

Kapitel 4

Reguläre Sprachen

Abschnitt 4.1

Reguläre Ausdrücke

Reguläre Ausdrücke sind ein Formalismus, um Sprachen zu beschreiben, und zwar nicht als Ergebnisse von Berechnungen, wie es endliche Automaten tun, sondern „direkt“, oder **deklarativ**.

Reguläre Ausdrücke sind ein Formalismus, um Sprachen zu beschreiben, und zwar nicht als Ergebnisse von Berechnungen, wie es endliche Automaten tun, sondern „direkt“, oder **deklarativ**.

Sie verwenden dazu die Operationen auf Sprachen, die wir bereits kennengelernt haben: Mengenoperationen, Verkettung, Iteration.

Beispiel 4.1

Der reguläre Ausdruck

$$a(a + b)^*bb$$

über dem Alphabet $\Sigma = \{a, b\}$ beschreibt die Sprache über Σ , die aus allen Wörtern besteht die mit einem a beginnen und mit zwei bs enden.

Beispiel 4.1

Der reguläre Ausdruck

$$a(a + b)^*bb$$

über dem Alphabet $\Sigma = \{a, b\}$ beschreibt die Sprache über Σ , die aus allen Wörtern besteht die mit einem a beginnen und mit zwei bs enden.

Beispiel 4.2

Der folgende reguläre Ausdruck in UNIX BRE (Basic Regular Expression) Syntax, wie sie zum Beispiel vom Befehl `grep` verwendet wird, beschreibt Emailadressen der RWTH Aachen:

`[a-z0-9\.-]*@[a-z0-9\.-]*\.\rwth-aachen\.`

(nur solche, die aus Kleinbuchstaben, Ziffern, Punkten, und Bindestrichen bestehen).

Definition 4.3

Sei Σ ein Alphabet. Die Menge RA_{Σ} der **regulären Ausdrücke** über Σ ist rekursiv wie folgt definiert.

Basisregeln.

- ▶ $\emptyset \in RA_{\Sigma}$,
- ▶ $\varepsilon \in RA_{\Sigma}$,
- ▶ $a \in RA_{\Sigma}$ für alle $a \in \Sigma$.

Rekursive Regeln.

- ▶ Wenn $r, s \in RA_{\Sigma}$, dann $(r + s) \in RA_{\Sigma}$.
- ▶ Wenn $r, s \in RA_{\Sigma}$, dann $(r \cdot s) \in RA_{\Sigma}$.
- ▶ Wenn $r \in RA_{\Sigma}$, dann $r^* \in RA_{\Sigma}$.

Definition 4.3

Sei Σ ein Alphabet. Die Menge RA_{Σ} der **regulären Ausdrücke** über Σ ist rekursiv wie folgt definiert.

Basisregeln.

- ▶ $\emptyset \in RA_{\Sigma}$,
- ▶ $\varepsilon \in RA_{\Sigma}$,
- ▶ $a \in RA_{\Sigma}$ für alle $a \in \Sigma$.

Rekursive Regeln.

- ▶ Wenn $r, s \in RA_{\Sigma}$, dann $(r + s) \in RA_{\Sigma}$.
- ▶ Wenn $r, s \in RA_{\Sigma}$, dann $(r \cdot s) \in RA_{\Sigma}$.
- ▶ Wenn $r \in RA_{\Sigma}$, dann $r^* \in RA_{\Sigma}$.

Beispiel 4.4

Sei $\Sigma = \{a, b, c\}$.

$$\emptyset, \quad ((\varepsilon + a)b)^+ (b \cdot c)^*, \quad a^{**} \in RA_{\Sigma}.$$

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot
- ▶ Weglassen von Außenklammern

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot
- ▶ Weglassen von Außenklammern
- ▶ Weglassen von Klammern, soweit nach folgenden Präzedenzregeln überflüssig:

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot
- ▶ Weglassen von Außenklammern
- ▶ Weglassen von Klammern, soweit nach folgenden Präzedenzregeln überflüssig:
 - ▶ $*$ bindet stärker als \cdot ,

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot
- ▶ Weglassen von Außenklammern
- ▶ Weglassen von Klammern, soweit nach folgenden Präzedenzregeln überflüssig:
 - ▶ $*$ bindet stärker als \cdot ,
 - ▶ \cdot bindet stärker als $+$.

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot
- ▶ Weglassen von Außenklammern
- ▶ Weglassen von Klammern, soweit nach folgenden Präzedenzregeln überflüssig:
 - ▶ $*$ bindet stärker als \cdot ,
 - ▶ \cdot bindet stärker als $+$.
- ▶ Weglassen von Klammern bei Mehrfach-Summen, Mehrfach-Produkten

$$((a+b)+c)$$

$$\leadsto (a+b)+c$$

$$\leadsto a+b+c$$

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot
- ▶ Weglassen von Außenklammern
- ▶ Weglassen von Klammern, soweit nach folgenden Präzedenzregeln überflüssig:
 - ▶ $*$ bindet stärker als \cdot ,
 - ▶ \cdot bindet stärker als $+$.
- ▶ Weglassen von Klammern bei Mehrfach-Summen, Mehrfach-Produkten
- ▶ Große Summenzeichen: für eine Menge $S = \{a_1, \dots, a_k\} \subseteq \Sigma$ schreiben wir

$$\sum_{a \in S} a \quad \text{oder} \quad \sum_{i=1}^k a_i \quad \text{statt} \quad (a_1 + a_2 + \dots + a_k).$$

Bei Angabe konkreter Ausdrücke erlauben wir:

- ▶ Weglassen des Punktes \cdot
- ▶ Weglassen von Außenklammern
- ▶ Weglassen von Klammern, soweit nach folgenden Präzedenzregeln überflüssig:
 - ▶ $*$ bindet stärker als \cdot ,
 - ▶ \cdot bindet stärker als $+$.
- ▶ Weglassen von Klammern bei Mehrfach-Summen, Mehrfach-Produkten
- ▶ Große Summenzeichen: für eine Menge $S = \{a_1, \dots, a_k\} \subseteq \Sigma$ schreiben wir

$$\sum_{a \in S} a \quad \text{oder} \quad \sum_{i=1}^k a_i \quad \text{statt} \quad (a_1 + a_2 + \dots + a_k).$$

Beispiel 4.5

Statt $((a \cdot b^*) + b) + a$ schreiben wir $ab^* + b + a$

Reguläre Ausdrücke sollen Sprachen beschreiben. Um also die Semantik zu definieren, müssen wir jedem regulären Ausdruck eine Sprache zuordnen.

Reguläre Ausdrücke sollen Sprachen beschreiben. Um also die Semantik zu definieren, müssen wir jedem regulären Ausdruck eine Sprache zuordnen.

Definition 4.6

Rekursiv über den Aufbau von RA_{Σ} definieren wir eine Funktion L , die jedem regulären Ausdruck $r \in RA_{\Sigma}$ eine Sprache $L(r) \subseteq \Sigma^*$ zuordnet:

$$\text{Formel} : L : RA_{\Sigma} \longrightarrow 2^{\Sigma^*}$$

Reguläre Ausdrücke sollen Sprachen beschreiben. Um also die Semantik zu definieren, müssen wir jedem regulären Ausdruck eine Sprache zuordnen.

Definition 4.6

Rekursiv über den Aufbau von RA_{Σ} definieren wir eine Funktion L , die jedem regulären Ausdruck $r \in RA_{\Sigma}$ eine Sprache $L(r) \subseteq \Sigma^*$ zuordnet:

Rekursionsanfang.

- ▶ $L(\emptyset) := \emptyset$,
- ▶ $L(\varepsilon) := \{\varepsilon\}$,
- ▶ $L(a) := \{a\}$ für alle $a \in \Sigma$.

regulären
Ausdruck

Wort

Reguläre Ausdrücke sollen Sprachen beschreiben. Um also die Semantik zu definieren, müssen wir jedem regulären Ausdruck eine Sprache zuordnen.

Definition 4.6

Rekursiv über den Aufbau von RA_{Σ} definieren wir eine Funktion L , die jedem regulären Ausdruck $r \in RA_{\Sigma}$ eine Sprache $L(r) \subseteq \Sigma^*$ zuordnet:

Rekursionsanfang.

- ▶ $L(\emptyset) := \emptyset$,
- ▶ $L(\varepsilon) := \{\varepsilon\}$,
- ▶ $L(a) := \{a\}$ für alle $a \in \Sigma$.

Rekursionsschritt. Für alle $r, s \in RA_{\Sigma}$,

- ▶ $L((r + s)) := L(r) \cup L(s)$,
- ▶ $L((r \cdot s)) := L(r)L(s)$,
- ▶ $L(r^*) := L(r)^*$.

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

$L(r_2)$ ist die Menge aller Wörter in Σ_2^* , in denen höchstens einer der drei Buchstaben vorkommt.

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

$L(r_2)$ ist die Menge aller Wörter in Σ_2^* , in denen höchstens einer der drei Buchstaben vorkommt.

- ▶ $r_3 = (0^*1^*)^*$ über $\Sigma_3 = \{0, 1\}$.

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

$L(r_2)$ ist die Menge aller Wörter in Σ_2^* , in denen höchstens einer der drei Buchstaben vorkommt.

- ▶ $r_3 = (0^*1^*)^*$ über $\Sigma_3 = \{0, 1\}$.

$L(r_3) = \Sigma_3^*$.

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

$L(r_2)$ ist die Menge aller Wörter in Σ_2^* , in denen höchstens einer der drei Buchstaben vorkommt.

- ▶ $r_3 = (0^*1^*)^*$ über $\Sigma_3 = \{0, 1\}$.

$L(r_3) = \Sigma_3^*$.

- ▶ $r_4 = 0 + 1(0 + 1)^*$ über Σ_{ASCII} .

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

$L(r_2)$ ist die Menge aller Wörter in Σ_2^* , in denen höchstens einer der drei Buchstaben vorkommt.

- ▶ $r_3 = (0^*1^*)^*$ über $\Sigma_3 = \{0, 1\}$.

$L(r_3) = \Sigma_3^*$.

- ▶ $r_4 = 0 + 1(0 + 1)^*$ über Σ_{ASCII} .

$L(r_4) = \text{BIN}$ (vgl. Beispiel 3.6)

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

$L(r_2)$ ist die Menge aller Wörter in Σ_2^* , in denen höchstens einer der drei Buchstaben vorkommt.

- ▶ $r_3 = (0^*1^*)^*$ über $\Sigma_3 = \{0, 1\}$.

$L(r_3) = \Sigma_3^*$.

- ▶ $r_4 = 0 + 1(0 + 1)^*$ über Σ_{ASCII} .

$L(r_4) = \text{BIN}$ (vgl. Beispiel 3.6)

- ▶ $r_5 = (a^*b)^*$ über $\Sigma_5 = \{a, b\}$

Von regulären Ausdrücken zu Ihren Sprachen.

- ▶ $r_1 = aaa^*$ über $\Sigma_1 = \{a\}$.

$L(r_1)$ ist die Menge aller Wörter in Σ_1^* der Länge mindestens 2.

- ▶ $r_2 = a^* + b^* + c^*$ über $\Sigma_2 = \{a, b, c\}$

$L(r_2)$ ist die Menge aller Wörter in Σ_2^* , in denen höchstens einer der drei Buchstaben vorkommt.

- ▶ $r_3 = (0^*1^*)^*$ über $\Sigma_3 = \{0, 1\}$.

$L(r_3) = \Sigma_3^*$.

- ▶ $r_4 = 0 + 1(0 + 1)^*$ über Σ_{ASCII} .

$L(r_4) = \text{BIN}$ (vgl. Beispiel 3.6)

- ▶ $r_5 = (a^*b)^*$ über $\Sigma_5 = \{a, b\}$

$L(r_5)$ besteht aus dem leeren Wort und allen Wörtern in Σ_5^* , die mit b enden.

$r_5 = (a^*b)^*$ über $\Sigma_5 = \{a, b\}$.

Behauptung

Für alle $w \in \Sigma_5^*$:

$$w \in L(r_5) \iff w = \varepsilon \text{ oder } w \text{ endet mit } b.$$

Beweis.

Es gilt

$$L(r_5) = (\{a\}^*\{b\})^* = \bigcup_{n \geq 0} (\{a\}^*\{b\})^n. \quad (\star)$$

„ \implies “: Sei $w \in L(r_5)$. Dann gibt es ein $n \in \mathbb{N}$, so dass $w \in (\{a\}^*\{b\})^n$.

Wenn $w \in (\{a\}^*\{b\})^0$, so ist $w = \varepsilon$.

Sonst sei $n \geq 1$ so, dass $w \in (\{a\}^*\{b\})^n$. Nach Beobachtung 1.27 ist

$$w = v_1 \dots v_n$$

für geeignete $v_1, \dots, v_n \in \{a\}^*\{b\}$.

Wiederum nach Beobachtung 1.27 gibt es ein $m \geq 0$, so dass

$$v_n \in \{a\}^m\{b\},$$

das heißt, $v_n = a^m b$. Also ist der letzte Buchstabe von v_n ein b . Der letzte Buchstabe von v_n ist aber auch der letzte Buchstabe von w .

„ \impliedby “: Sei $w \in \Sigma_5^*$, so dass $w = \varepsilon$ oder dass der letzte Buchstabe von w ein b ist.

Wenn $w = \varepsilon$, so ist $w \in (\{a\}^*\{b\})^0 \subseteq (\{a\}^*\{b\})^* = L(r_5)$.

Sonst ist $w = a_1 \dots a_n$ für ein $n \geq 1$ und $a_1, \dots, a_n \in \Sigma_5$.

Sei $i_0 := 0$, und seien $1 \leq i_1 < i_2 < \dots < i_k$ die Indizes der b s in w , d.h., $a_{i_j} = b$ für $1 \leq j \leq k$ und $a_i = a$ für $i \in \{1, \dots, n\} \setminus \{i_1, \dots, i_k\}$.

Für $1 \leq j \leq k$ sei

$$v_j := a_{i_{j-1}+1} \dots a_{i_j}.$$

Weil der letzte Buchstabe von w ein b ist gilt $i_k = n$, also ist

$$w = v_1 \dots v_k.$$

Außerdem gilt

$$v_j = a_{i_{j-1}+1} \dots a_{i_j} = \underbrace{a \dots a}_{i_j - i_{j-1} - 1 \text{ mal}} b \in \{a\}^*\{b\}.$$

Damit

$$w \in (\{a\}^*\{b\})^k \subseteq (\{a\}^*\{b\})^* = L(r_5).$$

□

Von Sprachen zu regulären Ausdrücken.

- L_1 sei die Menge aller Wörter über $\Sigma_1 := \{0, 1\}$, die 1101 als Infix enthalten.

$$(0+1)^* 1101 (0+1)^*$$

$$(0^*1^*)^* 1101 (0+1)^*$$

Von Sprachen zu regulären Ausdrücken.

- ▶ L_1 sei die Menge aller Wörter über $\Sigma_1 := \{0, 1\}$, die 1101 als Infix enthalten.

$$L_1 = L(r_1) \text{ für } r_1 = (0 + 1)^* 1101 (0 + 1)^*.$$

Von Sprachen zu regulären Ausdrücken.

- ▶ L_1 sei die Menge aller Wörter über $\Sigma_1 := \{0, 1\}$, die 1101 als Infix enthalten.

$$L_1 = L(r_1) \text{ für } r_1 = (0 + 1)^* 1101 (0 + 1)^*.$$

- ▶ INT sei die Menge der Dezimaldarstellungen ganzer Zahlen über dem Alphabet Σ_{ASCII} .

Von Sprachen zu regulären Ausdrücken.

- ▶ L_1 sei die Menge aller Wörter über $\Sigma_1 := \{0, 1\}$, die 1101 als Infix enthalten.

$$L_1 = L(r_1) \text{ für } r_1 = (0 + 1)^* 1101 (0 + 1)^*.$$

- ▶ INT sei die Menge der Dezimaldarstellungen ganzer Zahlen über dem Alphabet Σ_{ASCII} .

Wir definieren zunächst den Hilfsausdruck

$$z := (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9).$$

Dann ist $\text{INT} = L(r)$ für

$$r = (\varepsilon + -)z(0 + z)^*.$$

Von Sprachen zu regulären Ausdrücken.

- ▶ L_1 sei die Menge aller Wörter über $\Sigma_1 := \{0, 1\}$, die 1101 als Infix enthalten.

$$L_1 = L(r_1) \text{ für } r_1 = (0 + 1)^* 1101 (0 + 1)^*.$$

- ▶ INT sei die Menge der Dezimaldarstellungen ganzer Zahlen über dem Alphabet Σ_{ASCII} .

Wir definieren zunächst den Hilfsausdruck

$$z := (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9).$$

Dann ist $\text{INT} = L(r)$ für

$$r = (\varepsilon + -)z(0 + z)^*.$$

- ▶ Sei Σ ein beliebiges Alphabet, etwa $\Sigma = \{a_1, \dots, a_k\}$.

Von Sprachen zu regulären Ausdrücken.

- ▶ L_1 sei die Menge aller Wörter über $\Sigma_1 := \{0, 1\}$, die 1101 als Infix enthalten.

$$L_1 = L(r_1) \text{ für } r_1 = (0 + 1)^* 1101 (0 + 1)^*.$$

- ▶ INT sei die Menge der Dezimaldarstellungen ganzer Zahlen über dem Alphabet Σ_{ASCII} .

Wir definieren zunächst den Hilfsausdruck

$$z := (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9).$$

Dann ist $\text{INT} = L(r)$ für

$$r = (\varepsilon + -)z(0 + z)^*.$$

- ▶ Sei Σ ein beliebiges Alphabet, etwa $\Sigma = \{a_1, \dots, a_k\}$.

Dann ist

$$\Sigma^* = L((a_1 + a_2 + \dots + a_k)^*).$$

$$\left(\sum_{a \in \Sigma} a \right)^*$$

Definition 4.9

1. Zwei reguläre Ausdrücke r, s sind **äquivalent** (wir schreiben $r \equiv s$), wenn $L(r) = L(s)$.

Definition 4.9

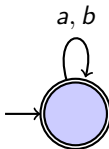
1. Zwei reguläre Ausdrücke r, s sind **äquivalent** (wir schreiben $r \equiv s$), wenn $L(r) = L(s)$.
2. Ein regulärer Ausdruck r und ein endlicher Automat (DFA, NFA oder ε -NFA) sind **äquivalent**, wenn $L(r) = L(\mathcal{A})$.

Definition 4.9

1. Zwei reguläre Ausdrücke r, s sind **äquivalent** (wir schreiben $r \equiv s$), wenn $L(r) = L(s)$.
2. Ein regulärer Ausdruck r und ein endlicher Automat (DFA, NFA oder ε -NFA) sind **äquivalent**, wenn $L(r) = L(\mathcal{A})$.

Beispiel 4.10

Die regulären Ausdrücke $(a + b)^*$ und $(a^*b^*)^*$ sind äquivalent, und sie sind äquivalent zum DFA



Definition 4.11

Eine Sprache $L \subseteq \Sigma^*$ ist **regulär**, wenn es einen regulären Ausdruck $r \in \text{RA}_\Sigma$ gibt, so dass $L = L(r)$.

Definition 4.11

Eine Sprache $L \subseteq \Sigma^*$ ist **regulär**, wenn es einen regulären Ausdruck $r \in \text{RA}_\Sigma$ gibt, so dass $L = L(r)$.

Satz 4.12

Jede reguläre Sprache ist FA-erkennbar.

Wir zeigen induktiv über den Aufbau eines regulären Ausdrucks $r \in \text{RA}_\Sigma$, dass $L(r)$ FA-erkennbar ist.

Induktionsanfang.

Behauptung: $L(\emptyset)$, $L(\varepsilon)$ und $L(a)$ für $a \in \Sigma$ sind FA-erkennbar.

Beweis: Alle endlichen Sprachen sind FA-erkennbar.

Im Einzelnen:

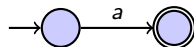
▶ $L(\emptyset) = \emptyset$ wird erkannt durch den NFA



▶ $L(\varepsilon) = \{\varepsilon\}$ wird erkannt durch den NFA



▶ $L(a) = \{a\}$ wird erkannt durch den NFA



Induktionsschritt. Seien $r, s \in \text{RA}_\Sigma$.

Induktionsannahme: $L(r)$ und $L(s)$ sind FA-erkennbar.

Behauptung: $L(r + s)$, $L(rs)$ und $L(r^*)$ sind FA-erkennbar.

Beweis:

- ▶ $L(r + s) = L(r) \cup L(s)$ ist FA-erkennbar nach Satz 2.40(2).
- ▶ $L(r \cdot s) = L(r)L(s)$ ist FA-erkennbar nach Satz 2.40(4).
- ▶ $L(r^*) = L(r)^*$ ist FA-erkennbar nach Satz 2.40(5).

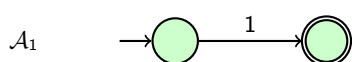
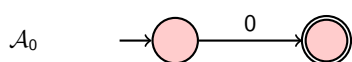
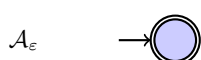
□

Wir konstruieren zum regulären Ausdruck

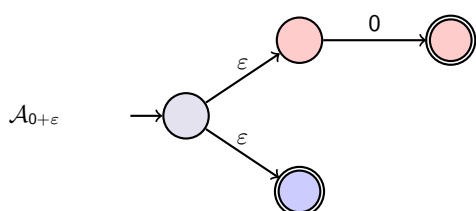
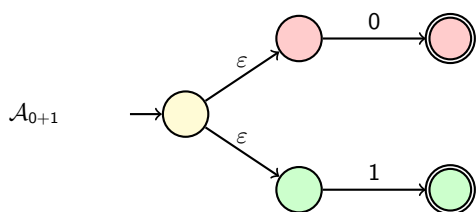
$$(0 + 1)^* 1(0 + \varepsilon)$$

einen äquivalenten ε -NFA.

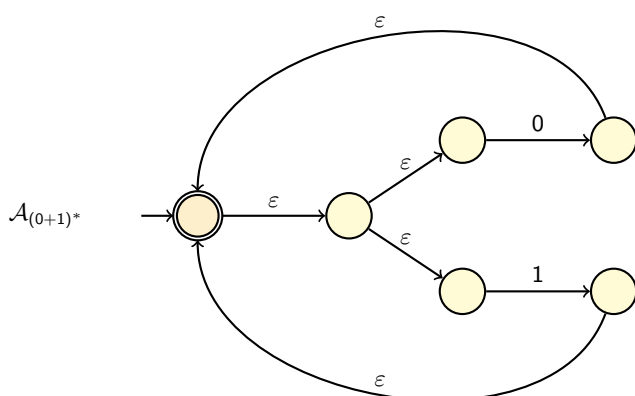
Zunächst erhalten wir für die Teilausdrücke $\varepsilon, 0, 1$:



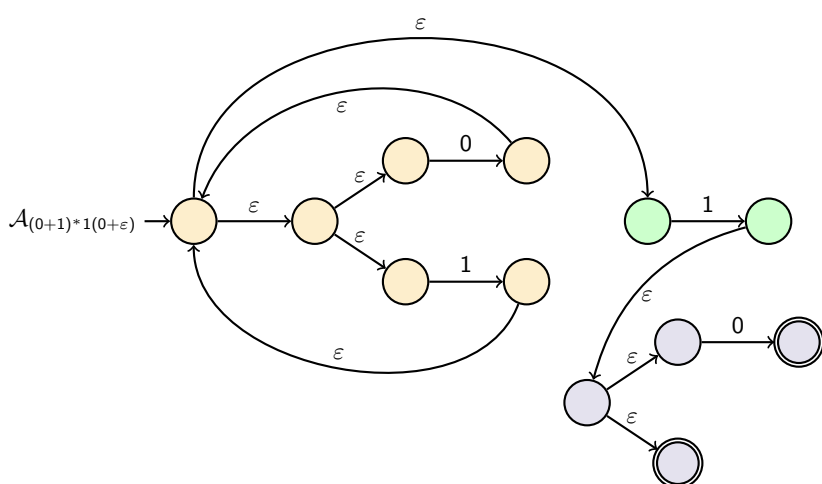
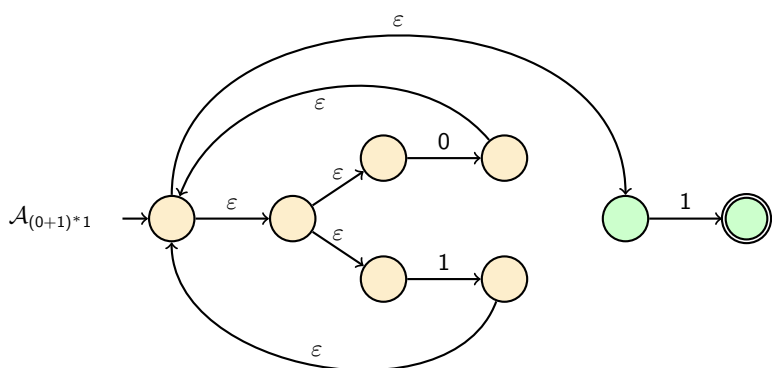
Daraus konstruieren wir für $(0 + 1)$ und $(0 + \varepsilon)$:



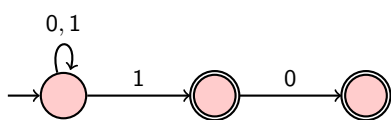
Aus \mathcal{A}_{0+1} erhalten wir für $(0 + 1)^*$:



Jetzt ergibt sich durch Verketten für $(0 + 1)^* 1$ und $(0 + 1)^* 1(0 + \varepsilon)$:



Ein einfacherer äquivalenter NFA ist



Diskussion

Die allgemeine Konstruktion liefert also in der Regel keinen „optimalen“ ε -NFA für einen regulären Ausdruck.

Dafür funktioniert diese Konstruktion immer und lässt sich leicht automatisieren.

Man kann dann immer noch versuchen, den resultierenden ε -NFA zu „optimieren“, etwa durch Elimination der ε -Transitionen.

FA-erkennbare Sprachen sind regulär

Satz 4.14

Jede FA-erkennbare Sprache ist regulär.

Sei $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ ein NFA.

Wir halten \mathcal{A} bis zum Ende des Beweises fest.

Sei $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ ein NFA.

Wir halten \mathcal{A} bis zum Ende des Beweises fest.

Ziel

Berechne für alle $q \in F$ einen regulären Ausdruck $r_{q_0 q}$, so dass

$$L(r_{q_0 q}) = \{w \mid q_0 \xrightarrow{w} q\}.$$

Sei $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ ein NFA.

Wir halten \mathcal{A} bis zum Ende des Beweises fest.

Ziel

Berechne für alle $q \in F$ einen regulären Ausdruck $r_{q_0 q}$, so dass

$$L(r_{q_0 q}) = \{w \mid q_0 \xrightarrow{w} q\}.$$

Dann ist

$$L(\mathcal{A}) = \bigcup_{q \in F} L(r_{q_0 q}) = L\left(\sum_{q \in F} r_{q_0 q}\right).$$

Sei $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ ein NFA.

Wir halten \mathcal{A} bis zum Ende des Beweises fest.

Ziel

Berechne für alle $q \in F$ einen regulären Ausdruck $r_{q_0 q}$, so dass

$$L(r_{q_0 q}) = \{w \mid q_0 \xrightarrow{w} q\}.$$

Dann ist

$$L(\mathcal{A}) = \bigcup_{q \in F} L(r_{q_0 q}) = L\left(\sum_{q \in F} r_{q_0 q}\right).$$

Um $r_{q_0 q}$ induktiv berechnen zu können, berechnen wir für alle $p, q \in Q$ einen regulären Ausdruck r_{pq} , so dass

$$L(r_{pq}) = \{w \mid p \xrightarrow{w} q\}.$$

Frage

Wie sollen wir r_{q_0q} induktiv berechnen?

Frage

Wie sollen wir $r_{q_0 q}$ induktiv berechnen?

Idee

Wir führen Induktion über die Anzahl der Zustände, die in einem Lauf von p nach q verwendet werden.

Technische Umsetzung

Für $p, q \in Q$ und $X \subseteq Q$ schreiben wir $p \xrightarrow[X]{w} q$ wenn es einen Lauf von p nach q über w gibt, der neben p, q nur Zustände aus X verwendet.

Technische Umsetzung

Für $p, q \in Q$ und $X \subseteq Q$ schreiben wir $p \xrightarrow[X]{w} q$ wenn es einen Lauf von p nach q über w gibt, der neben p, q nur Zustände aus X verwendet.

Genauer: Für $w = a_1 \dots a_n$,

$$p \xrightarrow[X]{w} q \iff \exists x_0 = p, x_1, \dots, x_n = q, \text{ so dass } x_1, \dots, x_{n-1} \in X$$
$$\text{und } (x_{i-1}, a_i, x_i) \in \Delta \text{ für } 1 \leq i \leq n.$$

Technische Umsetzung

Für $p, q \in Q$ und $X \subseteq Q$ schreiben wir $p \xrightarrow[X]{w} q$ wenn es einen Lauf von p nach q über w gibt, der neben p, q nur Zustände aus X verwendet.

Genauer: Für $w = a_1 \dots a_n$,

$$p \xrightarrow[X]{w} q \iff \exists x_0 = p, x_1, \dots, x_n = q, \text{ so dass } x_1, \dots, x_{n-1} \in X$$
$$\text{und } (x_{i-1}, a_i, x_i) \in \Delta \text{ für } 1 \leq i \leq n.$$

Lemma 4.15

Seien $p, q \in Q$ und $w \in \Sigma^*$

1. $p \xrightarrow[\emptyset]{w} q \iff w = \sigma \text{ für ein } \sigma \in \Sigma \cup \{\varepsilon\}, \text{ so dass } p \xrightarrow{\sigma} q.$

Technische Umsetzung

Für $p, q \in Q$ und $X \subseteq Q$ schreiben wir $p \xrightarrow[X]{w} q$ wenn es einen Lauf von p nach q über w gibt, der neben p, q nur Zustände aus X verwendet.

Genauer: Für $w = a_1 \dots a_n$,

$$p \xrightarrow[X]{w} q \iff \exists x_0 = p, x_1, \dots, x_n = q, \text{ so dass } x_1, \dots, x_{n-1} \in X \\ \text{und } (x_{i-1}, a_i, x_i) \in \Delta \text{ für } 1 \leq i \leq n.$$

Lemma 4.15

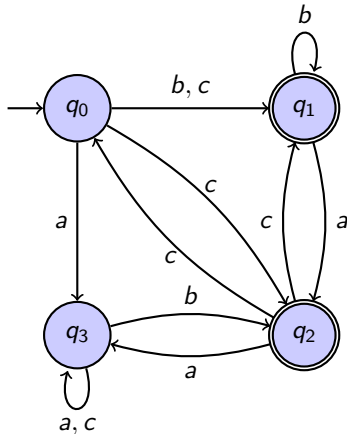
Seien $p, q \in Q$ und $w \in \Sigma^*$

1. $p \xrightarrow[\emptyset]{w} q \iff w = \sigma$ für ein $\sigma \in \Sigma \cup \{\varepsilon\}$, so dass $p \xrightarrow{\sigma} q$.
2. Für alle $X \neq \emptyset$ und $x \in X$ gilt

$$p \xrightarrow[X]{w} q \iff p \xrightarrow[X \setminus \{x\}]{w} q \text{ oder es gibt ein } k \geq 0 \text{ und eine Zerlegung} \\ w = tu_1 \dots u_k v \text{ von } w, \text{ so dass} \\ p \xrightarrow[X \setminus \{x\}]{t} x \xrightarrow[X \setminus \{x\}]{u_1} x \xrightarrow[X \setminus \{x\}]{u_2} x \dots x \xrightarrow[X \setminus \{x\}]{u_k} x \xrightarrow[X \setminus \{x\}]{v} q.$$

Beispiel 4.16

Sei \mathcal{A} der folgende NFA:



Es gilt:

$$q_0 \xrightarrow[\emptyset]{b} q_1, \quad q_0 \xrightarrow[\emptyset]{c} q_1$$

$$q_0 \xrightarrow[\{q_1\}]{bb} q_1$$

$$q_0 \xrightarrow[\{q_1\}]{cbb} q_1$$

$$q_0 \xrightarrow[\{q_2\}]{cc} q_1$$

$$q_0 \xrightarrow[\{q_1, q_2\}]{ccbbbb} q_1$$

$$q_0 \xrightarrow[\{q_2, q_3\}]{aacaccbc} q_1$$

$$q_0 \xrightarrow[\{q_1, q_2, q_3\}]{aacaccbcbbb} q_1$$

$$q_0 \xrightarrow[\{q_0, q_1, q_2, q_3\}]{abcbbbbb} q_1.$$

Lemma 4.15

Seien $p, q \in Q$ und $w \in \Sigma^*$

1. $p \xrightarrow[\emptyset]{w} q \iff w = \sigma$ für ein $\sigma \in \Sigma \cup \{\varepsilon\}$, so dass $p \xrightarrow{\sigma} q$.
2. Für alle $X \neq \emptyset$ und $x \in X$ gilt

$$p \xrightarrow[X]{w} q \iff p \xrightarrow[X \setminus \{x\}]{w} q \text{ oder es gibt ein } k \geq 0 \text{ und eine Zerlegung } w = tu_1 \dots u_k v \text{ von } w, \text{ so dass}$$

$$p \xrightarrow[X \setminus \{x\}]{t} x \xrightarrow[X \setminus \{x\}]{u_1} x \xrightarrow[X \setminus \{x\}]{u_2} x \dots x \xrightarrow[X \setminus \{x\}]{u_k} x \xrightarrow[X \setminus \{x\}]{v} q.$$

Beweis.

1. ist offensichtlich.

2. „ \Leftarrow “: Klar.

„ \Rightarrow “: Es gelte $p \xrightarrow[X]{w} q$.

Sei $w = a_1 \dots a_n$, und sei $(x_0 = p, x_1, \dots, x_n = q)$ ein Lauf von p nach q über w mit $x_1, \dots, x_{n-1} \in X$.

Falls $x \notin \{x_1, \dots, x_{n-1}\}$, so gilt $p \xrightarrow[X \setminus \{x\}]{w} q$.

Sonst seien $k \geq 0$ und $1 \leq i_0 < i_1 < \dots < i_k \leq n-1$ die Indizes i mit $x_i = x$.

Seien

$$t := a_1 \dots a_{i_0},$$

$$u_j := a_{i_{j-1}+1} \dots a_{i_j} \quad \text{für } 1 \leq j \leq k,$$

$$v := a_{i_k+1} \dots a_n$$

Dann gilt $w = tu_1 \dots u_k v$ und

$$p \xrightarrow[X \setminus \{x\}]{t} x \xrightarrow[X \setminus \{x\}]{u_1} x \xrightarrow[X \setminus \{x\}]{u_2} x \dots x \xrightarrow[X \setminus \{x\}]{u_k} x \xrightarrow[X \setminus \{x\}]{v} q.$$

□

Lemma 4.17

Für alle $X \subseteq Q$ und $p, q \in Q$ gibt es einen regulären Ausdruck $r_X(p, q)$, so dass

$$L(r_X(p, q)) = \{w \in \Sigma^* \mid p \xrightarrow[X]{w} q\}.$$

Beweis von Lemma 4.17

Wir beweisen das Lemma per Induktion über $m := |X|$.

Induktionsanfang: $m = 0$.

Dann ist $X = \emptyset$, und es gilt

$$p \xrightarrow{w}_X q \iff w = \sigma \text{ für ein } \sigma \in \Sigma \cup \{\varepsilon\} \text{ mit } p \xrightarrow{\sigma} q.$$

Seien $\sigma_1, \dots, \sigma_k$ alle $\sigma \in \Sigma \cup \{\varepsilon\}$ mit $p \xrightarrow{\sigma} q$. Wir setzen

$$r_{\emptyset}(p, q) := (\sigma_1 + \dots + \sigma_k).$$

Induktionsschritt: $m \rightarrow m + 1$. Sei $x \in X$ und $X' := X \setminus \{x\}$.

Nach Induktionsannahme finden wir Ausdrücke $r_{X'}(p, q)$, $r_{X'}(p, x)$, $r_{X'}(x, x)$, $r_{X'}(x, q)$ mit den gewünschten Eigenschaften.

Wir setzen

$$r_X(p, q) := r_{X'}(p, q) + r_{X'}(p, x)r_{X'}(x, x)^*r_{X'}(x, q).$$

Behauptung

Für alle $w \in \Sigma^*$ gilt

$$p \xrightarrow{w}_X q \iff w \in L(r_X(p, q))$$

Beweis.

Sei $w \in \Sigma^*$. Nach Lemma 4.15 gilt

$$p \xrightarrow{w}_X q \iff p \xrightarrow{w}_{X'} q \text{ oder es gibt ein } k \geq 0 \text{ und eine Zerlegung } w = tu_1 \dots u_k v \text{ von } w, \text{ so dass}$$

$$p \xrightarrow{t}_{X'} x \xrightarrow{u_1}_{X'} x \xrightarrow{u_2}_{X'} x \dots x \xrightarrow{u_k}_{X'} x \xrightarrow{v}_{X'} q.$$

Nach Induktionsannahme gilt

$$p \xrightarrow{w}_{X'} q \iff w \in L(r_{X'}(p, q))$$

und

$$p \xrightarrow{t}_{X'} x \iff t \in L(r_{X'}(p, x)),$$

$$x \xrightarrow{u_i}_{X'} x \iff u_i \in L(r_{X'}(x, x)),$$

$$x \xrightarrow{v}_{X'} q \iff v \in L(r_{X'}(x, q)).$$

Daraus folgt, dass $w = tu_1 \dots u_k v$ für ein $k \geq 0$ und $t, u_1, \dots, u_k, v \in \Sigma^*$ mit

$$p \xrightarrow{t}_{X'} x \xrightarrow{u_1}_{X'} x \xrightarrow{u_2}_{X'} x \dots x \xrightarrow{u_k}_{X'} x \xrightarrow{v}_{X'} q$$

genau dann wenn

$$w \in L(r_{X'}(p, x)r_{X'}(x, x)^*r_{X'}(x, q)).$$

Daraus ergibt sich die Behauptung und damit das Lemma. □

Satz 4.14

Jede FA-erkennbare Sprache ist regulär.

Beweis.

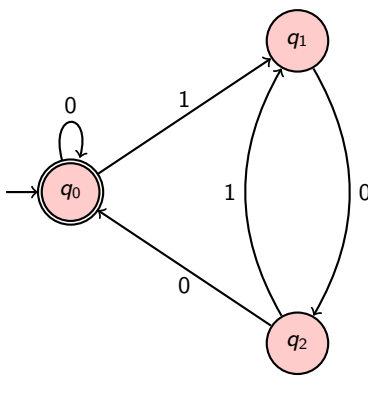
Wir betrachten wieder unseren NFA $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$.

Für alle $w \in \Sigma^*$ gilt

$$\begin{aligned} w \in L(\mathcal{A}) &\iff \exists q \in F : q_0 \xrightarrow[Q]{w} q \\ &\iff \exists q \in F : w \in L(r_Q(q_0, q)) \quad (\text{Lemma 4.17}) \\ &\iff w \in L\left(\sum_{q \in F} r_Q(q_0, q)\right). \end{aligned}$$

Also ist $r := \sum_{q \in F} r_Q(q_0, q)$ ein regulärer Ausdruck mit $L(r) = L(\mathcal{A})$. □

Sei \mathcal{A} folgender NFA:



Wir konstruieren einen regulären Ausdruck r mit $L(r) = L(\mathcal{A})$.

Zu berechnen: $r_Q(q_0, q_0)$

Wir berechnen: $r_Q(q_0, q_0)$.

$$X = Q, x = q_1$$

Mit $x = q_1$ erhalten wir

$$r_Q(q_0, q_0) = r_{\{q_0, q_2\}}(q_0, q_0) + r_{\{q_0, q_2\}}(q_0, q_1) r_{\{q_0, q_2\}}(q_1, q_1)^* r_{\{q_0, q_2\}}(q_1, q_0).$$

Noch zu berechnen: $r_{\{q_0, q_2\}}(q_0, q_0)$, $r_{\{q_0, q_2\}}(q_0, q_1)$, $r_{\{q_0, q_2\}}(q_1, q_1)$, $r_{\{q_0, q_2\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0, q_2\}}(q_0, q_0)$.

$$X = \{q_0, q_2\}, x = q_2$$

Mit $x = q_2$ erhalten wir

$$r_{\{q_0, q_2\}}(q_0, q_0) = r_{\{q_0\}}(q_0, q_0) + \underline{r_{\{q_0\}}(q_0, q_2)} r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_0).$$

Noch zu berechnen: $r_{\{q_0, q_2\}}(q_0, q_1)$, $r_{\{q_0, q_2\}}(q_1, q_1)$, $r_{\{q_0, q_2\}}(q_1, q_0)$, $r_{\{q_0\}}(q_0, q_0)$, $\underline{r_{\{q_0\}}(q_0, q_2)}$, $\underline{r_{\{q_0\}}(q_2, q_2)}$, $\underline{r_{\{q_0\}}(q_2, q_0)}$.

Wir berechnen: $r_{\{q_0, q_2\}}(q_0, q_1)$.

Mit $x = q_2$ erhalten wir

$$r_{\{q_0, q_2\}}(q_0, q_1) = r_{\{q_0\}}(q_0, q_1) + r_{\{q_0\}}(q_0, q_2) r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_1).$$

Noch zu berechnen: $r_{\{q_0, q_2\}}(q_1, q_1)$, $r_{\{q_0, q_2\}}(q_1, q_0)$, $r_{\{q_0\}}(q_0, q_0)$, $r_{\{q_0\}}(q_0, q_2)$, $r_{\{q_0\}}(q_2, q_2)$, $r_{\{q_0\}}(q_2, q_0)$, $r_{\{q_0\}}(q_0, q_1)$, $r_{\{q_0\}}(q_2, q_1)$

Wir berechnen: $r_{\{q_0, q_2\}}(q_1, q_1)$.

Mit $x = q_2$ erhalten wir

$$r_{\{q_0, q_2\}}(q_1, q_1) = r_{\{q_0\}}(q_1, q_1) + r_{\{q_0\}}(q_1, q_2) r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_1).$$

Noch zu berechnen: $r_{\{q_0, q_2\}}(q_1, q_0)$, $r_{\{q_0\}}(q_0, q_0)$, $r_{\{q_0\}}(q_0, q_2)$, $r_{\{q_0\}}(q_2, q_2)$, $r_{\{q_0\}}(q_2, q_0)$, $r_{\{q_0\}}(q_0, q_1)$, $r_{\{q_0\}}(q_2, q_1)$, $r_{\{q_0\}}(q_0, q_0)$

Wir berechnen: $r_{\{q_0, q_2\}}(q_1, q_0)$.

Mit $x = q_2$ erhalten wir

$$r_{\{q_0, q_2\}}(q_1, q_0) = r_{\{q_0\}}(q_1, q_0) + r_{\{q_0\}}(q_1, q_2) r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_0).$$

Noch zu berechnen: $r_{\{q_0\}}(q_0, q_0)$, $r_{\{q_0\}}(q_0, q_2)$, $r_{\{q_0\}}(q_2, q_2)$, $r_{\{q_0\}}(q_2, q_0)$, $r_{\{q_0\}}(q_0, q_1)$, $r_{\{q_0\}}(q_2, q_1)$, $r_{\{q_0\}}(q_0, q_0)$, $r_{\{q_0\}}(q_2, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_0, q_0)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_0, q_0) &= r_{\emptyset}(q_0, q_0) + r_{\emptyset}(q_0, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_0) \\ &= (0 + \varepsilon) + (0 + \varepsilon)(0 + \varepsilon)^*(0 + \varepsilon) \\ &\equiv 0^* \end{aligned}$$

Hier und im folgenden bedeutet $r \equiv r'$, dass die beiden Ausdrücke r und r' äquivalent sind, das heißt, $L(r) = L(r')$.

Noch zu berechnen: $r_{\{q_0\}}(q_0, q_2)$, $r_{\{q_0\}}(q_2, q_2)$, $r_{\{q_0\}}(q_2, q_0)$, $r_{\{q_0\}}(q_0, q_1)$, $r_{\{q_0\}}(q_2, q_1)$, $r_{\{q_0\}}(q_1, q_1)$, $r_{\{q_0\}}(q_1, q_2)$, $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_0, q_2)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_0, q_2) &= r_{\emptyset}(q_0, q_2) + r_{\emptyset}(q_0, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_2) \\ &\equiv \emptyset \end{aligned}$$

Noch zu berechnen: $r_{\{q_0\}}(q_2, q_2)$, $r_{\{q_0\}}(q_2, q_0)$, $r_{\{q_0\}}(q_0, q_1)$, $r_{\{q_0\}}(q_2, q_1)$, $r_{\{q_0\}}(q_1, q_1)$, $r_{\{q_0\}}(q_1, q_2)$, $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_2, q_2)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_2, q_2) &= r_{\emptyset}(q_2, q_2) + r_{\emptyset}(q_2, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_2) \\ &\equiv \varepsilon \end{aligned}$$

Noch zu berechnen: $r_{\{q_0\}}(q_2, q_0)$, $r_{\{q_0\}}(q_0, q_1)$, $r_{\{q_0\}}(q_2, q_1)$, $r_{\{q_0\}}(q_1, q_1)$, $r_{\{q_0\}}(q_1, q_2)$, $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_2, q_0)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_2, q_0) &= r_{\emptyset}(q_2, q_0) + r_{\emptyset}(q_2, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_0) \\ &= 0 + 0(0 + \varepsilon)^*(0 + \varepsilon) \\ &\equiv 00^* \end{aligned}$$

Noch zu berechnen: $r_{\{q_0\}}(q_0, q_1)$, $r_{\{q_0\}}(q_2, q_1)$, $r_{\{q_0\}}(q_1, q_1)$, $r_{\{q_0\}}(q_1, q_2)$, $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_0, q_1)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_0, q_1) &= r_{\emptyset}(q_0, q_1) + r_{\emptyset}(q_0, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_1) \\ &= 1 + (0 + \varepsilon)(0 + \varepsilon)^* 1 \\ &\equiv 0^* 1 \end{aligned}$$

Noch zu berechnen: $r_{\{q_0\}}(q_2, q_1)$, $r_{\{q_0\}}(q_1, q_1)$, $r_{\{q_0\}}(q_1, q_2)$, $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_2, q_1)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_2, q_1) &= r_{\emptyset}(q_2, q_1) + r_{\emptyset}(q_2, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_1) \\ &= 1 + 0(0 + \varepsilon)^* 1 \\ &\equiv 0^* 1 \end{aligned}$$

Noch zu berechnen: $r_{\{q_0\}}(q_1, q_1)$, $r_{\{q_0\}}(q_1, q_2)$, $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_1, q_1)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_1, q_1) &= r_{\emptyset}(q_1, q_1) + r_{\emptyset}(q_1, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_1) \\ &\equiv \varepsilon \end{aligned}$$

Noch zu berechnen: $r_{\{q_0\}}(q_1, q_2)$, $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_1, q_2)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_1, q_2) &= r_{\emptyset}(q_1, q_2) + r_{\emptyset}(q_1, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_2) \\ &\equiv 0 \end{aligned}$$

Noch zu berechnen: $r_{\{q_0\}}(q_1, q_0)$

Wir berechnen: $r_{\{q_0\}}(q_1, q_0)$.

Mit $x = q_0$ erhalten wir

$$\begin{aligned} r_{\{q_0\}}(q_1, q_0) &= r_{\emptyset}(q_1, q_0) + r_{\emptyset}(q_1, q_0) r_{\emptyset}(q_0, q_0)^* r_{\emptyset}(q_0, q_0) \\ &\equiv \emptyset \end{aligned}$$

Wir haben damit folgende Werte für $r_{\{q_0\}}(q_i, q_j)$:

$$\begin{aligned} r_{\{q_0\}}(q_0, q_0) &\equiv 0^*, \\ r_{\{q_0\}}(q_0, q_1) &\equiv 0^* 1, \\ r_{\{q_0\}}(q_0, q_2) &\equiv \emptyset, \\ r_{\{q_0\}}(q_1, q_0) &\equiv \emptyset, \\ r_{\{q_0\}}(q_1, q_1) &\equiv \varepsilon, \\ r_{\{q_0\}}(q_1, q_2) &\equiv 0, \\ r_{\{q_0\}}(q_2, q_0) &\equiv 00^*, \\ r_{\{q_0\}}(q_2, q_1) &\equiv 0^* 1, \\ r_{\{q_0\}}(q_2, q_2) &\equiv \varepsilon. \end{aligned}$$

Daraus ergibt sich

$$\begin{aligned} r_{\{q_0, q_2\}}(q_0, q_0) &= r_{\{q_0\}}(q_0, q_0) + r_{\{q_0\}}(q_0, q_2) r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_0) \\ &\equiv 0^* + \emptyset \varepsilon^* 00^* \\ &\equiv 0^*. \end{aligned}$$

$$\begin{aligned} r_{\{q_0, q_2\}}(q_0, q_1) &= r_{\{q_0\}}(q_0, q_1) + r_{\{q_0\}}(q_0, q_2) r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_1) \\ &\equiv 0^* 1 \end{aligned}$$

$$\begin{aligned} r_{\{q_0, q_2\}}(q_1, q_1) &= r_{\{q_0\}}(q_1, q_1) + r_{\{q_0\}}(q_1, q_2) r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_1) \\ &\equiv \varepsilon + 0 \varepsilon^* 0^* 1 \\ &\equiv \varepsilon + 00^* 1 \end{aligned}$$

$$\begin{aligned} r_{\{q_0, q_2\}}(q_1, q_0) &= r_{\{q_0\}}(q_1, q_0) + r_{\{q_0\}}(q_1, q_2) r_{\{q_0\}}(q_2, q_2)^* r_{\{q_0\}}(q_2, q_0) \\ &\equiv \emptyset + 0 \varepsilon^* 00^* \\ &\equiv 000^* \end{aligned}$$

Daraus erhalten wir schließlich

$$\begin{aligned} r_Q(q_0, q_0) &= r_{\{q_0, q_2\}}(q_0, q_0) + r_{\{q_0, q_2\}}(q_0, q_1) r_{\{q_0, q_2\}}(q_1, q_1)^* r_{\{q_0, q_2\}}(q_1, q_0) \\ &\equiv 0^* + 0^* 1 (\varepsilon + 00^* 1)^* 000^* \end{aligned}$$

Dieser Ausdruck lässt sich noch vereinfachen zum äquivalenten Ausdruck $\varepsilon + (0 + 10)^* 0$.

Insgesamt haben wir also folgenden Satz bewiesen.

Insgesamt haben wir also folgenden Satz bewiesen.

Satz 4.19 (Äquivalenzsatz von Kleene)

Eine Sprache ist genau dann regulär, wenn sie FA-erkennbar ist.

Insgesamt haben wir also folgenden Satz bewiesen.

Satz 4.19 (Äquivalenzsatz von Kleene)

Eine Sprache ist genau dann regulär, wenn sie FA-erkennbar ist.

Von nun an sprechen wir nicht mehr von FA-erkennbaren Sprachen, sondern nur noch von **regulären Sprachen**.

Aus Satz 2.40 folgt, dass die Klasse der regulären Sprachen unter den Mengenoperationen, Verkettung und Iteration abgeschlossen ist.

Aus Satz 2.40 folgt, dass die Klasse der regulären Sprachen unter den Mengenoperationen, Verkettung und Iteration abgeschlossen ist.

Das hat zur Folge, dass wir die regulären Ausdrücke um zusätzliche Operatoren für Komplement und Durchschnitt erweitern könnten, **ohne dadurch zusätzliche Sprachen beschreiben zu können**.

Aus Satz 2.40 folgt, dass die Klasse der regulären Sprachen unter den Mengenoperationen, Verkettung und Iteration abgeschlossen ist.

Das hat zur Folge, dass wir die regulären Ausdrücke um zusätzliche Operatoren für Komplement und Durchschnitt erweitern könnten, **ohne dadurch zusätzliche Sprachen beschreiben zu können**.

Anders ausgedrückt: Komplement und Durchschnitt lassen sich mit den vorhandenen Operatoren für Vereinigung, Verkettung und Iteration simulieren.

(Das ist nicht offensichtlich!)

Anwendung: Textsuche mit regulären Ausdrücken

Wir können mit regulären Ausdrücken “Muster” spezifizieren, die wir in einem Text finden möchten.

Anwendung: Textsuche mit regulären Ausdrücken

Wir können mit regulären Ausdrücken “Muster” spezifizieren, die wir in einem Text finden möchten.

Beispiel: Der reguläre Ausdruck

$$(F + f)o(SAP + sap)$$

über dem Alphabet Σ_{ASCII} spezifiziert das Wort „FoSAP“ mit verschiedenen Groß-/Kleinschreibungsvarianten.

Anwendung: Textsuche mit regulären Ausdrücken

Wir können mit regulären Ausdrücken “Muster” spezifizieren, die wir in einem Text finden möchten.

Beispiel: Der reguläre Ausdruck

$$(F + f)o(SAP + sap)$$

über dem Alphabet Σ_{ASCII} spezifiziert das Wort „FoSAP“ mit verschiedenen Groß-/Kleinschreibungsvarianten.

In Unix gibt es Befehle wie `grep` und `egrep`, die die Textsuche mit regulären Ausdrücken direkt umsetzen. Weiterhin bieten viele Programmiersprachen (z.B. Perl) und Editoren (z.B. emacs) die Möglichkeit zur Textsuche mit regulären Ausdrücken.

Anwendung: Textsuche mit regulären Ausdrücken

Wir können mit regulären Ausdrücken “Muster” spezifizieren, die wir in einem Text finden möchten.

Beispiel: Der reguläre Ausdruck

$$(F + f)o(SAP + sap)$$

über dem Alphabet Σ_{ASCII} spezifiziert das Wort „FoSAP“ mit verschiedenen Groß-/Kleinschreibungsvarianten.

In Unix gibt es Befehle wie `grep` und `egrep`, die die Textsuche mit regulären Ausdrücken direkt umsetzen. Weiterhin bieten viele Programmiersprachen (z.B. Perl) und Editoren (z.B. emacs) die Möglichkeit zur Textsuche mit regulären Ausdrücken.

Compiler verwenden die Suche mit regulären Ausdrücken in der **lexikalischen Analyse**, um den Programmtext in sogenannte „Token“ zu zerlegen.

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmus:

1. Sei $r' := \left(\sum_{a \in \Sigma} a \right)^* r \left(\sum_{a \in \Sigma} a \right)^*$.

Dann gilt $T \in L(r') \iff T$ enthält ein $w \in L(r)$ als Infix.

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmus:

1. Sei $r' := \left(\sum_{a \in \Sigma} a \right)^* r \left(\sum_{a \in \Sigma} a \right)^*$.

Dann gilt $T \in L(r') \iff T$ enthält ein $w \in L(r)$ als Infix.

2. Berechne zu r' äquivalenten ε -NFA \mathcal{A}
3. Berechne zu \mathcal{A} äquivalenten NFA \mathcal{A}' .

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmus:

1. Sei $r' := \left(\sum_{a \in \Sigma} a \right)^* r \left(\sum_{a \in \Sigma} a \right)^*$.

Dann gilt $T \in L(r') \iff T$ enthält ein $w \in L(r)$ als Infix.

2. Berechne zu r' äquivalenten ε -NFA \mathcal{A}
3. Berechne zu \mathcal{A} äquivalenten NFA \mathcal{A}' .

Variante 1

4. Entscheide, ob $T \in L(\mathcal{A}')$.

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmus:

1. Sei $r' := \left(\sum_{a \in \Sigma} a\right)^* r \left(\sum_{a \in \Sigma} a\right)^*$.

Dann gilt $T \in L(r') \iff T$ enthält ein $w \in L(r)$ als Infix.

2. Berechne zu r' äquivalenten ε -NFA \mathcal{A}
3. Berechne zu \mathcal{A} äquivalenten NFA \mathcal{A}' .

Variante 1

4. Entscheide, ob $T \in L(\mathcal{A}')$.

Laufzeit $O(kn)$,

wobei k = Länge von r

und n = Länge von T .

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmus:

1. Sei $r' := \left(\sum_{a \in \Sigma} a\right)^* r \left(\sum_{a \in \Sigma} a\right)^*$.

Dann gilt $T \in L(r') \iff T$ enthält ein $w \in L(r)$ als Infix.

2. Berechne zu r' äquivalenten ε -NFA \mathcal{A}
3. Berechne zu \mathcal{A} äquivalenten NFA \mathcal{A}' .

Variante 1

4. Entscheide, ob $T \in L(\mathcal{A}')$.

Laufzeit $O(kn)$,
wobei k = Länge von r
und n = Länge von T .

Variante 2

4. Berechne zu \mathcal{A}' äquivalenten DFA \mathcal{A}'' .
5. Entscheide, ob $T \in L(\mathcal{A}'')$.

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmus:

1. Sei $r' := \left(\sum_{a \in \Sigma} a \right)^* r \left(\sum_{a \in \Sigma} a \right)^*$.

Dann gilt $T \in L(r') \iff T$ enthält ein $w \in L(r)$ als Infix.

2. Berechne zu r' äquivalenten ε -NFA \mathcal{A}
3. Berechne zu \mathcal{A} äquivalenten NFA \mathcal{A}' .

Variante 1

4. Entscheide, ob $T \in L(\mathcal{A}')$.

Laufzeit $O(kn)$,
wobei k = Länge von r
und n = Länge von T .

Variante 2

4. Berechne zu \mathcal{A}' äquivalenten DFA \mathcal{A}'' .
5. Entscheide, ob $T \in L(\mathcal{A}'')$.

Laufzeit $2^{O(k)} + O(n)$

Algorithmische Umsetzung

Eingabe: Text $T \subseteq \Sigma^*$ und regulärer Ausdruck $r \in \text{RA}_\Sigma$.

Aufgabe: Entscheide, ob T ein Wort aus $L(r)$ als Infix enthält.

Algorithmus:

1. Sei $r' := \left(\sum_{a \in \Sigma} a\right)^* r \left(\sum_{a \in \Sigma} a\right)^*$.

Dann gilt $T \in L(r') \iff T$ enthält ein $w \in L(r)$ als Infix.

2. Berechne zu r' äquivalenten ε -NFA \mathcal{A}
3. Berechne zu \mathcal{A} äquivalenten NFA \mathcal{A}' .

Variante 1

4. Entscheide, ob $T \in L(\mathcal{A}')$.

Laufzeit $O(kn)$,

wobei k = Länge von r

und n = Länge von T .

Variante 2

4. Berechne zu \mathcal{A}' äquivalenten DFA \mathcal{A}'' .

5. Entscheide, ob $T \in L(\mathcal{A}'')$.

Laufzeit $2^{O(k)} + O(n)$

Normalerweise ist n viel größer als k . Je nachdem wieviel ist die erste oder zweite Variante besser.

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$
- ▶ Punkt "." steht für ein beliebiges Zeichen

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$
- ▶ Punkt "." steht für ein beliebiges Zeichen
- ▶ Buchstabenmenge nach ASCII-Reihenfolge durch Angabe des ersten und letzten Buchstaben (z.B. $[a-z]$, $[a-z0-9]$)

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$
- ▶ Punkt "." steht für ein beliebiges Zeichen
- ▶ Buchstabenmenge nach ASCII-Reihenfolge durch Angabe des ersten und letzten Buchstaben (z.B. $[a-z]$, $[a-z0-9]$)
- ▶ $|$ statt $+$

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$
- ▶ Punkt "." steht für ein beliebiges Zeichen
- ▶ Buchstabenmenge nach ASCII-Reihenfolge durch Angabe des ersten und letzten Buchstaben (z.B. $[a-z]$, $[a-z0-9]$)
- ▶ $|$ statt $+$
- ▶ $r?$ statt $\varepsilon + r$

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$
- ▶ Punkt "." steht für ein beliebiges Zeichen
- ▶ Buchstabenmenge nach ASCII-Reihenfolge durch Angabe des ersten und letzten Buchstaben (z.B. $[a-z]$, $[a-z0-9]$)
- ▶ $|$ statt $+$
- ▶ $r?$ statt $\varepsilon + r$
- ▶ $r+$ statt $r^* r$

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$
- ▶ Punkt "." steht für ein beliebiges Zeichen
- ▶ Buchstabenmenge nach ASCII-Reihenfolge durch Angabe des ersten und letzten Buchstaben (z.B. $[a-z]$, $[a-z0-9]$)
- ▶ $|$ statt $+$
- ▶ $r?$ statt $\varepsilon + r$
- ▶ $r+$ statt $r^* r$
- ▶ $r\{4\}$ statt $rrrr$

Reguläre Ausdrücke in UNIX

Die wichtigsten Elemente der Unix-Syntax für reguläre Ausdrücke sind:

- ▶ $[a_1 a_2 \dots a_n]$ statt $a_1 + a_2 + \dots + a_n$
- ▶ Punkt "." steht für ein beliebiges Zeichen
- ▶ Buchstabenmenge nach ASCII-Reihenfolge durch Angabe des ersten und letzten Buchstaben (z.B. $[a-z]$, $[a-z0-9]$)
- ▶ $|$ statt $+$
- ▶ $r?$ statt $\varepsilon + r$
- ▶ $r+$ statt $r^* r$
- ▶ $r\{4\}$ statt $rrrr$

Im Detail verwenden allerdings verschiedene Unixbefehle und Programme verschiedene Varianten der Syntax.

Internationale Version der Aachener Telefonnummern

Internationale Version der Aachener Telefonnummern

(+49) 0241-9876543

+49-241-9876543

0049-241-9876543

Internationale Version der Aachener Telefonnummern

(+49) 0241-9876543

+49-241-9876543

0049-241-9876543

Regulärer Ausdruck in UNIX ERE (Extended Regular Expression) Syntax

`(\\(\\+49\\) 0|\\+49-|0049-)241-[1-9][0-9]*`

Abschnitt 4.2

Das Pumping-Lemma

Welche Sprachen sind regulär?

Alle?

Nein, denn es gibt mehr Sprachen als reguläre Ausdrücke.

Welche Sprachen sind regulär?

Alle?

Nein, denn es gibt mehr Sprachen als reguläre Ausdrücke.

Intuition

L ist regulär, wenn der Test auf Mitgliedschaft eines Wortes w in L durch buchstabenweises Durchlesen unter Zuhilfenahme eines beschränkten Speichers möglich ist.

Beispiel 4.21

$$L_1 := \{w \in \{a, b\}^* \mid w \text{ endet auf } b \iff w \text{ beginnt mit } a\}$$

Beispiel 4.21

$L_1 := \{w \in \{a, b\}^* \mid w \text{ endet auf } b \iff w \text{ beginnt mit } a\}$

Intuition: Regulär, denn man muss sich nur den ersten Buchstaben des Wortes merken und am Ende mit dem letzten vergleichen.
(Endlicher Speicher genügt.)

Beispiel 4.21

$$L_1 := \{w \in \{a, b\}^* \mid w \text{ endet auf } b \iff w \text{ beginnt mit } a\}$$

Intuition: Regulär, denn man muss sich nur den ersten Buchstaben des Wortes merken und am Ende mit dem letzten vergleichen.
(Endlicher Speicher genügt.)

Beispiel 4.22

$$L_2 := \{w \in \{0, 1\}^* \mid |w|_1 \geq 3748\}.$$

Beispiel 4.21

$$L_1 := \{w \in \{a, b\}^* \mid w \text{ endet auf } b \iff w \text{ beginnt mit } a\}$$

Intuition: Regulär, denn man muss sich nur den ersten Buchstaben des Wortes merken und am Ende mit dem letzten vergleichen.
(Endlicher Speicher genügt.)

Beispiel 4.22

$$L_2 := \{w \in \{0, 1\}^* \mid |w|_1 \geq 3748\}.$$

Intuition: Regulär, denn man muss sich die Zahl der 1en merken, bis man 3748 gesehen hat. Dazu reichen 3749 Zustände.

Beispiel 4.23

$$L_3 := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist durch } 9 \text{ teilbar}\}$$

Beispiel 4.23

$L_3 := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist durch } 9 \text{ teilbar}\}$

Intuition: Regulär, denn man muss sich bloss die Anzahl der bisher gesehenen 0en modulo 9 merken. Dazu reichen 9 Zustände.

Beispiel 4.23

$$L_3 := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist durch } 9 \text{ teilbar}\}$$

Intuition: Regulär, denn man muss sich bloss die Anzahl der bisher gesehenen 0en modulo 9 merken. Dazu reichen 9 Zustände.

Beispiel 4.24

$$L_4 := \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$$

Beispiel 4.23

$$L_3 := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist durch } 9 \text{ teilbar}\}$$

Intuition: Regulär, denn man muss sich bloss die Anzahl der bisher gesehenen 0en modulo 9 merken. Dazu reichen 9 Zustände.

Beispiel 4.24

$$L_4 := \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$$

Intuition: Nicht regulär, denn man muss sich die Differenz zwischen bislang gesehenen 1en und 0en merken. Dazu reicht ein endlicher Speicher nicht aus.

Beispiel 4.25

$$L_5 := \{ww \mid w \in \{a, b\}^*\}$$

Beispiel 4.25

$$L_5 := \{ww \mid w \in \{a, b\}^*\}$$

Intuition: Nicht regulär, denn man muss sich zumindest bis zur Mitte das ganze bisher gelesene Wort merken. Dazu reicht ein endlicher Speicher nicht aus.

Beispiel 4.25

$$L_5 := \{ww \mid w \in \{a, b\}^*\}$$

Intuition: Nicht regulär, denn man muss sich zumindest bis zur Mitte das ganze bisher gelesene Wort merken. Dazu reicht ein endlicher Speicher nicht aus.

Beispiel 4.26

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammerausdrücke (vgl. Beispiel 3.8).

Beispiel 4.25

$$L_5 := \{ww \mid w \in \{a, b\}^*\}$$

Intuition: Nicht regulär, denn man muss sich zumindest bis zur Mitte das ganze bisher gelesene Wort merken. Dazu reicht ein endlicher Speicher nicht aus.

Beispiel 4.26

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammerausdrücke (vgl. Beispiel 3.8).

Intuition: Nicht regulär, denn man muss sich zu jedem Zeitpunkt die Anzahl der bisher geöffneten, aber noch nicht geschlossenen Klammern merken. Dazu reicht ein endlicher Speicher nicht aus.

Beispiel 4.27

Die Sprache $\text{LIST}(\text{BIN}) \subseteq \Sigma_{\text{ASCII}}^*$ (vgl. Beispiel 3.7).

Beispiel 4.27

Die Sprache $\text{LIST}(\text{BIN}) \subseteq \Sigma_{\text{ASCII}}^*$ (vgl. Beispiel 3.7).

Intuition: Nicht regulär, denn man muss sich merken, wieviele Klammern schon aufgegangen sind, um sie wieder schließen zu können.

Beispiel 4.27

Die Sprache $\text{LIST}(\text{BIN}) \subseteq \Sigma_{\text{ASCII}}^*$ (vgl. Beispiel 3.7).

Intuition: Nicht regulär, denn man muss sich merken, wieviele Klammern schon aufgegangen sind, um sie wieder schließen zu können.

Beispiel 4.28

Die Sprache $\text{ART} \subseteq \Sigma_{\text{ASCII}}^*$ (vgl. Beispiel 3.20).

Beispiel 4.27

Die Sprache $\text{LIST}(\text{BIN}) \subseteq \Sigma_{\text{ASCII}}^*$ (vgl. Beispiel 3.7).

Intuition: Nicht regulär, denn man muss sich merken, wieviele Klammern schon aufgegangen sind, um sie wieder schließen zu können.

Beispiel 4.28

Die Sprache $\text{ART} \subseteq \Sigma_{\text{ASCII}}^*$ (vgl. Beispiel 3.20).

Intuition: Nicht regulär, denn auch hier muss man sich merken, wieviele Klammern schon aufgegangen sind, um sie wieder schließen zu können.

Beweis der Nicht-Regularität

Intuition ist schön und gut, aber wie lässt sich die Nicht-Regularität beweisen?

Behauptung

Die Sprache

$$L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$$

ist nicht regulär.

Beweis.

Annahme: L_4 ist regulär.

(Wir werden aus dieser Annahme einen Widerspruch herleiten.)

Sei $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ ein DFA mit $L(\mathcal{A}) = L_4$.

Sei $n := |Q|$ die Anzahl der Zustände von \mathcal{A} .

Wir betrachten das Wort

$$w = 0^n 1^n \in L_4 = L(\mathcal{A}).$$

Sei

$$(r_0, a, r_1, a, \dots, a, r_n, b, r_{n+1}, b, \dots, b, r_{2n})$$

der akzeptierende Lauf von \mathcal{A} auf w .

Unter den $(n+1)$ Zuständen r_0, \dots, r_n muss mindestens einer doppelt auftreten. Etwa

$$r_i = r_j, \quad \text{wobei } 1 \leq i < j \leq n$$

Dann ist auch

$$(r_0, a, r_1, a, \dots, a, r_i, a, r_{j+1}, a, \dots, a, r_n, b, r_{n+1}, b, \dots, b, r_{2n})$$

ein akzeptierender Lauf von \mathcal{A} .

Das akzeptierte Wort enthält allerdings weniger a s als b s, ist also nicht in L_4 .

Widerspruch!



Das Pumping Lemma (intuitiv)

Pumping Lemma

Ist L regulär, so lässt sich jedes hinreichend lange Wort aus L so dreiteilen, dass man den mittleren Teil streichen oder wiederholen kann, ohne L zu verlassen.

Das Pumping Lemma (intuitiv)

Pumping Lemma

Ist L regulär, so lässt sich jedes hinreichend lange Wort aus L so dreiteilen, dass man den mittleren Teil streichen oder wiederholen kann, ohne L zu verlassen.

Beweisidee

Hat ein Wort w mindestens so viele Buchstaben wie Zustände in einem NFA \mathcal{A} für L , wird in einem Lauf über w ein Zustand p wiederholt.

Das Pumping Lemma (intuitiv)

Pumping Lemma

Ist L regulär, so lässt sich jedes hinreichend lange Wort aus L so dreiteilen, dass man den mittleren Teil streichen oder wiederholen kann, ohne L zu verlassen.

Beweisidee

Hat ein Wort w mindestens so viele Buchstaben wie Zustände in einem NFA \mathcal{A} für L , wird in einem Lauf über w ein Zustand p wiederholt.

Daraus ergibt sich die gewünschte Dreiteilung von $w \in L$ in Segmente x, y, z :

$$\mathcal{A} : q_0 \xrightarrow{x} p \xrightarrow{y} p \xrightarrow{z} q \in F$$

Das Pumping Lemma (intuitiv)

Pumping Lemma

Ist L regulär, so lässt sich jedes hinreichend lange Wort aus L so dreiteilen, dass man den mittleren Teil streichen oder wiederholen kann, ohne L zu verlassen.

Beweisidee

Hat ein Wort w mindestens so viele Buchstaben wie Zustände in einem NFA \mathcal{A} für L , wird in einem Lauf über w ein Zustand p wiederholt. Daraus ergibt sich die gewünschte Dreiteilung von $w \in L$ in Segmente x, y, z :

$$\mathcal{A} : q_0 \xrightarrow{x} p \xrightarrow{y} p \xrightarrow{z} q \in F$$

Zum Beispiel gilt dann auch

$$\begin{aligned}\mathcal{A} : q_0 &\xrightarrow{x} p \xrightarrow{y} p \xrightarrow{y} p \xrightarrow{z} q \\ \mathcal{A} : q_0 &\xrightarrow{x} p \xrightarrow{z} q\end{aligned}$$

Das Pumping Lemma (intuitiv)

Pumping Lemma

Ist L regulär, so lässt sich jedes hinreichend lange Wort aus L so dreiteilen, dass man den mittleren Teil streichen oder wiederholen kann, ohne L zu verlassen.

Beweisidee

Hat ein Wort w mindestens so viele Buchstaben wie Zustände in einem NFA \mathcal{A} für L , wird in einem Lauf über w ein Zustand p wiederholt. Daraus ergibt sich die gewünschte Dreiteilung von $w \in L$ in Segmente x, y, z :

$$\mathcal{A} : q_0 \xrightarrow{x} p \xrightarrow{y} p \xrightarrow{z} q \in F$$

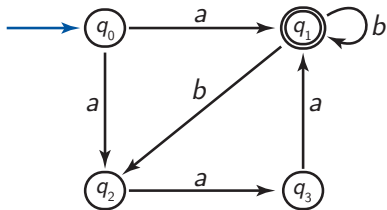
Zum Beispiel gilt dann auch

$$\begin{aligned}\mathcal{A} : q_0 &\xrightarrow{x} p \xrightarrow{y} p \xrightarrow{y} p \xrightarrow{z} q \\ \mathcal{A} : q_0 &\xrightarrow{x} p \xrightarrow{z} q\end{aligned}$$

Damit sind auch $xyyz$ und xz in L .

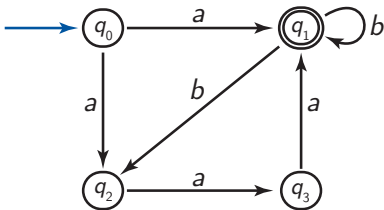
Beispiel 4.29

Sei $L = L(\mathcal{A})$ für folgenden NFA \mathcal{A} .



Beispiel 4.29

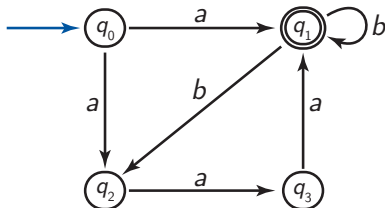
Sei $L = L(\mathcal{A})$ für folgenden NFA \mathcal{A} .



Wir betrachten das Wort $w = \textcolor{red}{aaabaabb}$ der Länge $|w| = 8 \geq 4 = |Q|$.

Beispiel 4.29

Sei $L = L(\mathcal{A})$ für folgenden NFA \mathcal{A} .



Wir betrachten das Wort $w = \textcolor{red}{aaabaabb}$ der Länge $|w| = 8 \geq 4 = |Q|$.

Es gilt $w \in L$; ein akzeptierender Lauf von \mathcal{A} auf w ist

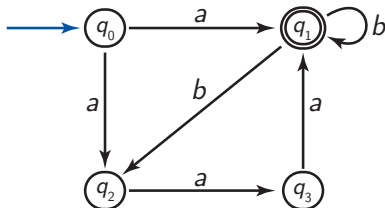
$$(q_0, \textcolor{red}{a}, q_2, \textcolor{red}{a}, q_3, \textcolor{red}{a}, q_1, \textcolor{red}{b}, q_2, \textcolor{red}{a}, q_3, \textcolor{red}{a}, q_1, \textcolor{red}{b}, q_1, \textcolor{red}{b}, q_1)$$

Also

$$\mathcal{A} : q_0 \xrightarrow{\textcolor{red}{a}} q_2 \xrightarrow{\textcolor{red}{aab}} q_2 \xrightarrow{\textcolor{red}{aabb}} q_1$$

Beispiel 4.29

Sei $L = L(\mathcal{A})$ für folgenden NFA \mathcal{A} .



Wir betrachten das Wort $w = \textcolor{red}{aaabaabb}$ der Länge $|w| = 8 \geq 4 = |Q|$.

Es gilt $w \in L$; ein akzeptierender Lauf von \mathcal{A} auf w ist

$$(q_0, \textcolor{red}{a}, q_2, \textcolor{red}{a}, q_3, \textcolor{red}{a}, q_1, \textcolor{red}{b}, q_2, \textcolor{red}{a}, q_3, \textcolor{red}{a}, q_1, \textcolor{red}{b}, q_1, \textcolor{red}{b}, q_1)$$

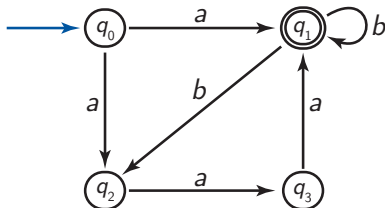
Also

$$\mathcal{A} : q_0 \xrightarrow{\textcolor{red}{a}} q_2 \xrightarrow{\textcolor{red}{aab}} q_2 \xrightarrow{\textcolor{red}{aabb}} q_1$$

Dreiteilung der Eingabe: $\textcolor{red}{a} \textcolor{red}{aab} \textcolor{red}{aabb}$.

Beispiel 4.29

Sei $L = L(\mathcal{A})$ für folgenden NFA \mathcal{A} .



Wir betrachten das Wort $w = \textcolor{red}{aaabaabb}$ der Länge $|w| = 8 \geq 4 = |Q|$.

Es gilt $w \in L$; ein akzeptierender Lauf von \mathcal{A} auf w ist

$$(q_0, \textcolor{red}{a}, q_2, \textcolor{red}{a}, q_3, \textcolor{red}{a}, q_1, \textcolor{red}{b}, q_2, \textcolor{red}{a}, q_3, \textcolor{red}{a}, q_1, \textcolor{red}{b}, q_1, \textcolor{red}{b}, q_1)$$

Also

$$\mathcal{A} : q_0 \xrightarrow{\textcolor{red}{a}} q_2 \xrightarrow{\textcolor{red}{aab}} q_2 \xrightarrow{\textcolor{red}{aabb}} q_1$$

Dreiteilung der Eingabe: $\textcolor{red}{a} \textcolor{red}{aab} \textcolor{red}{aabb}$.

Ebenfalls akzeptierte Wörter: $\textcolor{red}{a} \textcolor{red}{aabb}$ und $\textcolor{red}{a} \textcolor{red}{aab} \textcolor{red}{aab} \textcolor{red}{aabb}$

Lemma 4.30

Sei L regulär. Dann gibt es eine Zahl $n \geq 1$, so dass jedes Wort $w \in L$ mit $|w| \geq n$ zerlegbar ist in Wörter x, y, z mit

$$w = xyz,$$

die folgende Eigenschaften haben

1. $y \neq \varepsilon$,
2. $|xy| \leq n$,
3. $xy^kz \in L$ für alle $k \in \mathbb{N}$.

Sei $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ ein NFA mit $L(\mathcal{A}) = L$.

Wir setzen

$$n := |Q|.$$

Betrachte nun ein Wort $w = a_1 \dots a_m \in L$ der Länge $|w| = m \geq n$.

Sei

$$(r_0, a_1, r_1, \dots, a_m, r_m)$$

ein akzeptierender Lauf von \mathcal{A} auf w . Es gilt $r_0 = q_0$ und $r_m \in F$

Weil $m+1 > m \geq n$ gibt es i, j mit $0 \leq i < j \leq m$ so dass $r_i = r_j =: r$.

Sei $x := a_1 \dots a_i$ und $y := a_{i+1} \dots a_j$ und $z := a_{j+1} \dots a_m$.

Es gilt

(i) $y \neq \varepsilon$, weil $|y| = j - i > 0$.

(ii) $|xy| = j \leq n$.

Außerdem

$$\mathcal{A} : p_0 \xrightarrow{x} r \xrightarrow{y} r \xrightarrow{z} r_m \in F.$$

Aus $r \xrightarrow{y} r$ folgt $r \xrightarrow{y^k} r$ für alle $k \in \mathbb{N}$.

Also gilt auch

$$\mathcal{A} : p_0 \xrightarrow{x} r \xrightarrow{y^k} r \xrightarrow{z} r_m \in F$$

für alle $k \in \mathbb{N}$ und damit

(iii) $xy^kz \in L(\mathcal{A}) = L$ für alle $k \in \mathbb{N}$.



Beispiel 4.24 (Forts.)

Wir haben bereits gezeigt, dass die Sprache
 $L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ nicht regulär ist.

Beispiel 4.24 (Forts.)

Wir haben bereits gezeigt, dass die Sprache
 $L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ nicht regulär ist.

Hier ein **anderer Beweis** mit dem Pumping Lemma:

Beispiel 4.24 (Forts.)

Wir haben bereits gezeigt, dass die Sprache
 $L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ nicht regulär ist.

Hier ein **anderer Beweis** mit dem Pumping Lemma:

Angenommen, L_4 ist regulär.

Beispiel 4.24 (Forts.)

Wir haben bereits gezeigt, dass die Sprache
 $L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ nicht regulär ist.

Hier ein **anderer Beweis** mit dem Pumping Lemma:

Angenommen, L_4 ist regulär.

Wähle n zu L_4 gemäß Pumping Lemma und betrachte $w = 0^n 1^n \in L_4$.

Beispiel 4.24 (Forts.)

Wir haben bereits gezeigt, dass die Sprache
 $L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ nicht regulär ist.

Hier ein **anderer Beweis** mit dem Pumping Lemma:

Angenommen, L_4 ist regulär.

Wähle n zu L_4 gemäß Pumping Lemma und betrachte $w = 0^n 1^n \in L_4$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz = xy^0z \in L_4.$$

Beispiel 4.24 (Forts.)

Wir haben bereits gezeigt, dass die Sprache
 $L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ nicht regulär ist.

Hier ein **anderer Beweis** mit dem Pumping Lemma:

Angenommen, L_4 ist regulär.

Wähle n zu L_4 gemäß Pumping Lemma und betrachte $w = 0^n 1^n \in L_4$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz = xy^0z \in L_4.$$

Wegen $|xy| \leq n$ und $y \neq \varepsilon$ gilt $x = 0^j$ mit $j \geq 0$ und $y = 0^k$ mit $k > 0$.

Beispiel 4.24 (Forts.)

Wir haben bereits gezeigt, dass die Sprache
 $L_4 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ nicht regulär ist.

Hier ein **anderer Beweis** mit dem Pumping Lemma:

Angenommen, L_4 ist regulär.

Wähle n zu L_4 gemäß Pumping Lemma und betrachte $w = 0^n 1^n \in L_4$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz = xy^0z \in L_4.$$

Wegen $|xy| \leq n$ und $y \neq \varepsilon$ gilt $x = 0^j$ mit $j \geq 0$ und $y = 0^k$ mit $k > 0$.

Aber $xz = 0^{n-k}1^n \notin L_4$, weil $|xz|_0 = n - k < n = |xz|_1$.

Widerspruch!



Behauptung

Die Sprache $L_5 = \{ww \mid w \in \{a, b\}^\}$ ist nicht regulär.*

Behauptung

Die Sprache $L_5 = \{ww \mid w \in \{a, b\}^*\}$ ist nicht regulär.

Beweis.

Angenommen, L_5 ist regulär.

Behauptung

Die Sprache $L_5 = \{ww \mid w \in \{a, b\}^*\}$ ist nicht regulär.

Beweis.

Angenommen, L_5 ist regulär.

Wähle n zu L_5 gemäß Pumping Lemma und betrachte $w = 0^n 10^n 1 \in L_5$.

Behauptung

Die Sprache $L_5 = \{ww \mid w \in \{a, b\}^*\}$ ist nicht regulär.

Beweis.

Angenommen, L_5 ist regulär.

Wähle n zu L_5 gemäß Pumping Lemma und betrachte $w = 0^n 10^n 1 \in L_5$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz \in L_5.$$

Behauptung

Die Sprache $L_5 = \{ww \mid w \in \{a, b\}^*\}$ ist nicht regulär.

Beweis.

Angenommen, L_5 ist regulär.

Wähle n zu L_5 gemäß Pumping Lemma und betrachte $w = 0^n 10^n 1 \in L_5$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz \in L_5.$$

Wegen $|xy| \leq n$ und $y \neq \varepsilon$ gilt $x = 0^j$ mit $j \geq 0$ und $y = 0^k$ mit $k > 0$.

Behauptung

Die Sprache $L_5 = \{ww \mid w \in \{a, b\}^*\}$ ist nicht regulär.

Beweis.

Angenommen, L_5 ist regulär.

Wähle n zu L_5 gemäß Pumping Lemma und betrachte $w = 0^n 10^n 1 \in L_5$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz \in L_5.$$

Wegen $|xy| \leq n$ und $y \neq \varepsilon$ gilt $x = 0^j$ mit $j \geq 0$ und $y = 0^k$ mit $k > 0$.

Aber $xz = 0^{n-k} 10^n 1 \notin L_5$, weil $n - k < n$.

Widerspruch!



Behauptung

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammerausdrücke ist nicht regulär.

Behauptung

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammersausdrücke ist nicht regulär.

Beweis.

Angenommen, L_K ist regulär.

Behauptung

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammerausdrücke ist nicht regulär.

Beweis.

Angenommen, L_K ist regulär.

Wähle n zu L_K gemäß Pumping Lemma und betrachte $w = ({}^n)^n \in L_K$.

Behauptung

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammerausdrücke ist nicht regulär.

Beweis.

Angenommen, L_K ist regulär.

Wähle n zu L_K gemäß Pumping Lemma und betrachte $w = ({}^n)^n \in L_K$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz \in L_K.$$

Behauptung

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammerausdrücke ist nicht regulär.

Beweis.

Angenommen, L_K ist regulär.

Wähle n zu L_K gemäß Pumping Lemma und betrachte $w = ({}^n)^n \in L_K$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz \in L_K.$$

Wegen $|xy| \leq n$ und $y \neq \varepsilon$ gilt $x = ({}^j$ mit $j \geq 0$ und $y = ({}^k$ mit $k > 0$.

Behauptung

Die Sprache $L_K \subseteq \{ (,) \}$ der korrekten Klammerausdrücke ist nicht regulär.

Beweis.

Angenommen, L_K ist regulär.

Wähle n zu L_K gemäß Pumping Lemma und betrachte $w = ({}^n)^n \in L_K$.

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz \in L_K.$$

Wegen $|xy| \leq n$ und $y \neq \varepsilon$ gilt $x = ({}^j$ mit $j \geq 0$ und $y = ({}^k$ mit $k > 0$.

Aber $xz = ({}^{n-k})^n \notin L_K$, weil $n - k < n$.

Widerspruch!



Behauptung

Die Sprache $\text{ART} \subseteq \Sigma_{\text{ASCII}}^*$ ist nicht regulär.

Beweis.

Angenommen, ART ist regulär.

Wähle n zu ART gemäß Pumping Lemma und betrachte

$$w = ({}^n0(+0))^n = \underbrace{((\dots (0+0)+0) \dots +0)}_{n \text{ mal}} \underbrace{\dots +0)}_{n \text{ mal}} \in \text{ART}.$$

Beispiel: Für $n = 3$ wäre

$$w = (((0+0)+0)+0)$$

Pumping Lemma liefert Zerlegung

$$w = xyz \text{ mit } |xy| \leq n \text{ und } y \neq \varepsilon \text{ und } xz \in \text{ART}.$$

Wegen $|xy| \leq n$ und $y \neq \varepsilon$ gilt $x = ({}^j$ mit $j \geq 0$ und $y = ({}^k$ mit $k > 0$.

Aber $xz = ({}^{n-k}0(+0))^n \notin \text{ART}$, weil $n - k < n$.

Widerspruch!



Nichtreguläre Sprachen

Beispiele

- ▶ Menge der Wörter mit genau so vielen 0en wie 1en

Beispiele

- ▶ Menge der Wörter mit genau so vielen 0en wie 1en
- ▶ Menge der Wiederholungswörter ww .

Beispiele

- ▶ Menge der Wörter mit genau so vielen 0en wie 1en
- ▶ Menge der Wiederholungswörter ww .
- ▶ Menge der korrekten Klammerausdrücke.

Beispiele

- ▶ Menge der Wörter mit genau so vielen 0en wie 1en
- ▶ Menge der Wiederholungswörter ww .
- ▶ Menge der korrekten Klammerausdrücke.
- ▶ Listen (in der syntaktischen Form $LIST(BIN)$) und arithmetische Ausdrücke.

Beispiele

- ▶ Menge der Wörter mit genau so vielen 0en wie 1en
- ▶ Menge der Wiederholungswörter ww .
- ▶ Menge der korrekten Klammerausdrücke.
- ▶ Listen (in der syntaktischen Form $\text{LIST}(\text{BIN})$) und arithmetische Ausdrücke.

Merkregel

Endliche Automaten sind zu schwach für

- ▶ Worteigenschaften, deren Überprüfung eine Zählervariable mit Wertebereich \mathbb{N} erfordert,

Beispiele

- ▶ Menge der Wörter mit genau so vielen 0en wie 1en
- ▶ Menge der Wiederholungswörter ww .
- ▶ Menge der korrekten Klammerausdrücke.
- ▶ Listen (in der syntaktischen Form $\text{LIST}(\text{BIN})$) und arithmetische Ausdrücke.

Merkregel

Endliche Automaten sind zu schwach für

- ▶ Worteigenschaften, deren Überprüfung eine Zählervariable mit Wertebereich \mathbb{N} erfordert,
- ▶ Spezifikation der Wiederholung von Mustern beliebiger Länge,

Beispiele

- ▶ Menge der Wörter mit genau so vielen 0en wie 1en
- ▶ Menge der Wiederholungswörter ww .
- ▶ Menge der korrekten Klammerausdrücke.
- ▶ Listen (in der syntaktischen Form $\text{LIST}(\text{BIN})$) und arithmetische Ausdrücke.

Merkregel

Endliche Automaten sind zu schwach für

- ▶ Worteigenschaften, deren Überprüfung eine Zählervariable mit Wertebereich \mathbb{N} erfordert,
- ▶ Spezifikation der Wiederholung von Mustern beliebiger Länge,
- ▶ Mengen von Ausdrücken mit beliebiger Klammertiefe.

Abschnitt 4.3

Myhill-Nerode Äquivalenz

Unsere bisherigen Charakterisierung der regulären Sprachen beziehen sich auf verschiedene Formalismen (endliche Automaten, reguläre Ausdrücke) zur Berechnung bzw. Beschreibung der Sprachen.

Unsere bisherigen Charakterisierung der regulären Sprachen beziehen sich auf verschiedene Formalismen (endliche Automaten, reguläre Ausdrücke) zur Berechnung bzw. Beschreibung der Sprachen.

Ziel

Wir wollen reguläre Sprachen direkt anhand ihrer mathematischen Eigenschaften charakterisieren.

Unsere bisherigen Charakterisierung der regulären Sprachen beziehen sich auf verschiedene Formalismen (endliche Automaten, reguläre Ausdrücke) zur Berechnung bzw. Beschreibung der Sprachen.

Ziel

Wir wollen reguläre Sprachen direkt anhand ihrer mathematischen Eigenschaften charakterisieren.

Vorgehen

- ▶ Mit jeder Sprache über Σ assoziieren wir eine **Äquivalenzrelation** auf Σ^* , also eine Einteilung von Σ^* in Äquivalenzklassen.

Unsere bisherigen Charakterisierung der regulären Sprachen beziehen sich auf verschiedene Formalismen (endliche Automaten, reguläre Ausdrücke) zur Berechnung bzw. Beschreibung der Sprachen.

Ziel

Wir wollen reguläre Sprachen direkt anhand ihrer mathematischen Eigenschaften charakterisieren.

Vorgehen

- ▶ Mit jeder Sprache über Σ assoziieren wir eine **Äquivalenzrelation** auf Σ^* , also eine Einteilung von Σ^* in Äquivalenzklassen.
- ▶ Es wird sich herausstellen, dass die regulären Sprachen gerade die Sprachen mit nur endlich vielen Äquivalenzklassen sind.

Definition 4.31

Eine **Äquivalenzrelation** auf einer Menge X ist eine zweistellige Relation \sim auf X , so dass für alle $x, y, z \in X$ gilt:

- (i) $x \sim x$ (\sim ist **reflexiv**);
- (ii) wenn $x \sim y$, dann $y \sim x$ (\sim ist **symmetrisch**);
- (iii) wenn $x \sim y$ und $y \sim z$, dann $x \sim z$ (\sim ist **transitiv**).

Definition 4.31

Eine **Äquivalenzrelation** auf einer Menge X ist eine zweistellige Relation \sim auf X , so dass für alle $x, y, z \in X$ gilt:

- (i) $x \sim x$ (\sim ist **reflexiv**);
- (ii) wenn $x \sim y$, dann $y \sim x$ (\sim ist **symmetrisch**);
- (iii) wenn $x \sim y$ und $y \sim z$, dann $x \sim z$ (\sim ist **transitiv**).

Beispiele 4.32

1. Äquivalenzrelation \sim_1 auf \mathbb{N} : $x \sim_1 y \iff x \equiv y \pmod{7}$.

Definition 4.31

Eine **Äquivalenzrelation** auf einer Menge X ist eine zweistellige Relation \sim auf X , so dass für alle $x, y, z \in X$ gilt:

- (i) $x \sim x$ (\sim ist **reflexiv**);
- (ii) wenn $x \sim y$, dann $y \sim x$ (\sim ist **symmetrisch**);
- (iii) wenn $x \sim y$ und $y \sim z$, dann $x \sim z$ (\sim ist **transitiv**).

Beispiele 4.32

1. Äquivalenzrelation \sim_1 auf \mathbb{N} : $x \sim_1 y \iff x \equiv y \pmod{7}$.
2. Äquivalenzrelation \sim_2 auf Σ^* : $x \sim_2 y \iff |x| = |y|$.

Definition 4.31

Eine **Äquivalenzrelation** auf einer Menge X ist eine zweistellige Relation \sim auf X , so dass für alle $x, y, z \in X$ gilt:

- (i) $x \sim x$ (\sim ist **reflexiv**);
- (ii) wenn $x \sim y$, dann $y \sim x$ (\sim ist **symmetrisch**);
- (iii) wenn $x \sim y$ und $y \sim z$, dann $x \sim z$ (\sim ist **transitiv**).

Beispiele 4.32

1. Äquivalenzrelation \sim_1 auf \mathbb{N} : $x \sim_1 y \iff x \equiv y \pmod{7}$.
2. Äquivalenzrelation \sim_2 auf Σ^* : $x \sim_2 y \iff |x| = |y|$.
3. Äquivalenzrelation \sim_3 auf der Menge der DFAs mit Alphabet Σ :

$$\mathcal{A} \sim_3 \mathcal{B} \iff L(\mathcal{A}) = L(\mathcal{B}).$$

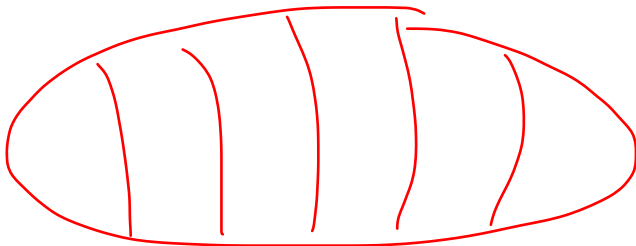
Definition 4.33

Eine **Partition** einer Menge X ist eine Menge $\mathcal{P} \subseteq 2^X$ von nichtleeren Teilmengen von X , so dass

(i) $P \cap Q = \emptyset$ für alle $P, Q \in \mathcal{P}$ mit $P \neq Q$;

(ii) $\bigcup_{P \in \mathcal{P}} P = X$.

Die Mengen $P \in \mathcal{P}$ nennen wir die **Teile** oder **Klassen** der Partition \mathcal{P} .



Definition 4.33

Eine **Partition** einer Menge X ist eine Menge $\mathcal{P} \subseteq 2^X$ von nichtleeren Teilmengen von X , so dass

(i) $P \cap Q = \emptyset$ für alle $P, Q \in \mathcal{P}$ mit $P \neq Q$;

(ii) $\bigcup_{P \in \mathcal{P}} P = X$.

Die Mengen $P \in \mathcal{P}$ nennen wir die **Teile** oder **Klassen** der Partition \mathcal{P} .

Beispiele 4.34

1. Partition von \mathbb{N} in gerade und ungerade Zahlen.

Definition 4.33

Eine **Partition** einer Menge X ist eine Menge $\mathcal{P} \subseteq 2^X$ von nichtleeren Teilmengen von X , so dass

(i) $P \cap Q = \emptyset$ für alle $P, Q \in \mathcal{P}$ mit $P \neq Q$;

(ii) $\bigcup_{P \in \mathcal{P}} P = X$.

Die Mengen $P \in \mathcal{P}$ nennen wir die **Teile** oder **Klassen** der Partition \mathcal{P} .

Beispiele 4.34

1. Partition von \mathbb{N} in gerade und ungerade Zahlen.
2. Partition von Σ^* in Wörter gleicher Länge.

Exkurs (Forts.): Äquivalenzrelationen und Partitionen

Definition 4.35

Sei \sim eine Äquivalenzrelation auf einer Menge X . Die Äquivalenzklasse eines Elementes $x \in X$ ist die Menge

$$x/\sim := \{y \in X \mid y \sim x\}.$$

Eigenschaft: $x \in x/\sim$

$$x \in y/\sim \Rightarrow y \in x/\sim$$

Exkurs (Forts.): Äquivalenzrelationen und Partitionen

Definition 4.35

Sei \sim eine Äquivalenzrelation auf einer Menge X . Die **Äquivalenzklasse** eines Elementes $x \in X$ ist die Menge

$$x/\sim := \{y \in X \mid y \sim x\}.$$

Satz 4.36

Sei X eine Menge.

1. Ist \sim eine Äquivalenzrelation auf X , so ist $\{x/\sim \mid x \in X\}$ eine Partition von X .

Exkurs (Forts.): Äquivalenzrelationen und Partitionen

Definition 4.35

Sei \sim eine Äquivalenzrelation auf einer Menge X . Die **Äquivalenzklasse** eines Elementes $x \in X$ ist die Menge

$$x/\sim := \{y \in X \mid y \sim x\}.$$

Satz 4.36

Sei X eine Menge.

1. Ist \sim eine Äquivalenzrelation auf X , so ist $\{x/\sim \mid x \in X\}$ eine Partition von X .
2. Ist \mathcal{P} eine Partition von X , so ist $\sim_{\mathcal{P}}$ mit

$$x \sim_{\mathcal{P}} y \iff x, y \in P \text{ für ein } P \in \mathcal{P}$$

eine Äquivalenzrelation auf X , und es gilt $\mathcal{P} = \{x/\sim_{\mathcal{P}} \mid x \in X\}$.

Exkurs (Forts.): Beispiele 4.37

1. Die Äquivalenzrelation \sim_1 auf \mathbb{N} mit

$$x \sim_1 y \iff x \equiv y \pmod{2}$$

entspricht der Partition in die geraden und ungeraden Zahlen.

Exkurs (Forts.): Beispiele 4.37

1. Die Äquivalenzrelation \sim_1 auf \mathbb{N} mit

$$x \sim_1 y \iff x \equiv y \pmod{2}$$

entspricht der Partition in die geraden und ungeraden Zahlen.

2. Die Äquivalenzrelation \sim_2 auf Σ^* mit

$$v \sim_2 w \iff |v| \equiv |w|$$

entspricht der Partition von Σ^* in Wörter gleicher Länge.

Definition 4.38

Der **Index** einer Äquivalenzrelation ist die Anzahl ihrer Äquivalenzklassen (eine positive natürliche Zahl oder „unendlich“).

Definition 4.38

Der **Index** einer Äquivalenzrelation ist die Anzahl ihrer Äquivalenzklassen (eine positive natürliche Zahl oder „unendlich“).

Notation

- ▶ Wir bezeichnen den Index von \sim durch **index**(\sim).

Definition 4.38

Der **Index** einer Äquivalenzrelation ist die Anzahl ihrer Äquivalenzklassen (eine positive natürliche Zahl oder „unendlich“).

Notation

- ▶ Wir bezeichnen den Index von \sim durch **index**(\sim).
- ▶ ∞ steht für „unendlich“.

Definition 4.38

Der **Index** einer Äquivalenzrelation ist die Anzahl ihrer Äquivalenzklassen (eine positive natürliche Zahl oder „unendlich“).

Notation

- ▶ Wir bezeichnen den Index von \sim durch $\text{index}(\sim)$.
- ▶ ∞ steht für „unendlich“.

Beispiele 4.39

1. Der Index der Äquivalenzrelation $(x \sim_1 y \iff x \equiv y \pmod{2})$ auf \mathbb{N} ist 2.

Definition 4.38

Der **Index** einer Äquivalenzrelation ist die Anzahl ihrer Äquivalenzklassen (eine positive natürliche Zahl oder „unendlich“).

Notation

- ▶ Wir bezeichnen den Index von \sim durch $\text{index}(\sim)$.
- ▶ ∞ steht für „unendlich“.

Beispiele 4.39

1. Der Index der Äquivalenzrelation $(x \sim_1 y \iff x \equiv y \pmod{2})$ auf \mathbb{N} ist 2.
2. Der Index der Äquivalenzrelation $(v \sim_2 w \iff |v| \equiv |w|)$ auf Σ^* ist ∞ .

Die Myhill-Nerode-Äquivalenz

Definition 4.40

Sei $L \subseteq \Sigma^*$ eine Sprache. Wir definieren die Äquivalenzrelation \sim_L auf Σ^* durch

$$v \sim_L w \quad :\Longleftrightarrow \quad \text{für alle } x \in \Sigma^* : vx \in L \Leftrightarrow wx \in L,$$

für alle $v, w \in \Sigma^*$.

Die Myhill-Nerode-Äquivalenz

Definition 4.40

Sei $L \subseteq \Sigma^*$ eine Sprache. Wir definieren die Äquivalenzrelation \sim_L auf Σ^* durch

$$v \sim_L w \quad :\Longleftrightarrow \quad \text{für alle } x \in \Sigma^* : vx \in L \Leftrightarrow wx \in L,$$

für alle $v, w \in \Sigma^*$.

Lemma 4.41

\sim_L ist eine Äquivalenzrelation auf Σ^* .

Die Myhill-Nerode-Äquivalenz

Definition 4.40

Sei $L \subseteq \Sigma^*$ eine Sprache. Wir definieren die Äquivalenzrelation \sim_L auf Σ^* durch

$$v \sim_L w \quad :\Longleftrightarrow \quad \text{für alle } x \in \Sigma^* : vx \in L \Leftrightarrow wx \in L,$$

für alle $v, w \in \Sigma^*$.

Lemma 4.41

\sim_L ist eine Äquivalenzrelation auf Σ^* .

Notation

- $w/_L$ (statt w/\sim_L) steht für die \sim_L -Äquivalenzklasse von w .

Die Myhill-Nerode-Äquivalenz

Definition 4.40

Sei $L \subseteq \Sigma^*$ eine Sprache. Wir definieren die Äquivalenzrelation \sim_L auf Σ^* durch

$$v \sim_L w \quad :\Longleftrightarrow \quad \text{für alle } x \in \Sigma^* : vx \in L \Leftrightarrow wx \in L,$$

für alle $v, w \in \Sigma^*$.

Lemma 4.41

\sim_L ist eine Äquivalenzrelation auf Σ^* .

Notation

- ▶ $w/_L$ (statt w/\sim_L) steht für die \sim_L -Äquivalenzklasse von w .
- ▶ $\Sigma^*/_L := \{w/_L \mid w \in \Sigma^*\}$ ist die Menge aller \sim_L -Äquivalenzklassen.

Die Myhill-Nerode-Äquivalenz

Definition 4.40

Sei $L \subseteq \Sigma^*$ eine Sprache. Wir definieren die Äquivalenzrelation \sim_L auf Σ^* durch

$$v \sim_L w \quad :\Longleftrightarrow \quad \text{für alle } x \in \Sigma^* : vx \in L \Leftrightarrow wx \in L,$$

für alle $v, w \in \Sigma^*$.

Lemma 4.41

\sim_L ist eine Äquivalenzrelation auf Σ^* .

Notation

- ▶ $w/_L$ (statt w/\sim_L) steht für die \sim_L -Äquivalenzklasse von w .
- ▶ $\Sigma^*/_L := \{w/_L \mid w \in \Sigma^*\}$ ist die Menge aller \sim_L -Äquivalenzklassen.
- ▶ $\text{index}(L) := \text{index}(\sim_L)$ ist der Index von \sim_L .

Beweis von Lemma 4.41

Reflexivität: Sei $v \in \Sigma^*$.

Trivialerweise gilt für alle $x \in \Sigma^*$: $vx \in L \iff vx \in L$.

Also $v \sim_L v$.

Symmetrie: Seien $v, w \in \Sigma^*$, so dass $v \sim_L w$.

Dann gilt für alle $x \in \Sigma^*$: $(vx \in L \iff wx \in L)$, und
damit auch $(wx \in L \iff vx \in L)$.

Also $w \sim_L v$.

Transitivität: Seien $u, v, w \in \Sigma^*$, so dass $u \sim_L v$ und $v \sim_L w$.

Dann gilt für alle $x \in \Sigma^*$: $(ux \in L \iff vx \in L)$ und
 $(vx \in L \iff wx \in L)$.

Damit gilt auch $(ux \in L \iff wx \in L)$.

Also $u \sim_L w$.



Sei

$$L := \{w \in \{a, b\}^* \mid w \text{ hat Infix } ab\}$$

Wir beginnen mit einigen Beobachtungen zu \sim_L :

- ▶ $\varepsilon \not\sim_L a$ (mit $x = b$, denn $\varepsilon b \notin L$, aber $ab \in L$)
- ▶ $\varepsilon \sim_L b$
- ▶ $a \sim_L aa$, denn:
 - $ax \in L$
 - $\iff x \text{ hat Infix } ab \text{ oder } x \text{ beginnt mit } b$
 - $\iff aax \in L$
- ▶ $a \not\sim_L ab$
- ▶ $\varepsilon \not\sim_L ab$

Behauptung

Die \sim_L -Klassen sind:

$$\varepsilon/L, \quad a/L, \quad ab/L$$

Beweis.

Wir haben bereits gezeigt, dass $\varepsilon \not\sim_L a$ und $\varepsilon \not\sim_L ab$ und $a \not\sim_L ab$.

Also sind die drei Äquivalenzklassen ε/L , a/L und ab/L verschieden.

Wir müssen noch zeigen, dass dies alle Äquivalenzklassen sind.

Sei dazu $v \in \{a, b\}^*$ beliebig. Wir zeigen, dass $v \sim_L \varepsilon$ oder $v \sim_L a$ oder $v \sim_L ab$.

1. Fall: v enthält ab als Infix.

Dann gilt $vx \in L$ für alle $x \in \Sigma^*$.

Weil auch $abx \in L$ für alle $x \in \Sigma^*$, folgt

$$v \sim_L ab.$$

2. Fall: v enthält ab nicht als Infix, und v endet mit a .

Dann gilt für alle $x \in \Sigma^*$:

$$\begin{aligned} vx \in L &\iff x \text{ beginnt mit } b \text{ oder enthält } ab \text{ als Infix} \\ &\iff ax \in L. \end{aligned}$$

Also

$$v \sim_L a.$$

3. Fall: v enthält ab nicht als Infix, und v endet nicht mit a .

Dann gilt für alle $x \in \Sigma^*$:

$$\begin{aligned} vx \in L &\iff x \text{ enthält } ab \text{ als Infix} \\ &\iff x \in L. \end{aligned}$$

Also

$$v \sim_L \varepsilon.$$

□

Aus der Behauptung folgt $\text{index}(L) = 3$.

Sei

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

Wir wissen aus Beispiel 4.24, dass L nicht regulär ist.

Wieder beginnen wir mit einigen Beobachtungen zu \sim_L :

- ▶ $\varepsilon \not\sim_L 1$, denn $\varepsilon \in L$ und $1 \notin L$.
- ▶ $0 \not\sim_L 000$, denn $01 \in L$ und $0001 \notin L$.

Behauptung

Für alle $i, j \in \mathbb{N}$ mit $i < j$ gilt

$$0^i \not\sim_L 0^j.$$

Beweis.

Wir wählen $x := 1^i$. Dann gilt $0^i x \in L$ und $0^j x \notin L$. □

Aus der Behauptung folgt sofort, dass $\text{index}(L) = \infty$.

Satz 4.44

Eine Sprache L ist genau dann regulär, wenn $\text{index}(L) < \infty$.

Als eine erste Anwendung des Satzes von Nerode zeigen wir, dass die Sprache $\text{LIST}(\text{BIN}) \subseteq \Sigma_{\text{ASCII}}^*$ aus den Beispielen 3.7 und 4.27 nicht regulär ist.

Wir zeigen:

$$\text{index}(\text{LIST}(\text{BIN})) = \infty \quad (*)$$

Dann folgt aus dem Satz von Nerode, dass $\text{LIST}(\text{BIN})$ nicht regulär ist.

Für alle $i \in \mathbb{N}$ sei

$$v_i := (\text{cons}(0,))^n.$$

Beispiel:

$$v_3 = \text{cons}(0, \text{cons}(0, \text{cons}(0,$$

Behauptung

Für alle $i, j \in \mathbb{N}$ mit $i < j$ gilt

$$v_i \not\sim_{\text{LIST}(\text{BIN})} v_j$$

Beweis.

Wir wählen

$$x := \text{nil})^i$$

Dann gilt

$$v_i x = \underbrace{\text{cons}(0, \text{cons}(0, \dots \text{cons}(0, \text{nil}))}_{i \text{ mal}}) \dots) \in \text{LIST}(\text{BIN})$$

und

$$v_j x = \underbrace{\text{cons}(0, \text{cons}(0, \dots \text{cons}(0, \text{nil}))}_{j \text{ mal}}) \dots) \notin \text{LIST}(\text{BIN}).$$

□

Aus der Behauptung folgt $\text{index}(\text{LIST}(\text{BIN})) = \infty$.

Lemma 4.46

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $L = L(\mathcal{A})$. Dann gilt

$$\text{index}(L) \leq |Q|.$$

Lemma 4.46

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $L = L(\mathcal{A})$. Dann gilt

$$\text{index}(L) \leq |Q|.$$

Korollar 4.47

Für jede reguläre Sprache L gilt $\text{index}(L) < \infty$.

Lemma 4.46

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $L = L(\mathcal{A})$. Dann gilt

$$\text{index}(L) \leq |Q|.$$

Beweis.

Für alle $v \in \Sigma^*$ sei $q_v \in Q$ der eindeutige Zustand mit

$$\mathcal{A} : q_0 \xrightarrow{v} q_v.$$

Behauptung 1

Seien $v, w \in \Sigma^*$ mit $q_v = q_w$. Dann gilt

$$v \sim_L w.$$

Beweis.

Wir müssen zeigen, dass für alle $x \in \Sigma^*$ gilt:

$$vx \in L \iff wx \in L. \quad (*)$$

Sei also $x \in \Sigma^*$.

Es gilt

$$\begin{aligned} vx \in L = L(\mathcal{A}) &\iff \exists q \in F : \mathcal{A} : q_0 \xrightarrow{vx} q \\ &\iff \exists q \in F : \mathcal{A} : q_v \xrightarrow{x} q. \end{aligned} \quad (**)$$

und

$$\begin{aligned} wx \in L = L(\mathcal{A}) &\iff \exists q \in F : \mathcal{A} : q_0 \xrightarrow{wx} q \\ &\iff \exists q \in F : \mathcal{A} : q_w \xrightarrow{x} q. \end{aligned} \quad (***)$$

Weil $q_v = q_w$ sind die Bedingungen (**) und (***) identisch, also gilt (*).

Damit ist die Behauptung bewiesen. □

Durch Negation folgt aus der Behauptung

$$v \not\sim_L w \implies q_v \neq q_w.$$

Es gibt also höchstens soviele Äquivalenzklassen wie Zustände. □

Um den Beweis des Satzes von Nerode abzuschließen, müssen wir noch zeigen:

$$\text{index}(L) < \infty \implies L \text{ ist regulär.}$$

Um den Beweis des Satzes von Nerode abzuschließen, müssen wir noch zeigen:

$$\text{index}(L) < \infty \implies L \text{ ist regulär.}$$

Idee

Wir verwenden die Äquivalenzklassen von \sim_L als Zustände eines DFAs für L .

Lemma 4.48

Sei $L \subseteq \Sigma^*$. Dann gilt für alle $u, v, w \in \Sigma^*$:

$$u \sim_L v \quad \implies \quad uw \sim_L vw.$$

Lemma 4.48

Sei $L \subseteq \Sigma^*$. Dann gilt für alle $u, v, w \in \Sigma^*$:

$$u \sim_L v \implies uw \sim_L vw.$$

Beweis.

Es gelte $u \sim_L v$. Dann gilt für alle $x \in \Sigma^*$:

$$(uw)x \in L \iff u(wx) \in L \iff v(wx) \in L \iff (vw)x \in L.$$

Also gilt $uw \sim_L vw$. □

Definition 4.49

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Der **Myhill-Nerode DFA** für L ist das Tupel

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

Definition 4.49

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Der **Myhill-Nerode DFA** für L ist das Tupel

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

► $Q_L := \Sigma^* / L = \{v/L \mid v \in \Sigma^*\},$

Definition 4.49

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Der **Myhill-Nerode DFA** für L ist das Tupel

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

- ▶ $Q_L := \Sigma^* / L = \{v/L \mid v \in \Sigma^*\},$
- ▶ $\delta_L : Q_L \times \Sigma \rightarrow Q_L$ definiert durch

$$\delta_L(v/L, a) = va/L,$$

Definition 4.49

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Der **Myhill-Nerode DFA** für L ist das Tupel

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

- ▶ $Q_L := \Sigma^* / L = \{v/L \mid v \in \Sigma^*\},$
- ▶ $\delta_L : Q_L \times \Sigma \rightarrow Q_L$ definiert durch

$$\delta_L(v/L, a) = va/L,$$

- ▶ $q_{L0} := \varepsilon/L,$

Definition 4.49

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Der **Myhill-Nerode DFA** für L ist das Tupel

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

- ▶ $Q_L := \Sigma^*/L = \{v/L \mid v \in \Sigma^*\},$
- ▶ $\delta_L : Q_L \times \Sigma \rightarrow Q_L$ definiert durch

$$\delta_L(v/L, a) = va/L,$$

- ▶ $q_{L0} := \varepsilon/L,$
- ▶ $F_L := \{v/L \mid v/L \cap L \neq \emptyset\}.$

Definition 4.49

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Der **Myhill-Nerode DFA** für L ist das Tupel

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

- ▶ $Q_L := \Sigma^* / L = \{v/L \mid v \in \Sigma^*\},$
- ▶ $\delta_L : Q_L \times \Sigma \rightarrow Q_L$ definiert durch

$$\delta_L(v/L, a) = va/L,$$

- ▶ $q_{L0} := \varepsilon/L,$
- ▶ $F_L := \{v/L \mid v/L \cap L \neq \emptyset\}.$

Lemma 4.50

\mathcal{A}_L ist ein DFA.

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

- ▶ $Q_L := \Sigma^* /_L = \{v/_L \mid v \in \Sigma^*\},$
- ▶ $\delta_L : Q_L \times \Sigma \rightarrow Q_L$ definiert durch

$$\delta_L(v/_L, a) = va/_L,$$

- ▶ $q_{L0} := \varepsilon/_L,$
- ▶ $F_L := \{v/_L \mid v/_L \cap L \neq \emptyset\}.$

Was müssen wir überhaupt beweisen?

- ▶ Q_L ist eine endliche Menge, weil $\text{index}(L) < \infty.$
- ▶ Σ ist ein Alphabet.
- ▶ Offensichtlich ist $q_{L0} \in Q_L$ und $F_L \subseteq Q_L.$
- ▶ Wir müssen einzig zeigen, dass δ_L wohldefiniert ist.

Das bedeutet, dass der Wert $\delta_L(v/_L, a)$ nicht von der Wahl des Repräsentanten v , sondern nur von der Äquivalenzklasse $v/_L$ abhängt.

Anders ausgedrückt: Ist $v/_L = v'/_L$, so gilt

$$\delta_L(v/_L, a) = va/_L = v'a/_L = \delta_L(v'/_L, a).$$

Das folgt sofort aus dem Kongruenzlemma 4.46, nach dem

$$v \sim_L v' \implies va \sim_L v'a \text{ gilt.}$$



Lemma 4.51

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Dann gilt

$$L = L(\mathcal{A}_L).$$

Lemma 4.51

Sei $L \subseteq \Sigma^*$ mit $\text{index}(L) < \infty$. Dann gilt

$$L = L(\mathcal{A}_L).$$

Das Lemma zeigt, dass jede Sprache mit endlichem Index DFA-erkennbar, also regulär ist. Damit ist der Satz von Nerode bewiesen.

Behauptung 1

Für alle $w \in \Sigma^*$ gilt:

$$\mathcal{A}_L : q_{L0} \xrightarrow{w} w/L.$$

Beweis.

Induktion über den Aufbau von $w \in \Sigma^*$.

Induktionsanfang: $w = \varepsilon$

Es gilt $\mathcal{A}_L : q_{L0} \xrightarrow{\varepsilon} q_{L0} = \varepsilon/L$.

Induktionsschritt: $w \rightarrow wa$ für $w \in \Sigma^*$, $a \in \Sigma$.

Sei $q \in Q_L$, so dass $\mathcal{A}_L : q_{L0} \xrightarrow{wa} q$.

Sei $q' \in Q_L$, so dass $\mathcal{A}_L : q_{L0} \xrightarrow{w} q' \xrightarrow{a} q$. Dann gilt $q = \delta_L(q', a)$.

Nach Induktionsannahme gilt $q' = w/L$ und damit

$$q = \delta_L(q', a) = \delta_L(w/L, a) = wa/L.$$

□

Behauptung 2

Für alle $w \in \Sigma^*$ gilt

$$w \in L \iff w/L \in F_L.$$

Beweis.

„ \implies “: $w \in L \implies w/L \cap L \neq \emptyset \implies w/L \in F_L$.

„ \impliedby “: Sei $w/L \in F_L$. Dann gilt $w/L \cap L \neq \emptyset$. Sei $w' \in w/L \cap L$. Dann gilt $w \sim_L w'$ und damit

$$w' = w'\varepsilon \in L \implies w = w\varepsilon \in L.$$

□

Um das Lemma zu beweisen, müssen wir zeigen: Für alle $w \in \Sigma^*$ gilt

$$\mathcal{A}_L \text{ akzeptiert } w \iff w \in L.$$

Sei $w \in \Sigma^*$. Es gilt

$$\begin{aligned} \mathcal{A}_L \text{ akzeptiert } w &\iff \exists q \in F_L : \mathcal{A}_L : q_{L0} \xrightarrow{w} q \\ &\iff w/L \in F_L && \text{(Behauptung 1)} \\ &\iff w \in L && \text{(Behauptung 2).} \end{aligned}$$

□

Sei $L \subseteq \{a, b\}^*$ ist die Sprache, die aus allen Wörtern mit Infix ab oder Suffix ba besteht.

Myhill-Nerode Äquivalenzklassen für L :

ab/L alle Wörter mit Infix ab

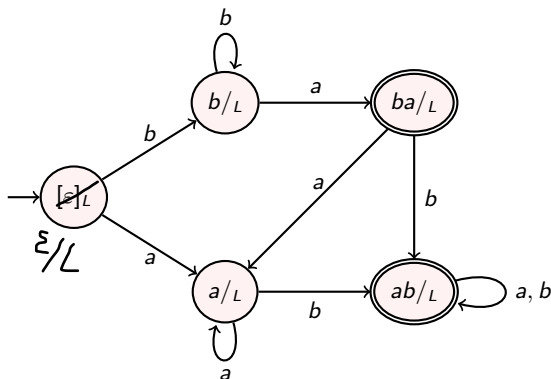
ba/L alle Wörter ohne Infix ab und mit Suffix ba

a/L alle Wörter ohne Infix ab und ohne Suffix ba , die auf a enden

b/L alle Wörter ohne Infix ab , die auf b enden

ε/L das leere Wort

Myhill-Nerode DFA \mathcal{A}_L



Bemerkung 4.53

Wir haben es offen gelassen, wie wir die Myhill-Nerode Äquivalenzklassen bestimmt haben, und es ist auch nicht ganz einfach, dies zu tun.

Im nächsten Kapitel werden wir ein Verfahren kennen lernen, den Myhill-Nerode DFA effizient zu berechnen.