

Theoretische Informatik I

Einheit 2.5

Eigenschaften regulärer Sprachen



1. Abschlusseigenschaften
2. Prüfen von Eigenschaften
3. Wann sind Sprachen nicht regulär?

WICHTIGE EIGENSCHAFTEN FORMALER SPRACHEN

- **Abschlusseigenschaften**

- Wie können Sprachen elegant zusammengesetzt werden?
- Erlaubt schematische Komposition von Sprachbausteinen

- **Entscheidbarkeitsfragen**

- Kann man bestimmte Eigenschaften automatisch testen?
- **Wortproblem** (Zugehörigkeit eines Wortes zur Sprache)
- **Vergleiche** zwischen Sprachen (nichtleer, Teilmenge, gleich, ...)

- **Grenzen einer Sprachklasse**

- Wie einfach strukturiert müssen die Sprachen der Klasse sein?
- Welche Sprachen gehören nicht zur Klasse?

**Aus theoretischer Sicht sind das
die wirklich interessanten Fragen**

ABSCHLUSSEIGENSCHAFTEN, WOZU?

Zeige, dass bestimmte Operationen auf regulären Sprachen wieder zu regulären Sprachen führen

- **Wiederverwendung von “Sprachmodulen”**
 - Schematische Komposition von
 - Grammatiken zur Erzeugung von Sprachen
 - Automaten zur Erkennung von Sprachen
 - Regulären Ausdrücken
- **Schematische Konstruktion ist effektiver**
 - Fehlerfreier Aufbau sehr komplexer Grammatiken / Automaten
 - + Schematische Optimierung / Minimierung
 - Konstruktion “von Hand” oft fehleranfällig
- **Beispiel: Literale einer Programmiersprache**
 - Bilde Automaten für **Tokenklassen**: Zahlen, Bezeichner, Schlüsselwörter, ...
 - Konstruktion liefert Automaten für alle Arten von Literalen

ABSCHLUSSEIGENSCHAFTEN, PRÄZISIERT

Zeige: L_1, L_2 regulär $\Rightarrow L_1 \text{ op } L_2$ regulär

• Es gilt Abgeschlossenheit unter neun Operationen

- Die **Vereinigung** zweier regulärer Sprachen ist regulär $L_1 \cup L_2$
- Das **Komplement** einer regulären Sprache ist regulär \bar{L}
- Der **Durchschnitt** zweier regulärer Sprachen ist regulär $L_1 \cap L_2$
- Die **Differenz** zweier regulärer Sprachen ist regulär $L_1 - L_2$
- Die **Spiegelung** einer regulären Sprache ist regulär L^R
- Die **Hülle** einer regulären Sprache ist regulär L^*
- Die **Verkettung** zweier regulärer Sprachen ist regulär $L_1 \circ L_2$
- Das **Bild** einer regulären Sprache **unter Homomorphismen** ist regulär $h(L)$
- Das **Urbild** ... " " ... **unter Homomorphismen** ist regulär $h^{-1}(L)$

• Nachweis durch Verwendung aller Modelle

- DEA, (ϵ -)NEA, reguläre Ausdrücke, Typ-3 Grammatiken
- Modelle sind ineinander umwandelbar – **wähle das passendste**

Beweisführung mit regulären Ausdrücken

- **L_1, L_2 regulär $\Rightarrow L_1 \cup L_2$ regulär**

L_1, L_2 regulär

\Rightarrow Es gibt reguläre Ausdrücke E_1, E_2 mit $L_1 = L(E_1), L_2 = L(E_2)$

$\Rightarrow L_1 \cup L_2 = L(E_1) \cup L(E_2) = L(E_1 + E_2)$ **regulär**

- **L_1, L_2 regulär $\Rightarrow L_1 \circ L_2$ regulär**

L_1, L_2 regulär

\Rightarrow Es gibt reguläre Ausdrücke E_1, E_2 mit $L_1 = L(E_1), L_2 = L(E_2)$

$\Rightarrow L_1 \circ L_2 = L(E_1) \circ L(E_2) = L(E_1 \circ E_2)$ **regulär**

- **L regulär $\Rightarrow L^*$ regulär**

L regulär

\Rightarrow Es gibt einen regulären Ausdruck E mit $L = L(E)$

$\Rightarrow L^* = (L(E))^* = L(E^*)$ **regulär**

ABSCHLUSS UNTER KOMPLEMENTBILDUNG

Beweisführung mit endlichen Automaten

- **L regulär $\Rightarrow \overline{L}$ regulär**

Komplementiere akzeptierende Zustände des erkennenden Automaten

L regulär

\Rightarrow Es gibt einen DEA $A = (Q, \Sigma, \delta, q_0, F)$ mit $L = L(A)$

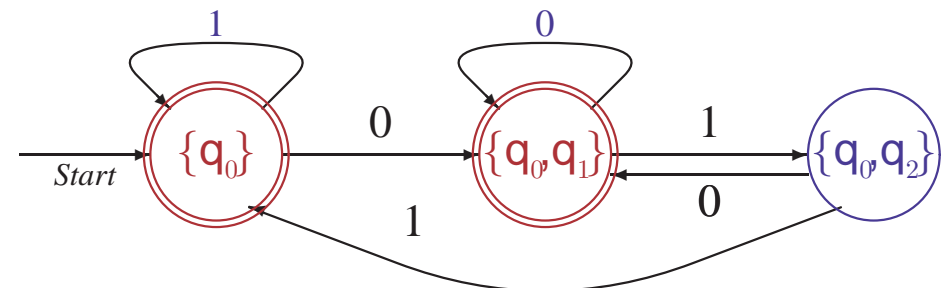
$\Rightarrow \overline{L} = \overline{L(A)} = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \notin F\} = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in Q - F\}$
 $= L((Q, \Sigma, \delta, q_0, Q - F))$ **regulär**

- **Beispiel: Komplementierung von $(0+1)^*01$**

– Zugehöriger DEA

– Komplementautomat erkennt
Wörter die **nicht** mit 01 enden

– **Regulärer Ausdruck durch Zustandseliminationsverfahren erzeugbar**



ABSCHLUSS UNTER DURCHSCHNITT UND DIFFERENZ

- **Einfache mathematische Beweise**

L_1, L_2 regulär $\Rightarrow L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ regulär

L_1, L_2 regulär $\Rightarrow L_1 - L_2 = L_1 \cap \overline{L_2}$ regulär

- **Produktkonstruktion auf endlichen Automaten**

Simultane Abarbeitung von Wörtern in beiden Automaten

L_1, L_2 regulär

\Rightarrow Es gibt DEAs $A_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1)$

und $A_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$

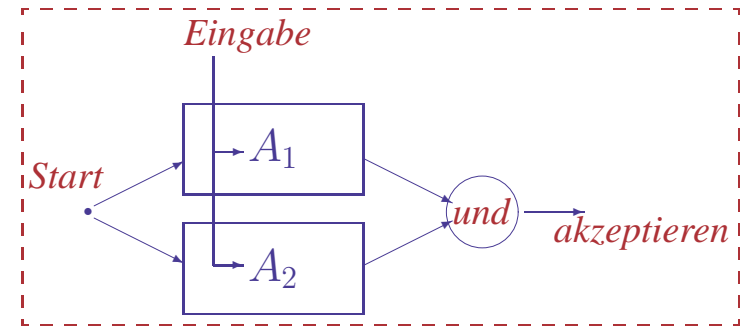
mit $L_1 = L(A_1), L_2 = L(A_2)$

$\Rightarrow L_1 \cap L_2 = \{w \in \Sigma^* \mid \hat{\delta}_1(q_{0,1}, w) \in F_1 \wedge \hat{\delta}_2(q_{0,2}, w) \in F_2\}$
 $= \{w \in \Sigma^* \mid (\hat{\delta}_1(q_{0,1}, w), \hat{\delta}_2(q_{0,2}, w)) \in F_1 \times F_2\}$

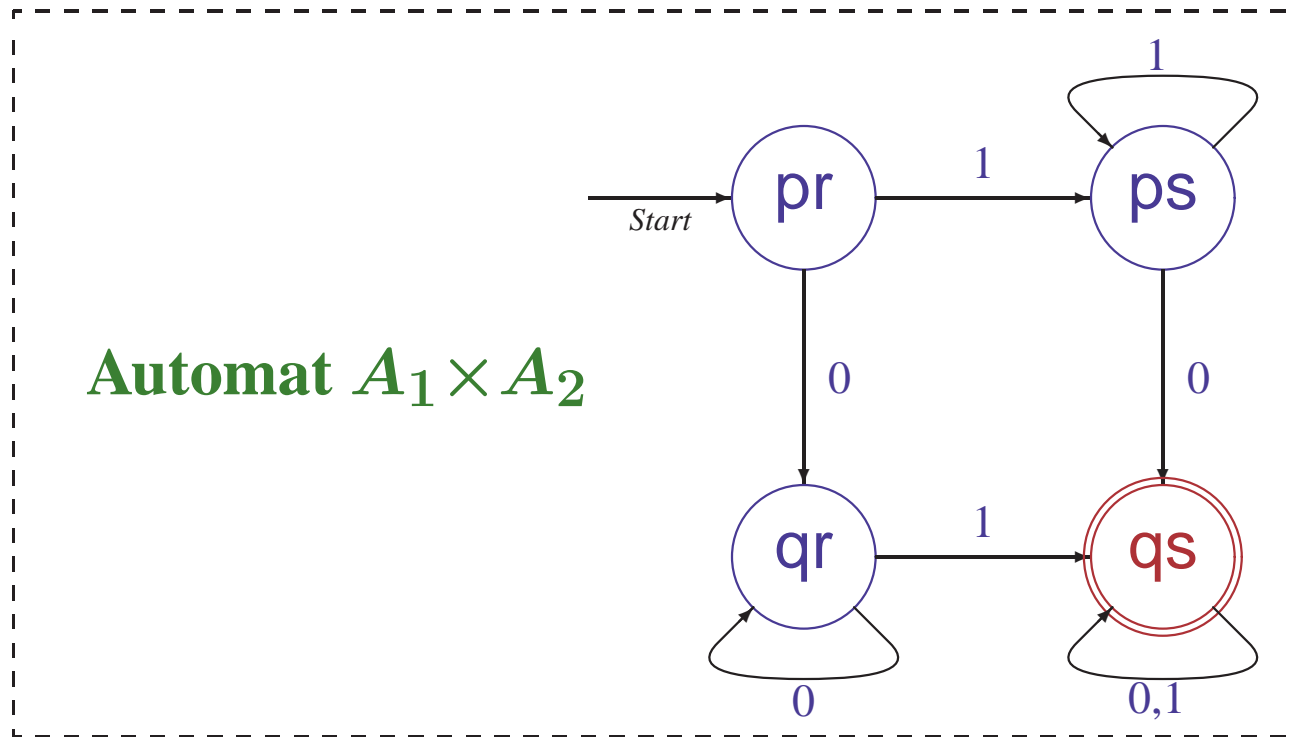
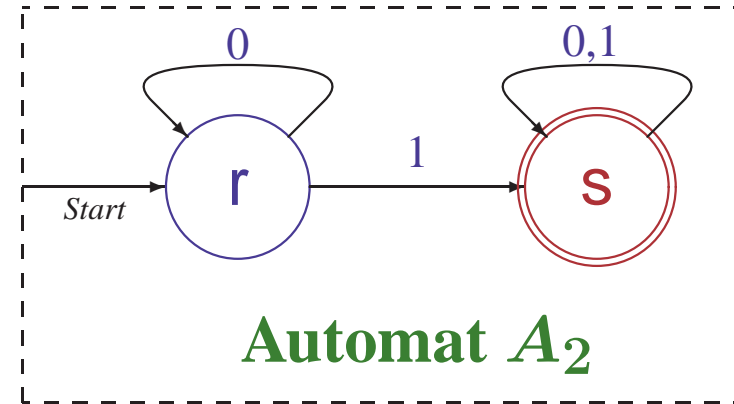
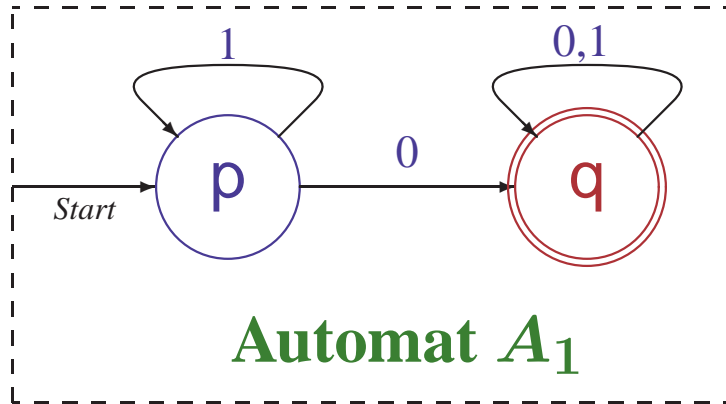
Konstruiere $A = (Q_1 \times Q_2, \Sigma, \delta, (q_{0,1}, q_{0,2}), F_1 \times F_2)$

mit $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$ für $p \in Q_1, q \in Q_2, a \in \Sigma$

$\Rightarrow L_1 \cap L_2 = L(A)$ regulär



PRODUKTKONSTRUKTION AM BEISPIEL



ABSCHLUSS UNTER SPIEGELUNG

L regulär $\Rightarrow L^R = \{w_n \dots w_1 \mid w_1 \dots w_n \in L\}$ regulär

• Beweisführung mit Automaten

- Bilde **Umkehrautomaten** zu $A = (Q, \Sigma, \delta, q_0, F)$ mit $L = L(A)$
 - Umkehrung der Pfeile im Diagramm: $\delta^R(q, a) = \{q' \mid \delta(q', a) = q\}$
 - q_0 wird zum akzeptierenden Zustand: $F^R = \{q_0\}$
 - Neuer Startzustand q_0^R mit ϵ -Übergängen zu allen $q \in F$

• Induktiver Beweis mit regulären Ausdrücken

Sei $L = L(E)$ für einen regulären Ausdruck

- Für $E \in \{\emptyset, \epsilon, a\}$ ist $L^R = L = L(E)$ regulär
- Für $E = E_1 + E_2$ ist $L^R = (L(E_1) \cup L(E_2))^R = L(E_1)^R \cup L(E_2)^R$ regulär
- Für $E = E_1 \circ E_2$ ist $L^R = (L(E_1) \circ L(E_2))^R = L(E_2)^R \circ L(E_1)^R$ regulär
- Für $E = E_1^*$ ist $L^R = L(E_1^*)^R = (L(E_1)^R)^*$ regulär

• Beispiel: Spiegelung von $L((0+1)0^*)$

- $L^R = L((0^*)^R (0+1)^R) = L((0^R)^* (0^R+1^R)) = L(0^*(0+1))$

BILD UND URBILD UNTER HOMOMORPHISMEN

- **Homomorphismus** $h: \Sigma^* \rightarrow \Sigma'^*$

- Funktion, die eindeutig durch Verhalten auf Symbolen definiert ist
- $h: \Sigma^* \rightarrow \Sigma'^*$ ist Homomorphismus, wenn $h(v_1..v_n) = h(v_1)..h(v_n)$ für $v_i \in \Sigma$
- Für $h_1(a) = 01$, $h_1(b) = 0101$ muß $h_1(aa) = 0101$, $h_1(ab) = 010101$ sein
- Homomorphismen sind mit endlichen (Ein-/Ausgabe) Automaten berechenbar

- **Bild einer Menge L unter einer Funktion h**

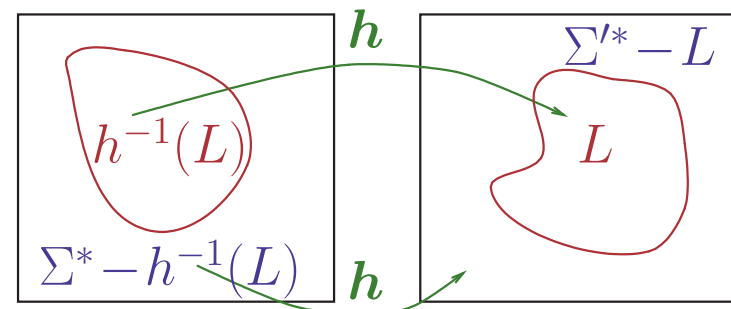
- Menge aller Funktionswerte von h bei Eingaben aus L
- $h(L) = \{h(w) \mid w \in L\} \subseteq \Sigma'^*$
- z.B. $h_1(\{ab, aa, b\}) = \{0101, 010101\}$, $h_1(\{a, b\}^*) = \{01\}^*$

- **Urbild einer Menge L unter einer beliebigen (!) Funktion h**

- Menge aller Eingaben, deren Funktionswerte in L liegen
- $h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$
- z.B. $h_1^{-1}(\{0101\}) = \{aa, b\}$,

$$h_1^{-1}(\{0101, 11, 1\}) = \{aa, b\},$$

$$h_1^{-1}(L((0101)^*)) = \{w \in \{a, b\}^* \mid \#_a(w) \text{ gerade}\}$$



ABSCHLUSSEIGENSCHAFT: BILD UNTER HOMOMORPHISMEN

$L \subseteq \Sigma^*$ regulär, $h: \Sigma^* \rightarrow \Sigma'^*$ Homomorphismus $\Rightarrow h(L)$ regulär

Beweis mit Grammatiken

L regulär

\Rightarrow Es gibt eine Typ-3 Grammatik $G = (V, \Sigma, P, S)$ mit $L = L(G)$

$\Rightarrow h(L) = h(L(G)) = \{h(v_1)..h(v_n) \in \Sigma'^* \mid S \xrightarrow{*} v_1..v_n\}$

Für $A \rightarrow a B \in P$ erzeuge Regeln $A \rightarrow a_1 B_1, B_1 \rightarrow a_2 B_2, \dots, B_{k-1} \rightarrow a_k B$,
wobei $h(a) = a_1..a_k$ und alle B_i neue Hilfsvariablen

Sei P_h die Menge dieser Regeln, V_h die Menge ihrer Hilfsvariablen

Für $G_h = (V_h, \Sigma', P_h, S)$ gilt $A \rightarrow a B \in P \Leftrightarrow A \xrightarrow{*}_{G_h} h(a) B$

und $S \xrightarrow{*}_G v_1..v_n \Leftrightarrow S \xrightarrow{*}_{G_h} h(v_1)..h(v_n)$

$\Rightarrow h(L) = \{h(v_1)..h(v_n) \in \Sigma'^* \mid S \xrightarrow{*}_{G_h} h(v_1)..h(v_n)\} = L(G_h)$ **regulär**

– Sonderbehandlung für $h(a)=\epsilon$ erforderlich, da Regel $A \rightarrow B$ unzulässig

Beweis mit regulären Ausdrücken in Hopcroft, Motwani, Ullman §4.2.3

ABSCHLUSS: URBILD UNTER HOMOMORPHISMEN

$L \subseteq \Sigma'^*$ regulär, $h: \Sigma^* \rightarrow \Sigma'^*$ Homomorphismus $\Rightarrow h^{-1}(L)$ regulär

Beweis mit endlichen Automaten

Berechnung von h vor Abarbeitung der Wörter im Automaten

L regulär

\Rightarrow Es gibt einen DEA $A = (Q, \Sigma', \delta, q_0, F)$

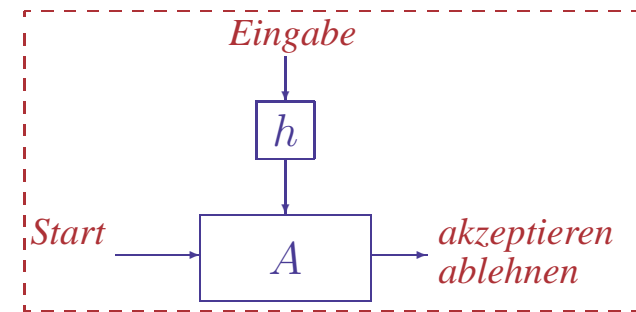
mit $L = L(A) = \{v \in \Sigma'^* \mid \hat{\delta}(q_0, v) \in F\}$

$\Rightarrow h^{-1}(L) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, h(w)) \in F\}$

Konstruiere $A_h = (Q, \Sigma, \delta_h, q_0, F)$ mit $\delta_h(q, a) = \hat{\delta}(q, h(a))$

Dann gilt $\hat{\delta}_h(q, w) = \hat{\delta}(q, h(w))$ für alle $q \in Q$ und $w \in \Sigma^*$

$\Rightarrow h^{-1}(L) = \{w \in \Sigma^* \mid \hat{\delta}_h(q_0, w) \in F\} = L(A_h)$ regulär



TESTS FÜR EIGENSCHAFTEN REGULÄRER SPRACHEN

- **Welche Eigenschaften sind automatisch prüfbar?**

- Ist die Sprache eines Automaten leer?
- **Zugehörigkeit**: Ist ein Wort w Element der Sprache eines Automaten?
- **Äquivalenz**: Beschreiben zwei Automaten dieselbe Sprache?

Gleiche Fragestellung für Grammatiken und reguläre Ausdrücke

- **Wechsel der Repräsentation ist effektiv**

NEA \mapsto DEA: Teilmengenkonstruktion (exponentielle Aufblähung möglich)

ϵ -NEA \mapsto DEA: Hüllenbildung + Teilmengenkonstruktion

DEA \mapsto ϵ -NEA/NEA: Modifikation der Präsentation (Mengenklammern)

DEA \mapsto RA: Zustandselimination (oder R_{ij}^k -Methode)

RA \mapsto ϵ -NEA: induktive Konstruktion von Automaten

DEA \mapsto Typ-3 Grammatik: Regeln für Überführungsschritte einführen

Typ-3 Grammatik \mapsto NEA: Überführungstabelle codiert Regeln

- **Es reicht, Tests für ein Modell zu beschreiben**

PRÜFE, OB EINE REGULÄRE SPRACHE LEER IST

• Nichttriviales Problem

- Automaten: Gibt es überhaupt einen akzeptierenden Pfad?
- Reguläre Ausdrücke: Wird mindestens ein einziges Wort charakterisiert?
- Grammatiken: Wird überhaupt ein Wort aus dem Startzustand erzeugt?

• Erreichbarkeitstest für DEA $A = (Q, \Sigma, \delta, q_0, F)$

- Wegen $\hat{\delta}(q_0, \epsilon) = q_0$ ist q_0 in 0 Schritten erreichbar
- q in k Schritten erreichbar, $\delta(q, a) = q' \Rightarrow q'$ in $k+1$ Schritten erreichbar
- $L(A) = \emptyset \Leftrightarrow$ kein $q \in F$ in $|Q|$ Schritten erreichbar

• Induktive Analyse für reguläre Ausdrücke

- $L(\emptyset) = \emptyset$, $L(\epsilon) \neq \emptyset$, $L(a) \neq \emptyset$
- $L((E)) = \emptyset \Leftrightarrow L(E) = \emptyset$ keine Änderung
- $L(E+F) = \emptyset \Leftrightarrow L(E) = \emptyset \wedge L(F) = \emptyset$ Vereinigung von Elementen
- $L(E \circ F) = \emptyset \Leftrightarrow L(E) = \emptyset \vee L(F) = \emptyset$ Elemente beider Sprachen nötig
- $L(E^*) \neq \emptyset$, ϵ gehört immer zu $L(E^*)$

TEST AUF ZUGEHÖRIGKEIT (WORTPROBLEM)

- **Unterschiedlich schwierig je nach Repräsentation**
 - Automaten: Gibt es einen akzeptierenden Pfad für das Wort w ?
 - Reguläre Ausdrücke: Wird w von der Charakterisierung erfasst?
 - Grammatiken: Kann w aus dem Startzustand erzeugt werden?
- **Abarbeitung durch DEA $A = (Q, \Sigma, \delta, q_0, F)$**
 - Bestimme $q := \hat{\delta}(q_0, w)$ und teste $q \in F$
 - Maximal $|w| + |F|$ Arbeitsschritte

**Test für andere Repräsentationen
durch Umwandlung in DEA**

TEST AUF ÄQUIVALENZ VON SPRACHEN

- **Wann sind zwei reguläre Sprachen gleich?**
 - Nichttrivial, da **Beschreibungsformen** sehr verschieden sein können
 - Verschiedene Automaten, Grammatiken, Ausdrücke, Mischformen, ...
- **Gibt es eine “kanonische” Repräsentation?**
 - z.B. · Transformiere alles in deterministische endliche Automaten
 - Erzeuge **Standardversion** mit kleinstmöglicher Anzahl von Zuständen
 - Äquivalenztest prüft dann, ob der **gleiche Standardautomat** erzeugt wird
- **Wie standardisiert man Automaten?**
 - **Entferne Zustände**, die vom Startzustand **unerreichbar** sind
 - **Fasse Zustände zusammen**, die für alle Wörter “äquivalent” sind
 - Es führen exakt dieselben Wörter zu akzeptierenden Zuständen
 - Ergibt **minimalen äquivalenten Automaten**

ÄQUIVALENZTEST FÜR ZUSTÄNDE

- **Äquivalenz der Zustände p und q ($p \cong q$)**

- Für alle Wörter $w \in \Sigma^*$ gilt $\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$
- Die Wörter müssen **nicht zum gleichen Zustand** führen

- **Positives Prüfverfahren schwierig**

- Man muss **alle** Wörter überprüfen, die von einem Zustand ausgehen
- Man kann sich auf Wörter der maximalen Länge $|Q|$ beschränken
- Besser: Nichtäquivalente (**unterscheidbare**) Zustände identifizieren

- **Methodik: Table-Filling Algorithmus**

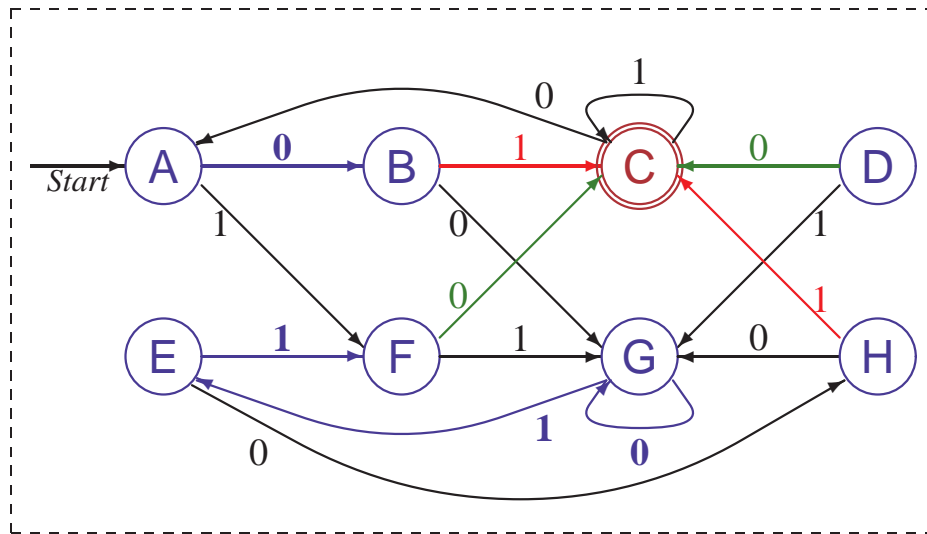
Markiere Unterscheidbarkeit von Zuständen in Tabelle

- Start: $p \not\cong q$, falls $p \in F$ und $q \notin F$
- Iteration: $p \not\cong q$, falls $\delta(p, a) \not\cong \delta(q, a)$ für ein $a \in \Sigma$

In jeder Iteration werden nur noch ungeklärte Paare überprüft

Nach **maximal $|Q|$ Iterationen** sind alle Unterschiede bestimmt

ÄQUIVALENZTEST AM BEISPIEL



	A	B	C	D	E	F	G	H
A	■	×	×	×		×	×	×
B	■	■	×	×	×	×	×	
C	■	■	■	×	×	×	×	×
D	■	■	■	■	×		×	×
E	■	■	■	■	■	×	×	×
F	■	■	■	■	■	■	×	×
G	■	■	■	■	■	■	■	×
H	■	■	■	■	■	■	■	■

Tabelle der Unterschiede

1. Unterscheide **akzeptierende Zustände (C)** von allen anderen
 - 2a. **Eingabesymbol 0:** Nur **D** und **F** führen zu akzeptierenden Zuständen
 - 2b. **Eingabesymbol 1:** Nur **B** und **H** führen zu akzeptierenden Zuständen
 3. Überprüfe Nachfolger von $\{A,E\}$, $\{A,G\}$, $\{B,H\}$, $\{D,F\}$ und $\{E,G\}$.
 4. Überprüfung von $\{A,E\}$, $\{B,H\}$ und $\{D,F\}$ gibt keine Unterschiede
- Äquivalenzklassen sind $\{A,E\}$, $\{B,H\}$, $\{D,F\}$, $\{C\}$ und $\{G\}$**

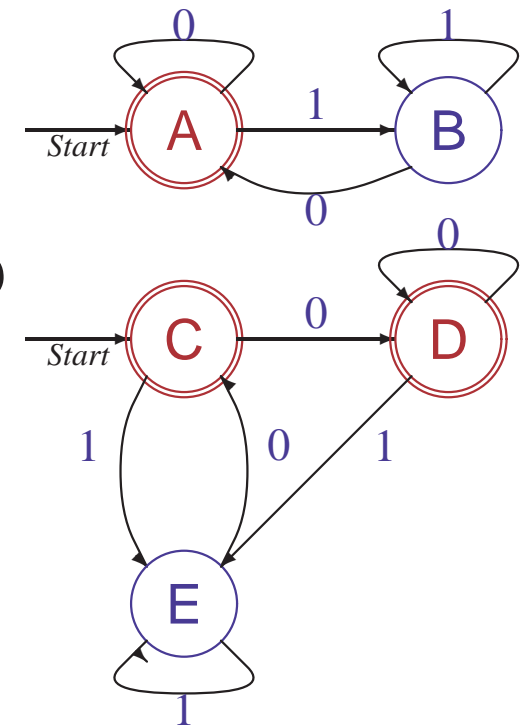
ÄQUIVALENZTEST FÜR SPRACHEN

• Prüfverfahren

- Standardisiere Beschreibungsform in zwei disjunkte DEAs A_1 und A_2
- Vereinige Automaten zu $A = (Q_1 \cup Q_2 \cup \{q'\}, \Sigma, \delta_1 \cup \delta_2, q', F_1 \cup F_2)$
 A enthält A_1 und A_2 als **unabhängige** Teile
- Bilde Äquivalenzklassen von A
und **teste ob $q_{0,1}$ und $q_{0,2}$ äquivalent** sind

• Zwei DEAs für $L(\epsilon + (0 + 1)^*0)$

- Äquivalenzklassen sind $\{A,C,D\}$ (alle Endzustände)
und $\{B,E\}$ (alle Nicht-Endzustände)
- **Da A und C äquivalent sind,**
sind die Automaten äquivalent



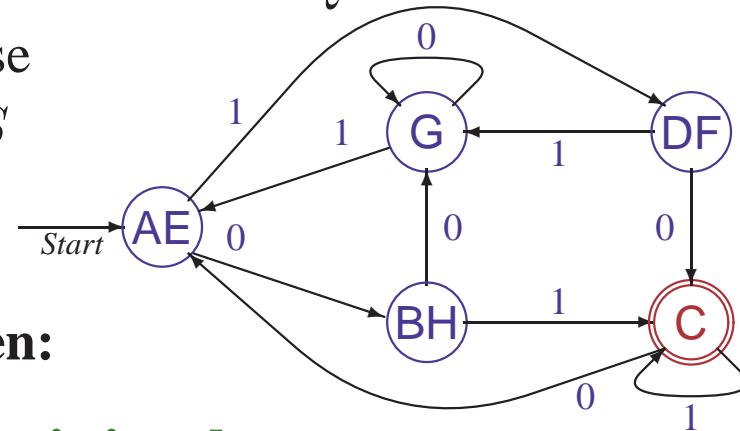
Konstruiere äquivalenten DEA mit minimaler Menge von Zuständen

- **Entferne überflüssige Zustände**

- q ist **überflüssig**, wenn $\hat{\delta}(q_0, w) \neq q$ für alle Wörter $w \in \Sigma^*$
- Reduziere Q zu Menge der erreichbaren Zustände (Verfahren auf Folie 13)

- **Fasse äquivalente Zustände zusammen**

- Bestimme Menge der Äquivalenzklassen von Q
- Setze Q' als Menge der Äquivalenzklassen von Q
- Setze $\delta'(S, a)$ als Äquivalenzklasse von $\delta(q, a)$ für ein beliebiges $q \in S$
Wohldefiniert, da alle Nachfolger äquivalenter Zustände äquivalent sind



Anwendung auf Beispielautomaten:

- **Resultierender Automat ist minimal**

● Automaten teilen Sprachen in Äquivalenzklassen

- Wörter, die zum gleichen Zustand führen, sind ununterscheidbar
- Wörter, die zu äquivalenten Zuständen führen, sind ununterscheidbar

Jede Fortsetzung der Wörter führt zum “gleichen” Ergebnis

$\hat{\delta}(q_0, u) \cong \hat{\delta}(q_0, v)$ bedeutet $\hat{\delta}(q_0, u w) \in F \Leftrightarrow \hat{\delta}(q_0, v w) \in F$ für alle $w \in \Sigma^*$

● Äquivalenzklassen hängen nur von der Sprache ab

- Für $L \subseteq \Sigma^*$ definiere Äquivalenzrelation \sim_L auf Σ^* :

• $u \sim_L v \equiv u w \in L \Leftrightarrow v w \in L$ gilt für alle $w \in \Sigma^*$

- Die Äquivalenzklasse eines Wortes v ist $[v]_L = \{u \in \Sigma^* \mid u \sim_L v\}$

- Σ^*/L bezeichnet die Menge der Äquivalenzklassen modulo \sim_L

• Für $L = \{0^n 1^m \mid n, m \in \mathbb{N}\}$ ist $\Sigma^*/L = \{[\epsilon]_L, [1]_L, [10]_L\}$

• Für $L = \{0^n 1^n \mid n \in \mathbb{N}\}$

ist $\Sigma^*/L = \{[\epsilon]_L, [0]_L, [1]_L, [00]_L, [01]_L, [000]_L, [001]_L, \dots\}$

Reguläre Sprachen haben nur endlich viele Äquivalenzklassen

DER SATZ VON MYHILL/NERODE

Eine Sprache L ist regulär, g.d.w Σ^*/L endlich ist

Beweis

\Rightarrow : Es sei L eine reguläre Sprache

Dann gibt es einen minimalen DEA $A = (Q, \Sigma, \delta, q_0, F)$ mit $L = L(A)$

Da A minimal ist, gilt für beliebige Wörter $u, v \in \Sigma^*$

$$\begin{aligned} \hat{\delta}(q_0, u) = \hat{\delta}(q_0, v) &\Leftrightarrow (\hat{\delta}(q_0, u w) \in F \Leftrightarrow \hat{\delta}(q_0, v w) \in F) \text{ für alle } w \in \Sigma^* \\ &\Leftrightarrow (u w \in L \Leftrightarrow v w \in L) \text{ für alle } w \in \Sigma^* \Leftrightarrow u \sim_L v \end{aligned}$$

Damit ist $|\Sigma^*/L|$ (der **Index von L**) gleich der Anzahl der Zustände in A

\Leftarrow : Es sei Σ^*/L endlich.

Konstruiere einen DEA $A = (\Sigma^*/L, \Sigma, \delta, [\epsilon]_L, F)$

mit $\delta([u]_L, a) = [u a]_L$ für alle $a \in \Sigma$ und $F = \{[v]_L \mid v \in L\}$

δ ist wohldefiniert, weil $u a \sim_L v a$ für alle $a \in \Sigma$ gilt, wenn $u \sim_L v$

und es gilt $w \in L(A) \Leftrightarrow \hat{\delta}([\epsilon]_L, w) \in F \Leftrightarrow [w]_L \in F \Leftrightarrow w \in L$

Wie zeigt man, dass eine Sprache L nicht regulär ist?

● Direkter Nachweis

- Zeige, dass kein endlicher Automat genau die Wörter von L erkennt
- Sprache muss **unendlich** sein und **komplizierte Struktur** haben
(Anzahl der Äquivalenzklassen muss unendlich sein)
- Technisches Hilfsmittel: **Pumping Lemma**

● Verwendung der **Abschlusseigenschaften**

- Zeige, dass Regularität von L dazu führen würde, dass eine als nichtregulär bekannte Sprache regulär sein müsste
- Häufige Technik: **(Ur-)bild unter Homomorphismen**

DAS PUMPING LEMMA FÜR REGULÄRE SPRACHEN

- **Warum ist $\{0^n 1^n \mid n \in \mathbb{N}\}$ nicht regulär?**

- Ein DEA muss alle Nullen beim Abarbeiten zählen und dann vergleichen
- Für $n > |Q|$ muss ein Zustand von A doppelt benutzt worden sein
- Eine δ -Schleife mit k Zuständen bedeutet, dass A auch $0^{n+k} 1^n$ akzeptiert

- **Allgemeine Version: Pumping Lemma**

Für jede reguläre Sprache $L \in \mathcal{L}_3$ gibt es eine Zahl $n \in \mathbb{N}$, so dass jedes Wort $w \in L$ mit Länge $|w| \geq n$ zerlegt werden kann in $w = x y z$ mit den Eigenschaften

(1) $y \neq \epsilon$,

(2) $|x y| \leq n$ und

(3) für alle $k \in \mathbb{N}$ ist $x y^k z \in L$

- **Aussage ist wechselseitig konstruktiv**

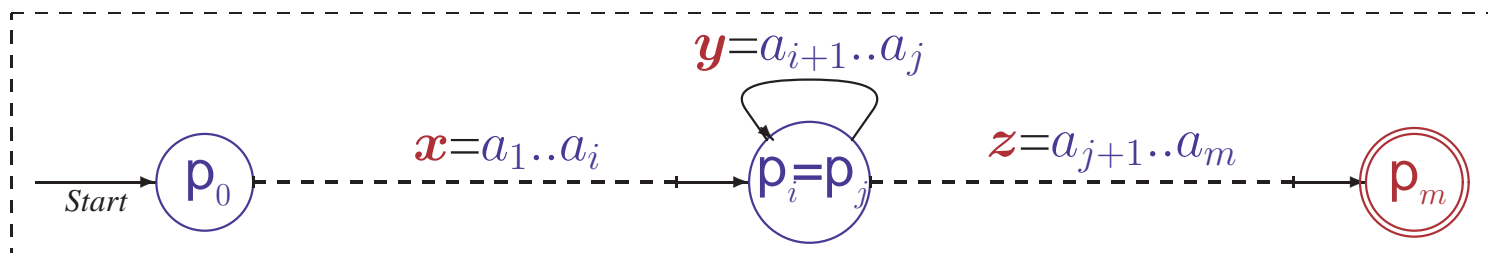
- Die Zahl n kann zu jeder regulären Sprache L bestimmt werden
- Die Zerlegung $w = x y z$ kann zu jedem Wort $w \in L$ bestimmt werden

BEWEIS DES PUMPING LEMMAS

Für jede Sprache $L \in \mathcal{L}_3$ gibt es ein $n \in \mathbb{N}$, so dass jedes $w \in L$ mit $|w| \geq n$ zerlegbar ist in $w = x y z$ mit den Eigenschaften
(1) $y \neq \epsilon$, (2) $|x y| \leq n$ und (3) für alle $k \in \mathbb{N}$ ist $x y^k z \in L$

• Beweis mit Automaten

- Sei L regulär und $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA mit $L = L(A)$
- Wähle $n = |Q|$. Betrachte $w = a_1 \dots a_m$ mit $|w| \geq n$ und $p_i := \hat{\delta}(q_0, a_1 \dots a_i)$
- Dann gibt es i, j mit $0 \leq i < j \leq n$ und $p_i = p_j$ (Schubfachprinzip)
- Zerlege w in $w = x y z$ mit $x = a_1 \dots a_i$, $y = a_{i+1} \dots a_j$ und $z = a_{j+1} \dots a_m$



- Per Konstruktion gilt $y \neq \epsilon$, $|x y| \leq n$ und $\hat{\delta}(p_i, y^k) = p_i$ für alle $k \in \mathbb{N}$
- Also $\hat{\delta}(q_0, x y^k z) = \hat{\delta}(p_i, y^k z) = \hat{\delta}(p_i, y z) = \hat{\delta}(q_0, x y z) = \hat{\delta}(q_0, w) \in F$

ANWENDUNGEN DES PUMPING LEMMAS

$L_1 = \{0^m 1^m \mid m \in \mathbb{N}\}$ ist nicht regulär

- **Verwende Kontraposition des Pumping Lemmas**

Eine Sprache L ist *nicht regulär*, wenn es kein $n \in \mathbb{N}$ gibt, so dass jedes $w \in L$ mit $|w| \geq n$ zerlegbar ist in $w = x y z$ mit den Eigenschaften

(1) $y \neq \epsilon$, (2) $|x y| \leq n$ und (3) für alle $k \in \mathbb{N}$ ist $x y^k z \in L$

Umformulierung: Ziehe Negation in die Bedingungen hinein

L ist *nicht regulär*, wenn es für jedes $n \in \mathbb{N}$ ein $w \in L$ mit $|w| \geq n$ gibt so dass für jede Zerlegung $w = x y z$ mit den Eigenschaften

(1) $y \neq \epsilon$ und (2) $|x y| \leq n$ ein $k \in \mathbb{N}$ existiert mit $x y^k z \notin L$

- **Kontrapositionsbeweis für $L_1 \notin \mathcal{L}_3$**

– Sei $n \in \mathbb{N}$ beliebig. Wir wählen $w = 0^m 1^m$ für ein $m > n$

– Sei $w = x y z$ eine beliebige Zerlegung mit $y \neq \epsilon$ und $|x y| \leq n$

Dann gilt $x = 0^i$, $y = 0^j$, $z = 0^{m-i-j} 1^m$ für ein $j \neq 0$ und $i + j \leq n$.

– Wir wählen $k = 0$. Dann ist $x y^0 z = 0^{m-j} 1^m \notin L_1$

– Aufgrund des Pumping Lemmas kann L_1 also nicht regulär sein.

ANWENDUNGEN DES PUMPING LEMMAS II

$$L_2 = \{w \in \{1\}^* \mid |w| \text{ ist Primzahl}\} \notin \mathcal{L}_3$$

• Beweis

- Sei $n \in \mathbb{N}$ beliebig.
- Wir wählen $w = 1^p$ für eine Primzahl $p > n + 1$
- Sei $w = xyz$ eine beliebige Zerlegung mit $y \neq \epsilon$ und $|xy| \leq n$
Dann gilt $x = 1^i$, $y = 1^j$, $z = 1^{p-i-j}$ für ein $j \neq 0$ und $i + j \leq n$.
- Wir wählen $k = p - j$.
Dann ist $xy^kz = 1^i 1^{j(p-j)} 1^{p-i-j} = 1^{i+j(p-j)+p-i-j} = 1^{(j+1)(p-j)} \notin L_2$
- Aufgrund des Pumping Lemmas kann L_2 also nicht regulär sein.

NACHWEIS VON $L \notin \mathcal{L}_3$ MIT ABSCHLUSSEIGENSCHAFTEN

- **Anwendung des Pumping Lemmas ist oft mühsam**

- Beweis für $L_3 = \{(^m)^m \mid m \in \mathbb{N}\} \notin \mathcal{L}_3$ identisch mit dem von L_1
- Beweis für $L_4 = \{w \in \{0, 1\}^* \mid \#_0(w) = \#_1(w)\} \notin \mathcal{L}_3$ ähnlich
($\#_1(w)$ ist die Anzahl der Einsen in w)

- **Verwende Umkehrung der Abschlusseigenschaften**

$$\bar{L} \notin \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

$$L^R \notin \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

$$h(L) \notin \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

$$h^{-1}(L) \notin \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

$$L \cup L' \notin \mathcal{L}_3 \wedge L' \in \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

$$L \cap L' \notin \mathcal{L}_3 \wedge L' \in \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

$$L \circ L' \notin \mathcal{L}_3 \wedge L' \in \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

$$L' \circ L \notin \mathcal{L}_3 \wedge L' \in \mathcal{L}_3 \Rightarrow L \notin \mathcal{L}_3$$

⋮

⋮

- **Anwendungsbeispiele**

$L_3 \notin \mathcal{L}_3$: Wähle Homomorphismus $h: \{(,)\} \rightarrow \{0,1\}$ mit $h(() = 0, h()) = 1$

Dann ist $h(L_3) = \{0^m 1^m \mid m \in \mathbb{N}\} = L_1 \notin \mathcal{L}_3$

$L_4 \notin \mathcal{L}_3$: Es gilt $L_4 \cap L(0^* \circ 1^*) = L_1 \notin \mathcal{L}_3$

DEAs können korrekte Klammersausdrücke nicht erkennen!

- **Abschlusseigenschaften**

- Operationen \cup , \cap , $\bar{}$, $-$, R , \circ , $*$, h , h^{-1} erhalten Regularität von Sprachen
- Verwendbar zum Nachweis von Regularität oder zur Widerlegung

- **Automatische Prüfungen**

- Man kann testen ob eine reguläre Sprache leer ist
- Man kann testen ob ein Wort zu einer regulären Sprache gehört
- Man kann testen ob zwei reguläre Sprachen gleich sind

- **Minimierung von Automaten**

- Ein Automat kann minimiert werden, indem man äquivalente Zustände zusammenlegt und unerreichbare Zustände entfernt

- **Pumping Lemma**

- Wiederholt man einen bestimmten Teil ausreichend großer Wörter einer regulären Sprache beliebig oft, so erhält man immer ein Wort der Sprache
- Verwendbar zur Widerlegung von Regularität

ZUSAMMENFASSUNG: REGULÄRE SPRACHEN

● **Drei Modelle**

- Endliche Automaten (DEA, NEA, ϵ -NEA) **erkennen** Wörter einer Sprache
- Reguläre Ausdrücke **beschreiben Struktur** der Wörter
- (Typ 3) Grammatiken **erzeugen** Wörter einer regulären Sprache

● **Alle drei Modelle sind äquivalent**

- ϵ -NEA \mapsto DEA: Teilmengenkonstruktion
- DEA \mapsto Typ-3 Grammatik: Verwandle Überföhrungsfunktion in Regeln
- Typ-3 Grammatik \mapsto NEA: Verwandle Regeln in Überföhrungsfunktion
- DEA \mapsto Reguläre Ausdrücke: Erzeuge Ausdrücke für Verarbeitungspfade oder eliminiere Zustände in RA Automaten
- Reguläre Ausdrücke \mapsto NEA: Iterative Konstruktion von Automaten

● **Wichtige Eigenschaften von \mathcal{L}_3**

- Abgeschlossen unter $\cup, \cap, \bar{}, -, ^R, \circ, *, h, h^{-1}$
- **Entscheidbarkeit** des Wortproblems und Gleichheit von Sprachen
- Endliche Automaten können automatisch **minimiert** werden
- **Nachweis der Nichtregularität** von Sprachen mit dem Pumping Lemma