

Einfuehrung in die Theoretische Informatik

Kyriakos Schwarz,Roland Hediger

HS 2014

Contents

1	Erste Woche	1
1.1	Sprachen	1
1.2	Endliche Automaten DFA	5
2	Zweite Woche	8
2.1	DFA, NFA	8

1 Erste Woche

1.1 Sprachen

Alphabet Σ : nichtleere endliche Menge (von Zeichen)

Wort ueber Σ : endliche Folge von Zeichen aus Σ

Leeres Wort: ϵ (epsilon)

Menge aller Woerter ueber Σ : Σ^*

Konkatenation von Woertern x, y ueber Σ :

$$x = x_1x_2\dots x_n \quad , x_i \in \Sigma$$

$$y = y_1y_2\dots y_n \quad , y_i \in \Sigma$$

$$x \cdot y = xy = x_1x_2\dots x_ny_1y_2\dots y_n$$

Java: + """ (ϵ)

Haskell: ++ """ (ϵ)

Monoid: Sei M eine Menge und

$\circ : M \times M \rightarrow^{total} M$ eine Verknuepfung

Das Paar (M, \circ) heisst ein Monoid, falls gilt:

1) $a \circ (b \circ c) = (a \circ b) \circ c \quad , \forall a, b, c \in M$

2) Es gibt ein $e \in M$ mit $a \circ e = a = e \circ a \quad , \forall a \in M$

□

Beispiel 1

$$M = \Sigma^*, \circ = \cdot$$

(Σ^*, \cdot) ist ein Monoid mit ϵ als neutralem Element

□

Beispiel 2

$$\{\{x = 5; y = 6; \}z = 7; \} \equiv \{x = 5; \{y = 6; z = 7; \}\}$$

Komposition von Anweisungen assoziativ

Neutrales Element: ; (Java) skip, NOP (no operation)

$$(x = 2 * x; x = x + 1;) \not\equiv (x = x + 1; x = 2 * x)$$

□

Sprache ueber Σ :

Menge von Woerter ueber Σ

Beispiele

$\{\}$ 0 Woerter

$\{0, 1, 01, 10\}$ Sprache uber $\Sigma = \{0, 1\}$
 Σ^*

$\{\epsilon\}$ 1 Wort

$\{\epsilon, 0, 00, 000, \dots\}$ uber $\Sigma = \{0\}$

Bem

Sprache kann ∞ viele Woerter enthalten
Jedes Wort ist aber endlich

Bem

$\epsilon \in \Sigma^*$

Σ^* immer ∞ gross

Operationen auf Sprachen

Seien L_1, L_2 Sprachen

$L_1 \cup L_2$ Vereinigungsmenge

$L_1 \cdot L_2 = \{xy \mid x \in L_1, y \in L_2\}$ (Kreuzprodukt)

Sei (M, \circ) ein Monoid. Dann def.

$$\begin{aligned} a^0 &= e & , a \in M \\ a^n &= a \circ a^{n-1} & , n > 0 \end{aligned}$$

□

$$L^0 = \{\epsilon\}$$

$$L^n = L \cdot L^{n-1} \quad , n > 0$$

Kleen' scher Stern

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

$$= \{x_1, x_2, \dots, x_k \mid k \geq 0, x_i \in L\}$$

Aufgabe

$$\Sigma = \{a, b, \dots, z\}, L_1 = \{good, bad\}, L_2 = \{cat, dog\}$$

$$L_1 \cup L_2 = \{bad, cat, dog, good\}$$

$$L_1 \cdot L_2 = \{goodcat, gooddog, badcat, baddog\}$$

$$L_1^0 = \{\epsilon\}$$

$$L_1^1 = \{good, bad\} = L_1 \cdot L_1^0 = L_1$$

$$L_1^2 = \{goodgood, goodbad, badgood, badbad\}$$

$$L_1^3 = \{goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, badgoodgood, badgoodbad, badbadgood, badbadbad\}$$

$$L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3 \cup \dots \\ = \{x_1, x_2, \dots, x_k \mid k \geq 0, x_i \in L_1\} = \{\epsilon, \dots\}$$

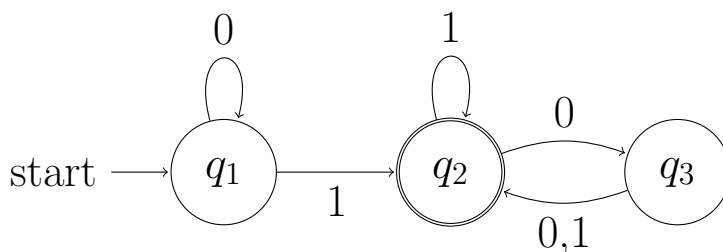
$$L_1 \cdot L_2 = \{goodcat, gooddog, badcat, baddog\} \neq L_2 \cdot L_1$$

$$|M| = \text{Anzahl Elemente von } M$$

1.2 Endliche Automaten DFA

deterministic finite automator

Statisch



Dynamisch

Verarbeitung Input: $\xrightarrow{1101}$

1. Start in q_1 Startzustand
2. Lese ①101 , $q_1 \rightarrow q_2$
3. Lese 1①01 , $q_2 \rightarrow q_2$
4. Lese 11②1 , $q_2 \rightarrow q_3$
5. Lese 110③ , $q_3 \rightarrow q_2$
6. Fertig + akzeptiere, da q_2 akzeptierender Zustand ist und die Eingabe fertig gelesen ist.

Liefert accept oder fertig

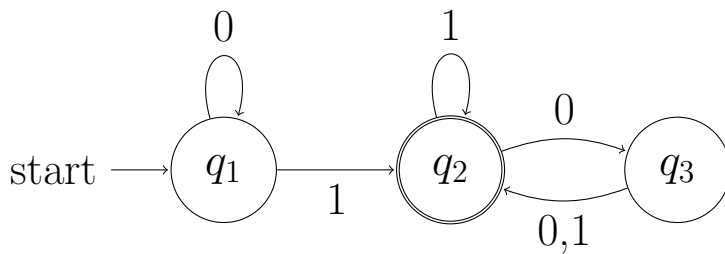
Terminiert immer!

Def DFA : Ein DFA ist ein 5-Tupel $(Q, \Sigma, \delta, q_0, F)$ mit:

1. Q ist eine endliche nichtleere Menge von Zuständen
 2. Σ ist das Eingabealphabet (z.B. 1101)
 3. $\delta : Q \times \Sigma \rightarrow^{total} Q$
 Transitionsfunktion
 4. q_0 Startzustand
 5. $F \subseteq Q$ Menge der akzeptierende Zustände
-

2 Zweite Woche

2.1 DFA, NFA



1. $Q = \{q_1, q_2, q_3\}$

2. $\Sigma = \{0, 1\}$

3. $\delta : Q \times \Sigma \rightarrow Q$

$$\delta(q_1, 0) = q_1, \delta(q_1, 1) = q_2, \dots$$

4. q_1 Start

5. $F = \{q_2\}$

Def Verarbeitung (dynamisch)

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA

Sei $w = x_1x_2x_3...x_m$ ein Wort ueber Σ mit $x_i \in \Sigma, n \geq 0$ [$n = 0 \rightarrow w = \epsilon$]

M akzeptiert w , wenn eine Folge von Zustaenden existiert $r_0, r_1, r_2, \dots, r_n$, mit:

1. $r_0 = q_0$
2. $r_i = \delta(r_{i-1}, x_i), i \in \{1...m\}$
3. $r_n \in F$

Sonst wird w verworfen

accept / reject

□

M erkennt Sprache L falls

$L = \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}$

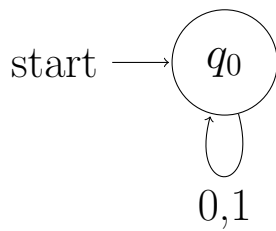
Eine Sprache heisst regulaer, wenn ein

DFA existiert, der die Sprache erkennt

Automat \rightarrow akzeptiert / verwirft Wort

\searrow
erkennt Sprache
 \uparrow
recognise

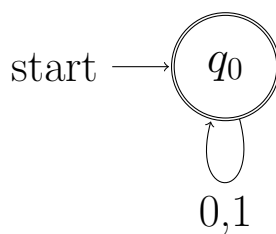
$M_2 : \quad \Sigma = \{0, 1\}$



akzeptiert kein Wort

erkennt \emptyset

$M_3 : \quad \Sigma = \{0, 1\}$



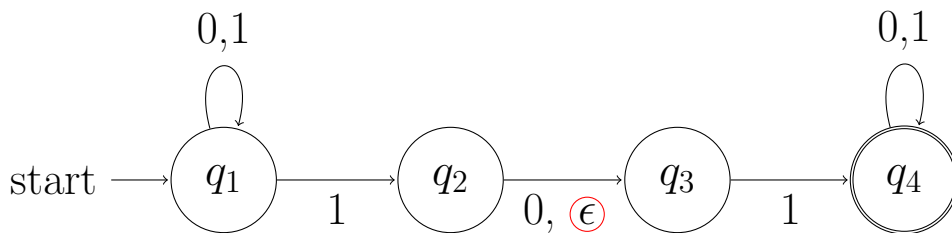
akzeptiert jedes Wort

erkennt Σ^*

Zwei DFA heissen aequivalent, wenn sie dieselbe Sprachen erkennen

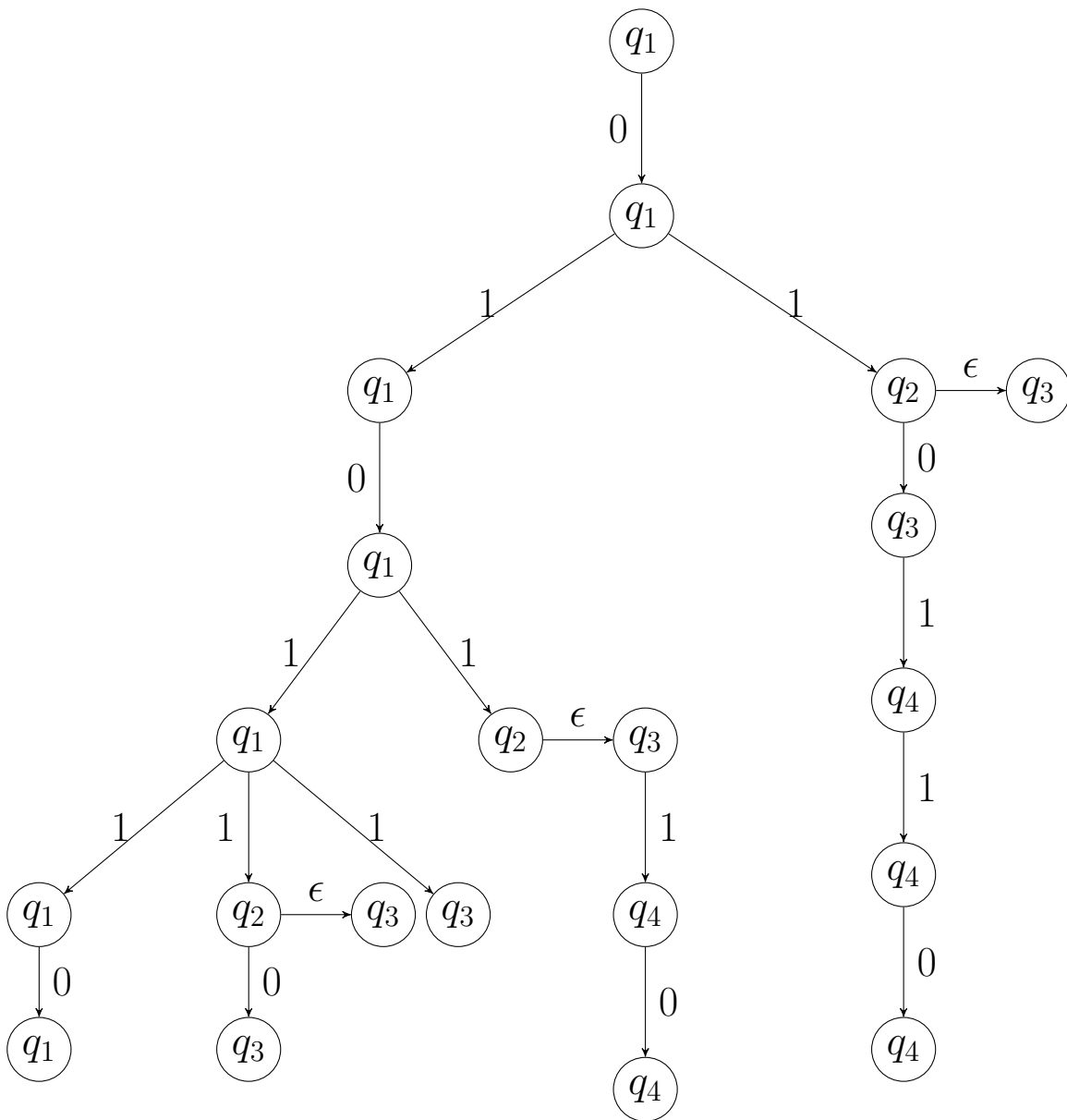
NFA: nichtdeterministischer FA

$N_1 : \quad \Sigma = \{0, 1\}$



Eingabe: 010110

Verarbeitung:



Def: $P = P(Q) = 2^Q = \underline{\text{Potenzmenge}}$
ist Menge aller Teilmengen von Q

$$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$$

Def DFA : Ein NFA ist ein 5-Tupel $(Q, \Sigma, \delta, q_0, F)$ mit:

1. Q ist eine endliche nichtleere Menge von Zuständen
 2. Σ ist das Eingabealphabet (z.B. 1101)
 3. $\delta : Q \times \Sigma_\epsilon \rightarrow^{total} P(Q)$
Transitionsfunktion
 4. $q_0 \in Q$ Startzustand
 5. $F \subseteq Q$ Menge der akzeptierenden Zustände
-