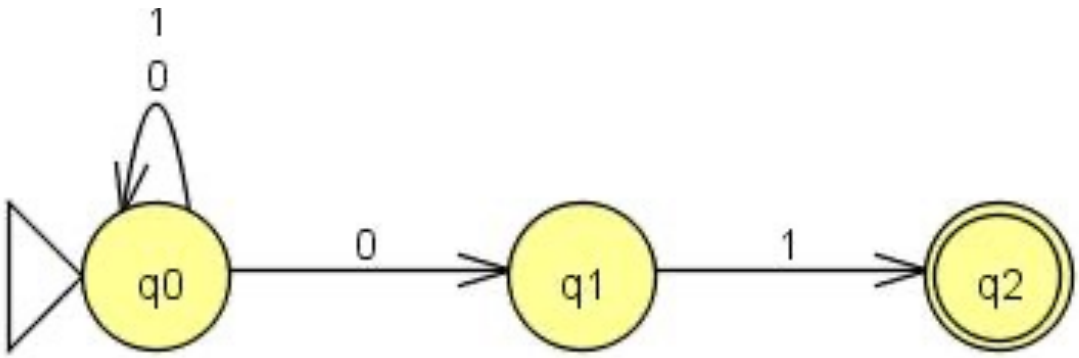
Theoretische Informatik
WS 2024/2025

Kapitel 2
Reguläre Sprachen

2.6
Reguläre Ausdrücke

Prof. Dr. Robert Preis
Fachbereich Informatik
Fachhochschule Dortmund
Robert.Preis@fh-dortmund.de

Wie beschreibe ich die Sprache eines Automaten?



Sprachlich:

Erst 0 oder 1 beliebig oft, dann eine 0 und dann eine 1.

Etwas formaler:

(0 oder 1) beliebig oft, dann 0, dann 1

Regulärer Ausdruck:

(0|1)* ∘ 0 ∘ 1

Wie beschreibe ich die Sprache eines Automaten?

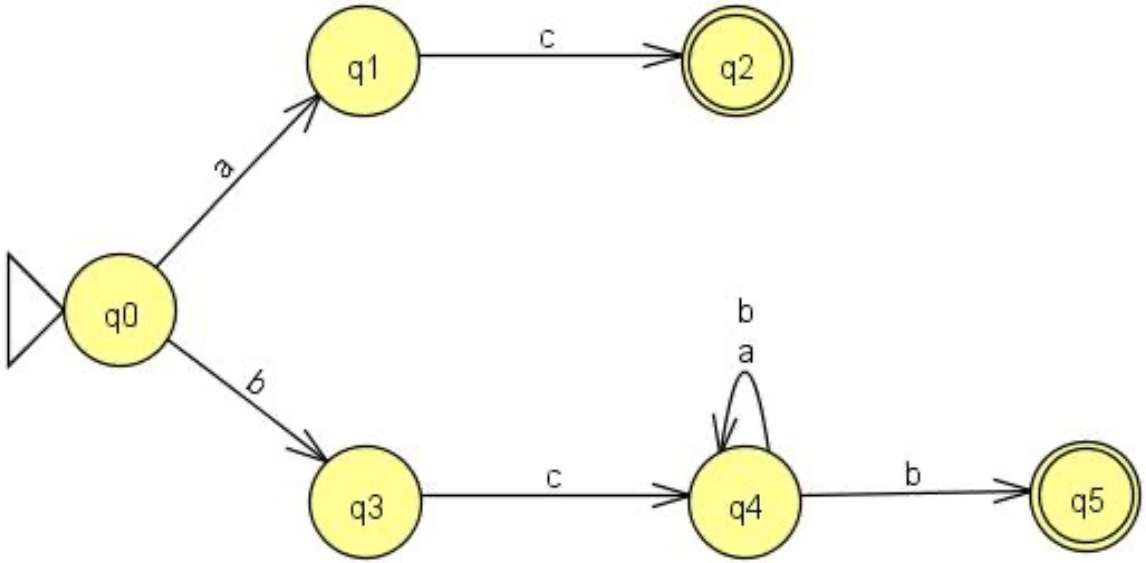
Sprachlich:

Entweder

- *erst ein a und dann ein c oder*
- *erst ein b, dann ein c, dann beliebig viele a oder b und dann ein b.*

Regulärer Ausdruck:

(a ◦ c) | (b ◦ c ◦ (a|b)* ◦ b)



Regeln:

1.	„a oder b“	a | b	(in der Literatur auch manchmal a + b)
2.	„erst a, dann b“	a ◦ b	
3.	„beliebig oft ein a“	a*	
4.	„Klammerung“	(a|b)	

Reguläre Ausdrücke

Syntax von regulären Ausdrücken über einem Alphabet Σ:

Es sind Terme über $\sum \cup \{\varnothing, \varepsilon\} \cup \{|, \circ, (,), *, ^+\}$ Beispiel: $(a \circ c) \mid (b \circ c \circ (a|b)^* \circ b)$

Sie haben einen induktiven Aufbau:

Initiale Sprachen	Initiale reguläre Ausdrücke
• {a}, {b}, {c},...	a, b, c, ...
• {ε}	ε
• ∅	∅

Falls E und F reguläre Ausdrücke, dann auch

• E|F	$L(E\|F) = L(E) \cup L(F)$
• E∘F	$L(E \circ F) = L(E) \circ L(F)$
• E*	$L(E^*) = (L(E))^*$
• E^+	$L(E^+) = (L(E))^+$
• (E)	L((E)) = L(E)

Sprache und Konventionen von Regulären Ausdrücken

Man kann beliebig große reguläre Ausdrücke generieren, z.B.

a◦b◦b◦a◦((a◦b)*|(a◦b|c*)|c◦a◦b*◦a)◦c◦c◦c◦(a|b)◦(b|ε)◦c◦b◦a

D.h. erst ein a, dann ein b, dann ein b, dann ein a, dann…

Konventionen:

- Zeichen für Konkatenation weglassen E◦F = EF
- Definitorische Abkürzungen $E^+ = EE^*$
- Prioritäten: „(a)“ vor „a*“ vor „ab“ vor „a|b“

Beispiel: *Alle Wörter aus a‘s und b‘s, die kein aa und kein bb enthalten.*

- Gerade Länge: (ab)*|(ba)*
- Ungerade Länge: a(ba)*|b(ab)*
- Gesamt: (ab)*|(ba)*|a(ba)*|b(ab)*
- kürzer: (ε|b) (ab)* | (ε|a) (ba)*

Äquivalenz und Algebraische Gesetze

Zwei reguläre Ausdrücke E und F sind äquivalent (abgekürzt: E≡F), wenn deren Sprachen identisch sind, d.h. wenn L(E) = L(F).

Algebraische Gesetze für reguläre Ausdrücke:

- Kommutativgesetz: $E|F \equiv F|E$
- Assoziativgesetz: $(E|F)|G \equiv E|(F|G)$ und $(EF)G \equiv E(FG)$
- Distributivgesetz: $(E|F)G \equiv EG|FG$ und $E(F|G) \equiv EF|EG$
- Idempotenzgesetz: $E|E \equiv E$
- Neutrale Elemente: $\varnothing|E \equiv E$ $\varepsilon E \equiv E$
- Auslöschendes Element: $\varnothing\, E \equiv \varnothing$

- Hüllengesetze: $\varnothing^* \equiv \varepsilon$, $\varepsilon^* \equiv \varepsilon$,
  $(E^*)^* \equiv E^*$,
  $E^+ \equiv EE^*$, $E^* \equiv \varepsilon|E^+$

Wie programmiere ich in reguläre Ausdrücke?

Reguläre Ausdrücke kann man ähnlich wie Grammatiken oder Automaten programmieren:

- {w∈{0,1}* | w enthält mindestens ein Paar aufeinanderfolgender Nullen oder ein Paar aufeinanderfolgender Einsen}

  (0|1)* 00 (0|1)* | (0|1)* 11 (0|1)*

  ≡ (0|1)* (00 | 11) (0|1)*

- {w∈{a,b}* | w ist eine beliebige (auch 0-fache) Wiederholung eines 2-er Strings}

  (aa)* | (ab)* | (ba)* | (bb)*

- {w∈{0,1}* | w enthält die Zeichenketten 00 und 11}

  (0|1)* 00 (0|1)* 11 (0|1)* | (0|1)* 11 (0|1)* 00 (0|1)*

  ≡ (0|1)* (00 (0|1)* 11 | 11 (0|1)* 00) (0|1)*

Umwandlung eines RA in einen Automaten: Induktionsanfang

Man kann jeden regulären Ausdruck E in einen ε-NEA A umwandeln, der die selbe Sprache akzeptiert, d.h. L(E)=L(A), so dass

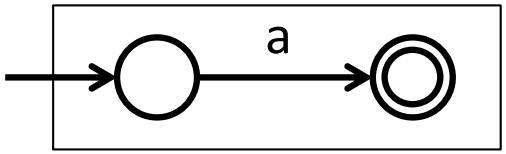
1. Es gibt genau einen Endzustand.
2. Es geht kein Übergang in den Startzustand.
3. Es geht kein Übergang aus dem Endzustand raus.

Diese Restriktionen machen das Zusammenbauen von Teilautomaten besonders einfach !

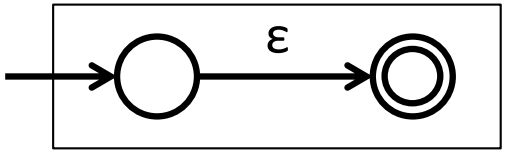
Wir bauen den Automaten mit Induktion auf:

Induktionsanfang der Umwandlung:

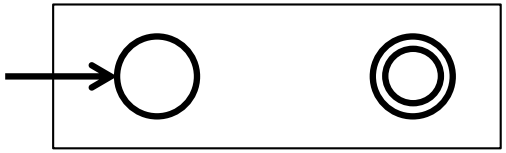
1. Für E = a (a ∈ ∑), d.h. L(E) = {a} wähle A =



2. Für E = ε , d.h. L(E) = {ε} wähle A =



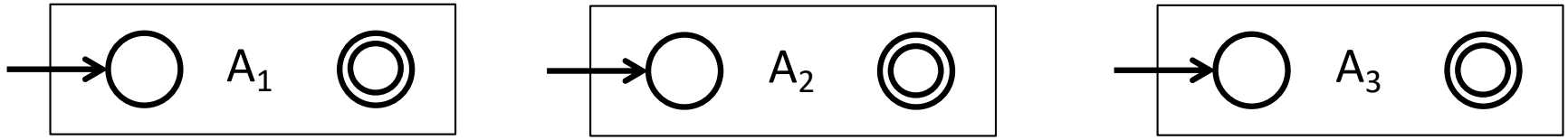
3. Für E = ∅, d.h. L(E) = {} wähle A =



Umwandlung eines RA in einen Automaten: Induktionsschritt

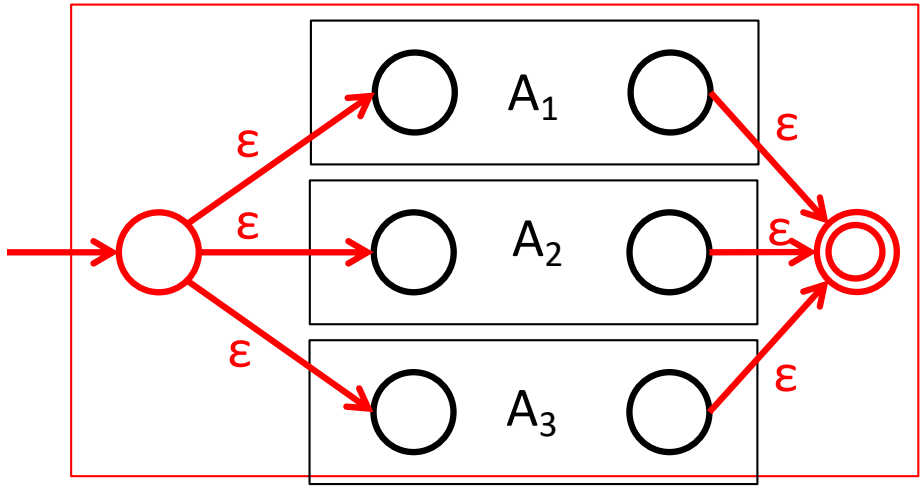
Induktionsannahme:

Seien A_1, A_2 und A_3 ε-NEAs für E_1, E_2 und E_3 mit jeweils einem Start und einem Endzustand:

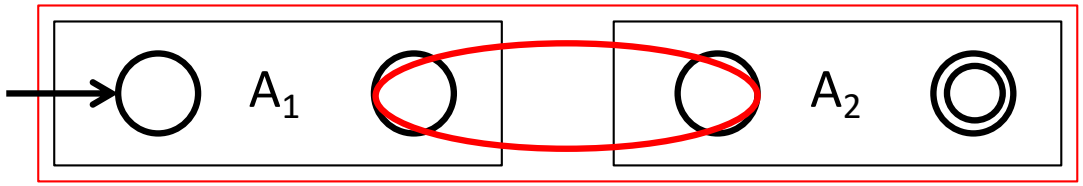


Induktionsschritt:

1. Für $E = E_1|E_2|E_3$, d.h.
   $L(E) = L(E_1) \cup L(E_2) \cup L(E_3)$ wähle A =



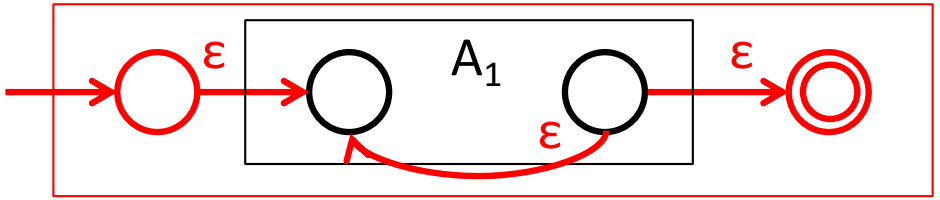
2. Für $E = E_1E_2$, d.h.
   $L(E) = L(E_1) \circ L(E_2)$ wähle A =



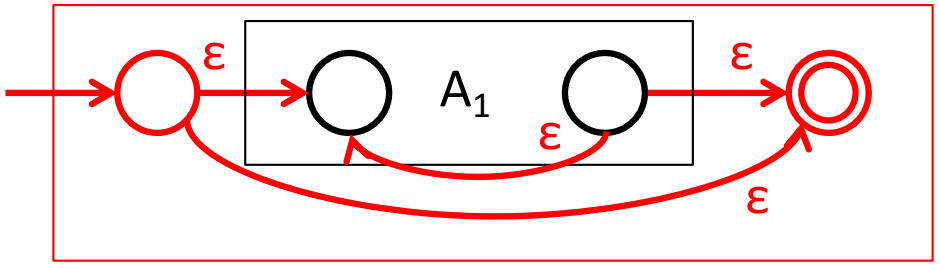
Umwandlung eines RA in einen Automaten: Induktionsschritt

Induktionsschritt:

3. Für $E = E_1^+$, d.h. $L(E) = L(E_1)^+$
   wähle A =



4. Für $E = E_1^*$, d.h. $L(E) = L(E_1)^*$
   wähle A =

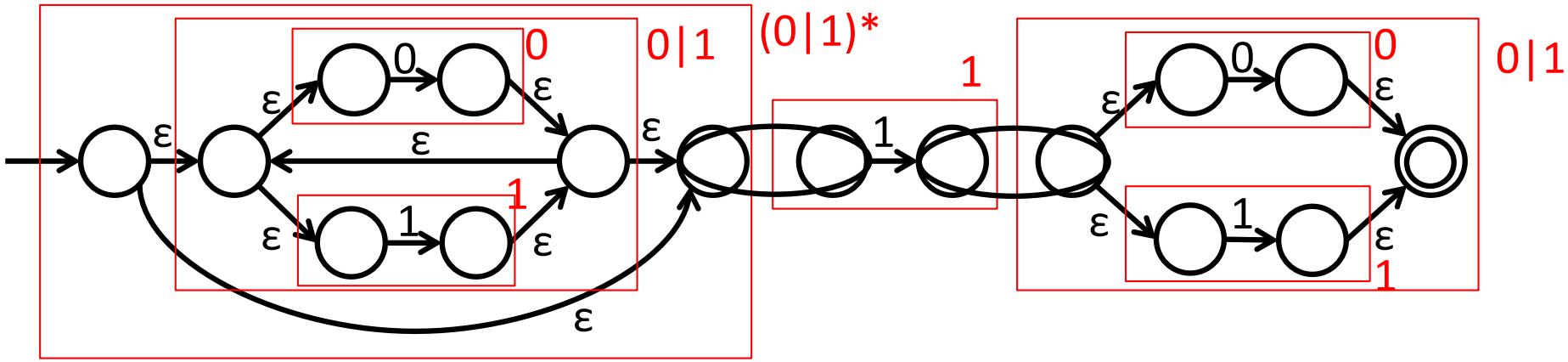


Rechenregel:

$$|\text{Zustände}| = 2\,(|\text{Buchstaben}| + |\text{“ODER“}| + |\text{“*“}| + |\text{“+“}|) - |\text{“}\circ\text{“}|$$

Beispiel einer Umwandlung

Konstruiere endlichen Automaten für (0|1)*1(0|1)



Rechenregel:

2 (|Buchstaben|+|“ODER“|+|“*“|+|“+“|)-|“◦“| = 2(5+2+1+0)-2 = 14

Alle Automaten befolgen die Regeln zum Zeitpunkt ihrer Erzeugung:

1. Es gibt genau einen Endzustand.
2. Es geht kein Übergang in den Startzustand.
3. Es geht kein Übergang aus dem Endzustand raus.

Zusammenfassung

- **Reguläre Ausdrücke** sind eine algebraische Notation für reguläre Sprachen.
- RAs bestehen aus dem **Alphabet, {ε, Ø} und {|, ◦ , *, (,)}.**
- RAs sind **äquivalent zu endlichen Automaten**.
- Es gibt **Anwendungen in Programmiersprachen und Suchmaschinen**.
- Man kann jeden RA in einen Automaten mit der selben Sprache umwandeln.
- Man kann jeden Automaten in einen RA der selben Sprache umwandeln (nicht in dieser Veranstaltung)