

# **Einfuehrung in die Theoretische Informatik**

Kyriakos Schwarz,Roland Hediger

HS 2014

# Contents

<b>1</b>	<b>Erste Woche</b>	<b>1</b>
1.1	Sprachen . . . . .	1
1.2	Endliche Automaten DFA . . . . .	5
<b>2</b>	<b>Zweite Woche</b>	<b>8</b>
2.1	DFA, NFA . . . . .	8
<b>3</b>	<b>Dritte Woche</b>	<b>14</b>
3.1	NFA . . . . .	14
<b>4</b>	<b>Vierte Woche</b>	<b>18</b>
4.1	Abgeschlossenheit . . . . .	18
4.2	RE . . . . .	20
<b>5</b>	<b>Funfte Woche</b>	<b>23</b>
5.1	Pumping Lemma . . . . .	23
<b>6</b>	<b>Sechste Woche</b>	<b>27</b>
6.1	Grammatiken . . . . .	27

# 1 Erste Woche

## 1.1 Sprachen

Alphabet  $\Sigma$ : nichtleere endliche Menge (von Zeichen)

Wort ueber  $\Sigma$ : endliche Folge von Zeichen aus  $\Sigma$

Leeres Wort:  $\epsilon$  (epsilon)

Menge aller Woerter ueber  $\Sigma$ :  $\Sigma^*$

Konkatenation von Woertern  $x, y$  ueber  $\Sigma$ :

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

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

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

Java: +                """ ( $\epsilon$ )

Haskell: ++            """ ( $\epsilon$ )

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

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

$(\Sigma^*, \cdot)$  ist ein Monoid mit  $\epsilon$  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  $\Sigma$ :

Menge von Woerter ueber  $\Sigma$

Beispiele

$\{\}$  0 Woerter

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

$\{\epsilon\}$  1 Wort

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

Bem

Sprache kann  $\infty$  viele Woerter enthalten  
Jedes Wort ist aber endlich

Bem

$\epsilon \in \Sigma^*$

$\Sigma^*$  immer  $\infty$  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

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

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

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

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

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

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

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

$$L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3 \cup \dots$$

$$= \{x_1, x_2, \dots, x_k \mid k \geq 0, x_i \in L_1\} = \{\epsilon, \dots\}$$

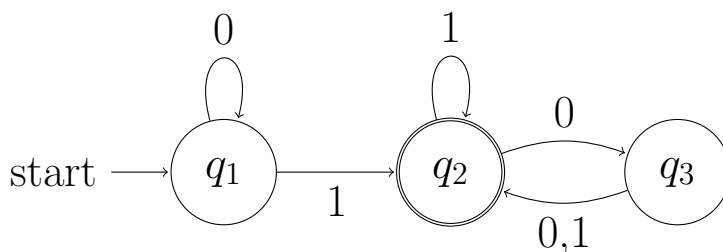
$$L_1 \cdot L_2 = \{goodcat, gooddog, badcat, baddog\} \neq L_2 \cdot L_1$$

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

## 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:

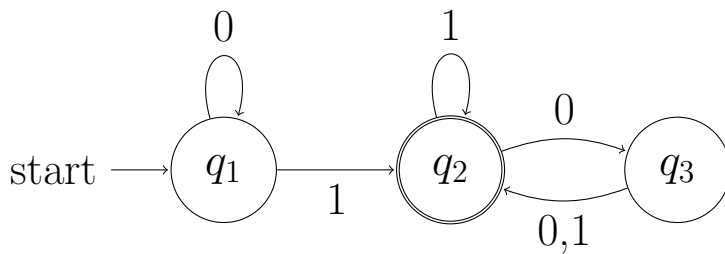
1.  $Q$  ist eine endliche nichtleere Menge von Zuständen
  2.  $\Sigma$  ist das Eingabealphabet (z.B. 1101)
  3.  $\delta : Q \times \Sigma \rightarrow^{total} Q$   
    Transitionsfunktion
  4.  $q_0$  Startzustand
  5.  $F \subseteq Q$  Menge der akzeptierende Zustände
-





## 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  $\Sigma$  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

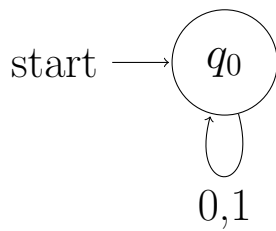
DFA existiert, der die Sprache erkennt

Automat  $\rightarrow$  akzeptiert / verwirft Wort

$\searrow$   
erkennt Sprache  
 $\uparrow$   
recognise

---

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

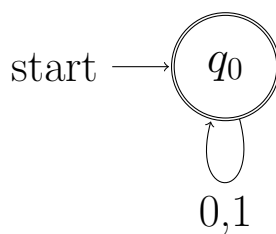


akzeptiert kein Wort

erkennt  $\emptyset$

---

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



akzeptiert jedes Wort

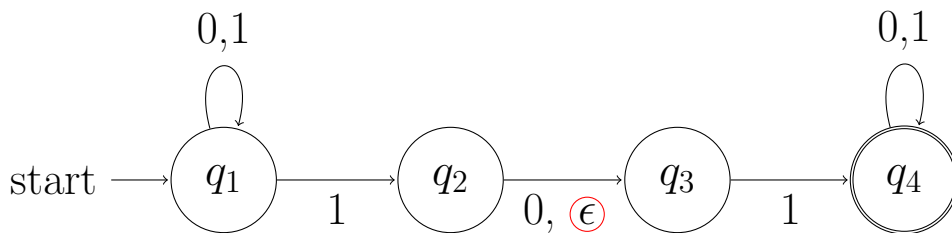
erkennt  $\Sigma^*$

---

Zwei DFA heissen aequivalent, wenn sie dieselbe Sprachen erkennen

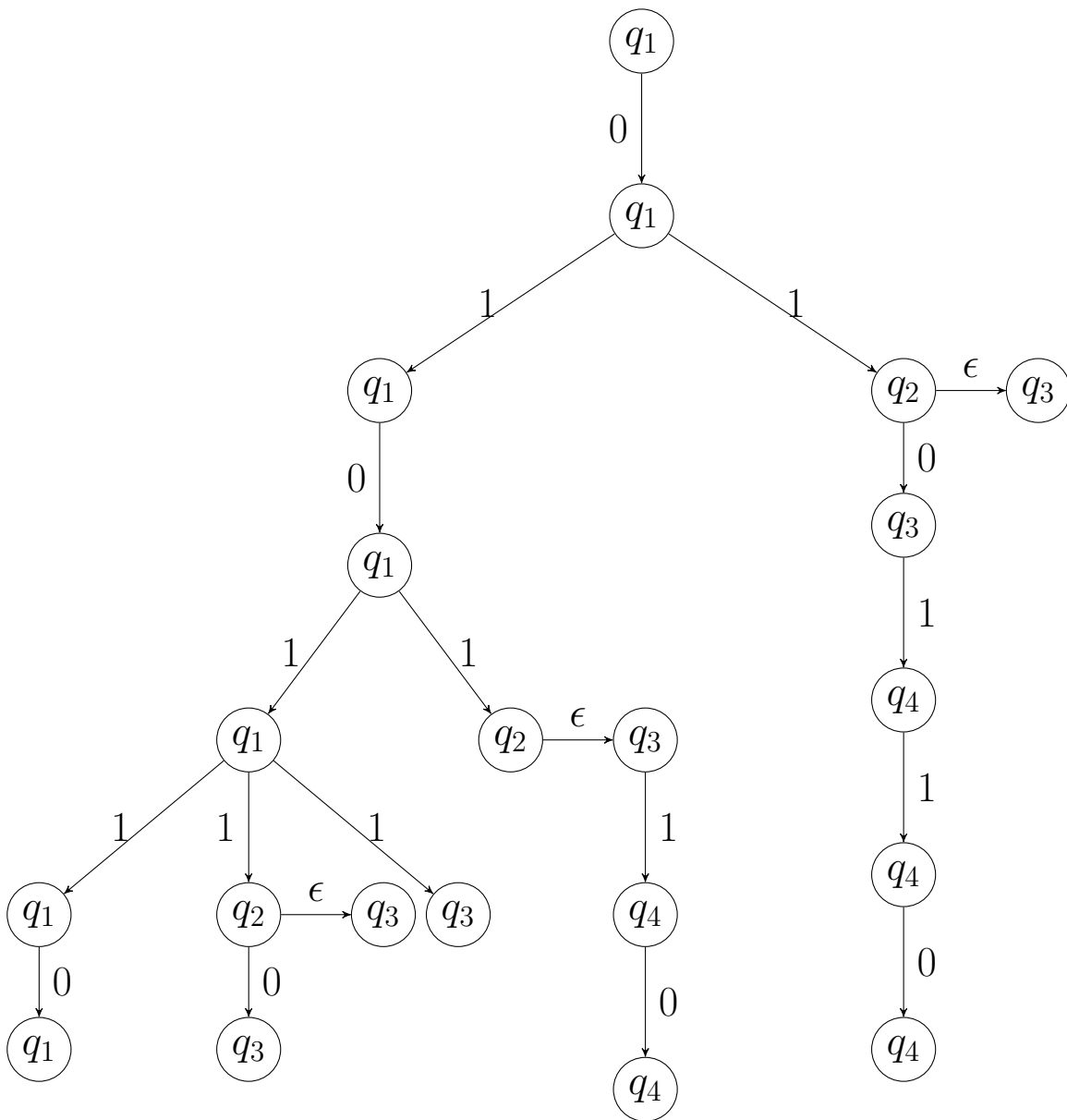
NFA: nichtdeterministischer FA

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



Eingabe: 010110

Verarbeitung:



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

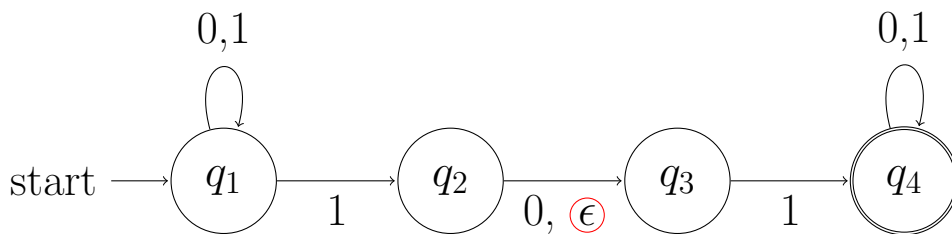
$$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$$

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.  $\Sigma$  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
-

## 3 Dritte Woche

### 3.1 NFA



#### Berechnung NFA

Sei  $N = (Q, \Sigma, \delta, q_0, F)$  ein NFA

Sei  $w = y_1 y_2 \dots y_m, y_i \in \Sigma \cup \epsilon$

Es existiere eine Folge von Zuständen

$r_0 r_1 r_2 \dots r_m, r_i \in Q$  mit

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

Dann akzeptiert  $N$  das Wort  $w$ , sonst verwirft es



□

Beispiel  $N_1$  auf 010110 (unseres Beispiel)

1<sup>er</sup> Weg

$q_1$		$q_1$		$q_2$		$q_3$		$q_4$		$q_4$		$q_4$
	0		1		0		1		1		0	

2<sup>er</sup> Weg

$q_1$		$q_1$		$q_1$		$q_1$		$q_2$		$q_3$		$q_4$		$q_4$
	0		1		0		1		$\epsilon$		1		1	

Theorem: Jeder NFA hat einen äquivalenten DFA □

Beweis: Sei  $N = (Q, \Sigma, \delta, q_0, F)$  ein NFA der  $L$  erkennt

Wir konstruieren ein DFA  $D = (Q', \Sigma', \delta', q'_0, F')$ , der ebenfalls  $L$  erkennt

1.  $Q' = 2^Q = P(Q)$

2.  $\delta'(R, \alpha) = \bigcup_{r \in R} \delta(r, \alpha)$   
 $\uparrow$   
 $\in Q'$

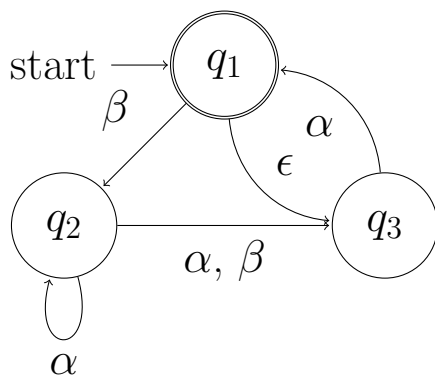
Sei  $E(R) = \{q \in Q \mid q \text{ kann von } R \text{ aus durch } \epsilon\text{-Trans erreicht werden}\}$

$$3. q'_0 = E(\{q_0\})$$

$$4. F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$$

□

### Uebung



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

$$\Sigma = \{\alpha, \beta\}$$

$$\delta = \{\delta(q_1, \beta) = q_2,$$

$$\delta(q_1, \epsilon) = q_3$$

$$\delta(q_2, \alpha) = q_2$$

$$\delta(q_2, \beta) = q_3$$

$$\delta(q_3, \alpha) = q_1\}$$

$$q_0 = q_1$$

$$F = \{q_1\}$$

→ DFA

$$1. Q' = P(Q) = \{\{\}, \{q_1\}, \{q_2\}, \{q_3\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$$

2.  $\delta'$ :

	$\alpha$	$\beta$
$\{\}$	$\{\}$	$\{\}$
$\{q_1\}$	$\{\}$	$\{q_2\}$
$\{q_2\}$	$\{q_2, q_3\}$	$\{q_3\}$
$\{q_3\}$	$\{q_1, q_3\}$	$\{\}$
$\{q_1, q_2\}$	$\{q_2, q_3\}$	$\{q_2, q_3\}$
$\{q_1, q_3\}$	$\{q_1, q_3\}$	$\{q_2\}$
$\{q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_3\}$
$\{q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$

$$3. q'_0 = E(\{q_0\}) = E(\{q_1\}) = \{q_1, q_3\}$$

$$4. F' = \{\{q_1\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_1, q_2, q_3\}\}$$

## 4 Vierte Woche

### 4.1 Abgeschlossenheit

Abgeschlossenheit

$$a, b \in \mathbb{N}, a + b \in \mathbb{N}$$

$$a - b \notin \mathbb{N}$$

↑

im Allgemeinen

Def

Eine Menge  $M$  heisst abgeschlossen unter einer Operation  $\circ$ , wenn  $a \circ b \in M$  fuer alle  $a, b \in M$

□

Satz

Die Menge der raegularen Sprachen ist abgeschlossen unter Vereinigung, Konkatenation und Kleenescher Stern

□

$\cup$  Vereinigung

1. Schon gezeigt durch DFAs
2. Nun mit NFAs:

<skizze>

$N_1$  erkennt  $L_1$

$N_2$  erkennt  $L_2$

$N$  erkennt  $L_1 \cup L_2$

Konkatenation

$L_1, L_2$  regulaer, dann  $L_1 \cdot L_2$  regulaer

<skizze>

Kleen'scher Stern

$L$  regulaer, dann  $L^*$  regulaer

<skizze>

$N$  erkennt  $L^*$

---

## 4.2 RE

Raegulere Ausdruecke (RE)

REs: Spezifikation

DFA / NFAs: Implementation

Arithmetischer Ausdruck:  $(5 + 3) * 4$  : String  
32 :  $\mathbb{N}$

Bedeutung von String ist  $\mathbb{N}$

RE:  $(0 \cup 1) \cdot 0^*$  : String  
 $\{0\} \{1\} \{0\}$   
 $\{0, 1\} \{\epsilon, 0, 00, 000, \dots\}$   
 $\{0, 00, 000, \dots, 1, 10, 100, \dots\}$

Bedeutung von String ist Sprache

---

Syntax und Semantik von REs

	RE Syntax	L(RE) Semantik
1.	$a$ fuer $a \in \Sigma$	$\{a\}$
2.	$\epsilon$	$\{\epsilon\}$
3.	$\emptyset$	$\emptyset$
4.	$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
5.	$(R_1 \circ R_2)$	$L(R_1) \cdot L(R_2)$
6.	$(R^*)$	$(L(R))^*$

6. (Hoch)  $\xrightarrow{\text{Praezidenzen}}$  (Niedrig) 1.

z.B.:  $a \cdot b \cup c$

bedeutet

$(a \cdot b) \cup c$

und nicht

$a \cdot (b \cup c)$

Zucker (Es kann nichts neues)

$$R^+ = R \circ R^*$$

also

$$R^* = R^+ \cup \epsilon$$

$$\Sigma = c_1 \cup c_2 \cup \dots \cup c_n \text{ mit } c_i \in \Sigma$$

## Beispiele

a) " $(0 \cup 1)^*01$ " bezeichnet alle Woerter, die mit 01 enden

---



## 5 Funfte Woche

### 5.1 Pumping Lemma

$\{0^n 1^n | n \geq 0\}$  nicht Regulär

Pumping Lemma:

Sei  $L$  eine Reguläre Sprache über  $\Sigma$

Dann existiert eine Zahl  $p \in \mathbb{N}$  (Pumping Länge) mit:

Jedes Wort  $s \in L$  mit  $|s| \geq p$  lässt sich schreiben als:

$s = xyz$  wobei  $x, y, z \in \Sigma^*$  mit folgenden eigenschaften:

1. (Aufpumpen)  $xy^iz \in L$  für alle  $i \geq 0$
2.  $|y| \geq 1$
3.  $|xy| \leq p$

Für jedes Wort existiert eine Zerlegung.

Logische Struktur :  $\{\exists p | \forall s \in L | \exists x, y, z | (1)2)3)\}$

Beweis:

Sei  $M$  ein DFA das  $L$  erkennt,  $M = \{Q, \Sigma, \delta, q_0, F\}$

Sei  $p$  die Anzahl Zustände :  $p = |Q|$

Sei  $s = s_1 \dots s_n \in L$  mit  $s_i \in \Sigma$  und  $n \geq p$

Sei  $r_1, r_2 \dots r_{n+1}$  die Folge der Zustände die  $M$  durchläuft um  $s$  zu akzeptieren.

$$s = s_1 \downarrow_{r_1} \dots \downarrow_{r_n} s_n \downarrow_{r_{n+1} \in F}$$

Länge  $n + 1 \geq p + 1$ , d.h.  $n + 1 > p$

Da  $M$   $p$  Zustände hat muss **(mindestens) ein Zustand (mindestens) zweimal unter den ersten  $p+1$  Zuständen vorkommen / durchgelaufen werden** (Eigenschaften des DFA - jedes Zeichen des Alphabet muss durch jeder Zustand verarbeitet werden)

### **Pidgeonhole Prinzip:**

Bezeichne  $r_j$  das erste Auftreten eines solchen Zustandes (Zustände paarweise verschieden bis zum gewissen Punkt).  $r_l$  das zweite Auftreten des Zustandes  $l > j$

(Diagram hier)

Aus dem Pidgeonholeprinzip folgt dass  $l \leq p + 1$  :

1.  $M$  akzeptiert  $xy^iz, i \geq 0$
2. (Zwischen  $r_j, r_l$  mind. 1 Zeichen)  $l \neq j$ , also  $|y| \geq 1$
3.  $|xy| = l - 1 \leq (p + 1) - 1 = p$

□

---

Beispiel:

$$\{0^n 1^n | n \geq 0\} = L_1$$

Sei  $L_1$  Regulär dann existiert ein  $p$  (Pumping Lemma).

Betrachte  $s = 0^p 1^p \in L$  :

Dann lässt sich  $s$  schreiben als  $s = xyz$  mit der Eigenschaft:  $s = xyz, x, y^i z \in L, |y| \geq 1$

1.  $y$  besteht nur aus Nullen :  $xyyz \notin L$
2.  $y$  besteht nur aus Einsen  $xyyz \notin L$
3.  $y$  besteht aus Nullen und Einsen  $\rightarrow xyyz$  : (Reihenfolge ist

falsch)

Widerspruch in jedem Fall  $\rightarrow L_1$  nicht regulär

---

## 6 Sechste Woche

### 6.1 Grammatiken

while  $m \neq n$  do  
    if  $m > n$  then  
         $m := m - n$   
    else  
         $n := n - m$   
    endif  
endwhile

$\Sigma = \{\underline{while}, \underline{do}, \underline{ident}, \dots\}$

$\text{cmd} ::= \text{AssiCmd}$   
 $\text{cmd} ::= \text{IfCmd}$   
 $\text{cmd} ::= \text{WhileCmd}$

$\text{WhileCmd} ::= \underline{\text{while}} \text{ expr } \underline{\text{do}} \text{ cmd } \underline{\text{endwhile}}$   
 $\text{IfCmd} \quad ::= \underline{\text{if}} \text{ expr } \underline{\text{then}} \text{ cmd } \underline{\text{else}} \text{ cmd } \underline{\text{endif}}$   
 $\text{IfCmd} \quad ::= \underline{\text{if}} \text{ expr } \underline{\text{then}} \text{ cmd } \underline{\text{endif}}$   
 $\text{AssiCmd} \quad ::= \underline{\text{ident}} := \text{expr}$   
 $\text{expr} \quad \quad ::= \dots$

$\text{IfCmd} ::= \underline{\text{if}} \text{ expr } \underline{\text{then}} \text{ cmd } \text{optElse} \underline{\text{endif}}$   
 $\text{optElse} ::= \epsilon$   
 $\text{optElse} ::= \underline{\text{else}} \text{ cmd}$

---

$\text{IfCmd} ::= \underline{\text{if}} \text{ expr } \underline{\text{then}} \text{ cmd } \underline{\text{endif}}$
--

Produktion

nicht-terminal Symbole

terminal Symbole

---

$\mu$ Deutsch

$\text{Satz} ::= \underline{\text{Subjekt}} \underline{\text{Praedikat}} \underline{\text{Objekt}} \underline{.}$

$\underline{\text{Subjekt}} ::= \underline{\text{Vogel}}$

$\underline{\text{Subjekt}} ::= \underline{\text{Katze}}$

$\underline{\text{Objekt}} ::= \underline{\text{Subjekt}}$

$\underline{\text{Praedikat}} ::= \underline{\text{frisst}}$

Satz

$\Rightarrow \text{Subjekt Praedikat Objekt } \underline{.}$

$\Rightarrow \underline{\text{Vogel}} \text{ Praedikat Objekt } \underline{.}$

$\Rightarrow \underline{\text{Vogel}} \text{ Praedikat Subjekt } \underline{.}$

$\Rightarrow \underline{\text{Vogel}} \text{ Praedikat } \underline{\text{Katze}} \underline{.}$

$\Rightarrow$  Vogel frisst Katze .

---

Eine Kontextfreie Grammatik ist ein 4-Tuple  $(V, \Sigma, R, S)$  mit

1.  $V$  : Alphabet der NTS (nicht-terminal symbole)
2.  $\Sigma$  : Alphabet der TS (terminal symbole)
3.  $R$  : endliche Menge von Produktionen  
Produktion  $\in V \times (V \cup \Sigma)^*$
4.  $S \in V$  : Startsymbol

---

Herleitung, Sprache

Seien  $u, v, w \in (V \cup \Sigma)^*$ ,  $A \xRightarrow{::=} w \in R$

Dann  $uAv \Rightarrow uww$

$\downarrow$   
"leitet her"

$u$  leitet  $v$  her,  $u \Rightarrow^* v$ , falls  $u = v$

oder es gibt eine Folge

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \dots \Rightarrow u_k \Rightarrow v$$

Sei  $G = (V, \Sigma, R, S)$

Dann ist

$$||| L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$


---