\mathcal{A}_L$

(3) Für $q \in Q, \sigma \in \Sigma$ gilt:

$$\begin{aligned}\pi(\delta(q,\sigma)) &= \pi(\delta(\delta^*(s, w_q), \sigma))\\ &= \pi(\delta^*(s, w_q\sigma))\\ &= [w_q\sigma]\\ &= \delta'([w_q], \sigma)\\ &= \delta'(\pi(q), \sigma)\end{aligned}$$

✎ δ' bezeichnet die Überführungsfunktion von $\mathcal{A}_L$

- π ist bijektiv, denn:
  - Aus dem Beweis von Satz 4.1 folgt: $\sim_{\mathcal{A}}$ ist eine Verfeinerung von $\sim_L$
  - Da $\sim_{\mathcal{A}}$ und $\sim_L$ gleich viele Klassen haben gilt also: $\sim_{\mathcal{A}} = \sim_L$

  ➡ π ist eine Bijektion

- Also ist π ein Isomorphismus und es folgt: $\mathcal{A} \cong \mathcal{A}_L$