Einfuehrung in die Theoretische Informatik

Kyriakos Schwarz,Roland Hediger

HS 2014

Contents

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:

$x = x_1 x_2 ... x_n \qquad ,x_i \in \Sigma$
$y = y_1 y_2 ... y_n \qquad ,y_i \in \Sigma$

$x \cdot y = xy = x_1 x_2 ... x_n y_1 y_2 ... y_n$

Java: $+$ "" (ϵ)
Haskell: $++$ "" (ϵ)

Monoid: Sei M eine Menge und

$\circ : M \times M \xrightarrow{total} M$ eine Verknuepfung

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

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

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

□

Beispiel 1

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

$(\Sigma^*, \cdot)$ 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
$\{\epsilon, 0, 00, 000, ...\}$ 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.

$a^0 = e \qquad ,a \in M$
$a^n = a \circ a^{n-1} \qquad ,n > 0$

□

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

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

Kleen' scher Stern

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

$= \{x_1, x_2, ..., x_k \mid k \geqslant 0, x_i \in L\}$

Aufgabe

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

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

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

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

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

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

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

$L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3 \cup ...$
$= \{x_1, x_2, ..., x_k \,|k \geqslant 0, x_i \in, L_1\} = \{\epsilon, ...\}$

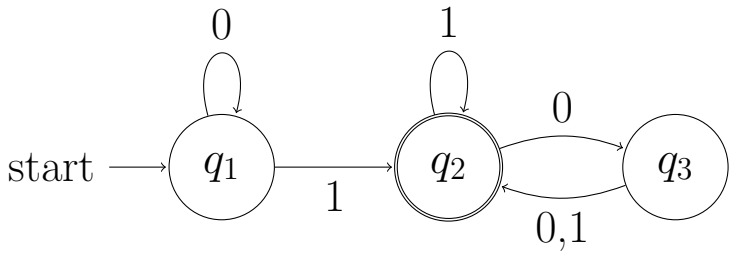
$L_1 \cdot L_2 = \{goodcat, gooddog, badcat, baddog\} \neq L_2 \cdot L_1$

$|M|$ = 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 , $q_1 \to q_2$
3. Lese 1①01 , $q_2 \to q_2$
4. Lese 11⓪1 , $q_2 \to q_3$
5. Lese 110① , $q_3 \to q_2$
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 Zustaenden

2. Σ ist das Eingabealphabet (z.B. 1101)

3. $\delta : Q \times \Sigma \xrightarrow{total} Q$
   Transitionsfunktion

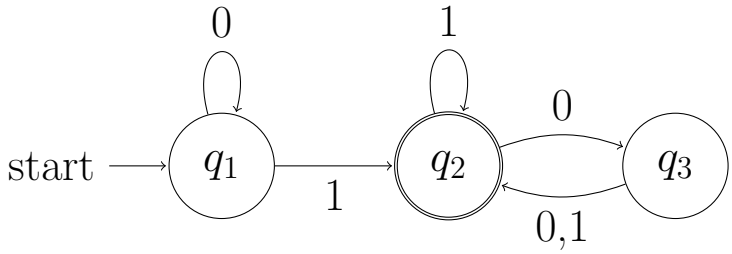
4. q_0 Startzustand

5. $F \subseteq Q$ Menge der akzeptierende Zustaende

---

2 Zweite Woche

2.1 DFA, NFA



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

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

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

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

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 $x_i \in \Sigma$, $n \geqslant 0$ [$n = 0 \rightarrow w = \epsilon$]

M akzeptiert w, wenn eine Folge von Zustaenden existiert $r_0, r_1, r_2, ..., r_n$, 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

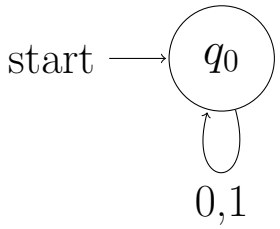
$\searrow$

erkennt Sprache

$\uparrow$

recognise

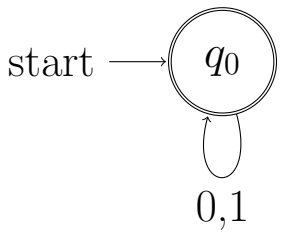
M_2 : $\quad \Sigma = \{0, 1\}$



akzeptiert kein Wort

erkennt $\emptyset$

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



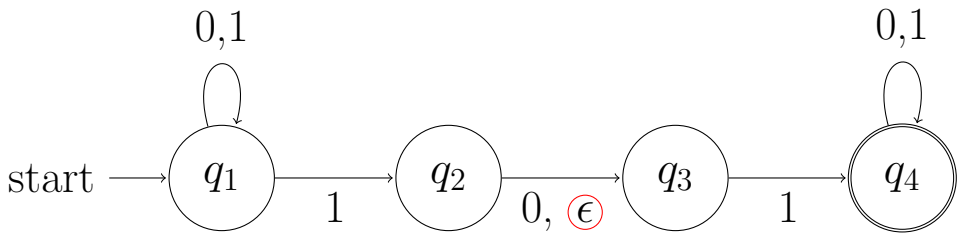
akzeptiert jedes Wort

erkennt Σ^*

Zwei DFA heissen aequivalent, wenn sie
dieselbe Sprachen erkennen

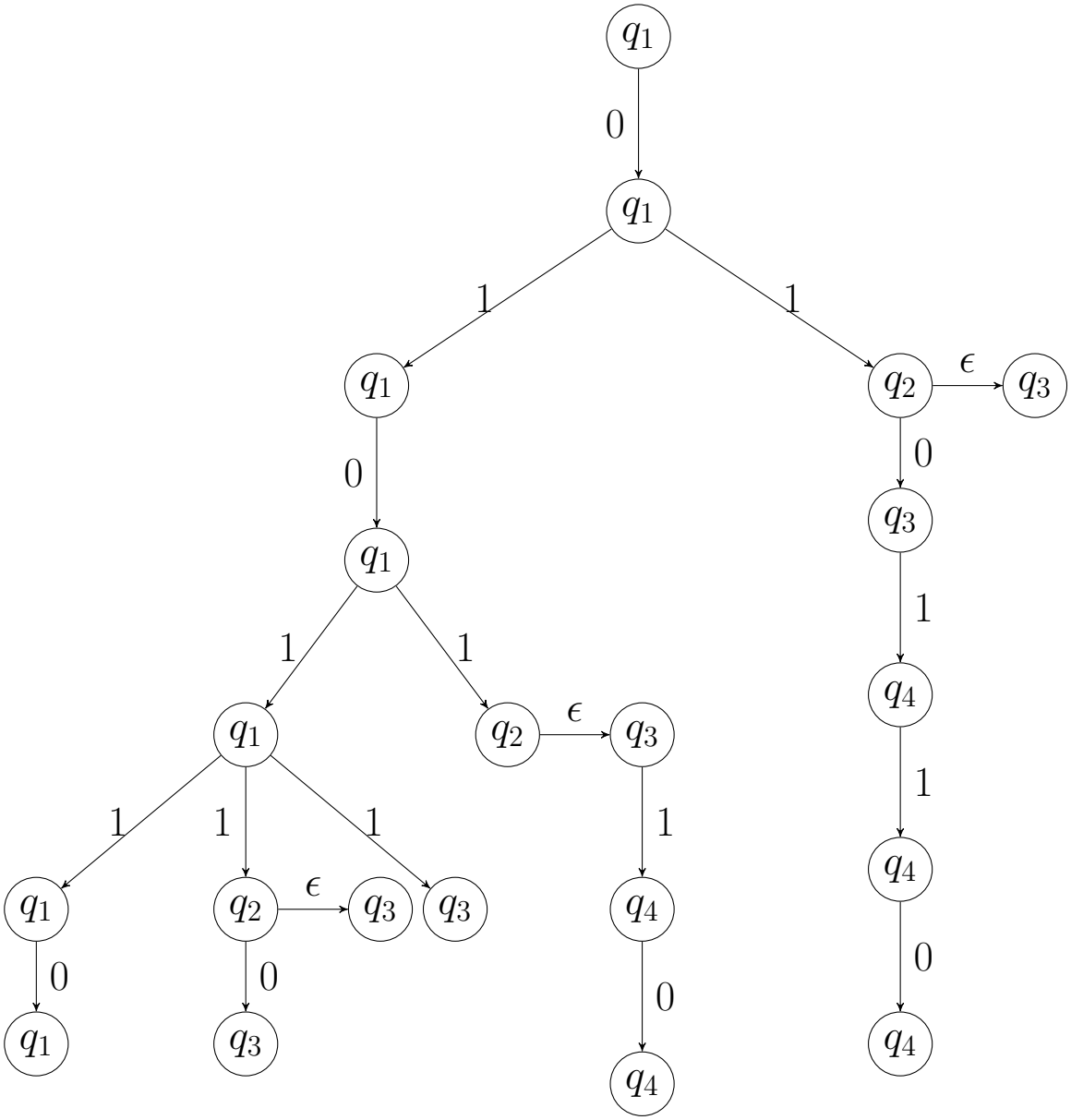
NFA: nichtdeterministischer FA

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



Eingabe: 010110

Verarbeitung:



Def: $P = P(Q) = 2^Q =$ 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 Zustaenden

2. Σ ist das Eingabealphabet (z.B. 1101)

3. $\delta : Q \times \Sigma_\epsilon \xrightarrow{total} P(Q)$
   Transitionsfunktion

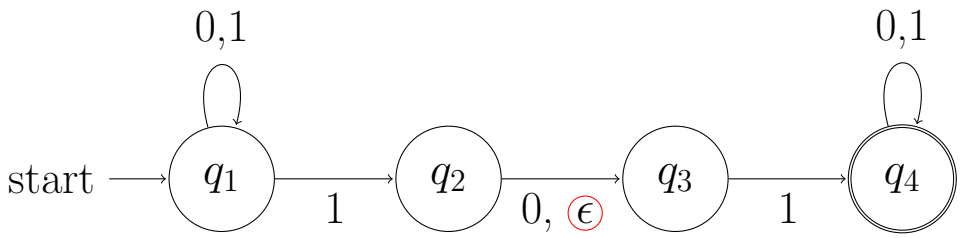
4. $q_0 \in Q$ Startzustand

5. $F \subseteq Q$ Menge der akzeptierende Zustaende

---

3 Dritte Woche

3.1 NFA



Berechnung NFA

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

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

Es existiere eine Folge von Zustaenden
$r_0 r_1 r_2 ... r_m$, $r_i \in Q$ mit

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

Dann akzeptiert N das Wort w, sonst verwirft es

□

Beispiel N_1 auf 010110 (unseres Beispiel)

1^{er} Weg

$$\begin{array}{ccccccccccccc} q_1 & & q_1 & & q_2 & & q_3 & & q_4 & & q_4 & & q_4 \\ & 0 & & 1 & & 0 & & 1 & & 1 & & 0 & \end{array}$$

2^{er} Weg

$$\begin{array}{ccccccccccccccc} q_1 & & q_1 & & q_1 & & q_1 & & q_2 & & q_3 & & q_4 & & q_4 \\ & 0 & & 1 & & 0 & & 1 & & \epsilon & & 1 & & 1 & \end{array}$$

Theorem: Jeder NFA hat einen aequivalenten 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. $\delta'(R, \alpha) = \bigcup_{r \in R} \delta(r, \alpha)$
   $\uparrow$
   $\in Q'$

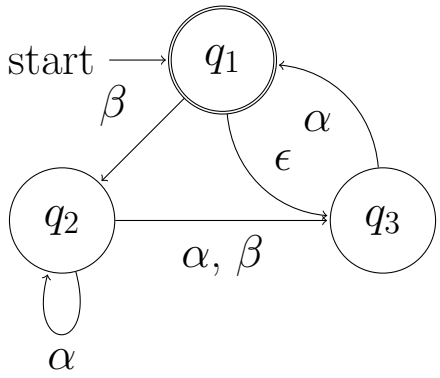
Sei $E(R) = \{q \in Q \,|\, q \text{ kann von } R \text{ aus durch } \epsilon\text{-Trans erreicht werden}\}$

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

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

□

Uebung



$Q = \{q_1, q_2, q_3\}$
$\Sigma = \{\alpha, \beta\}$
$\delta = \{\delta(q_1, \beta) = q_2,$
$\quad \delta(q_1, \epsilon) = q_3$
$\quad \delta(q_2, \alpha) = q_2$
$\quad \delta(q_2, \alpha) = q_3$
$\quad \delta(q_2, \beta) = q_3$
$\quad \delta(q_3, \alpha) = q_1\}$
$q_0 = q_1$
$F = \{q_1\}$

$\rightarrow$ 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. δ':

	α	β
$\{\}$	$\{\}$	$\{\}$
$\{q_1\}$	$\{\}$	$\{q_2\}$
$\{q_2\}$	$\{q_2, q_3\}$	$\{q_3\}$
$\{q_3\}$	$\{q_1, q_3\}$	$\{\}$
$\{q_1, q_2\}$	$\{q_2, q_3\}$	$\{q_2, q_3\}$
$\{q_1, q_3\}$	$\{q_1, q_3\}$	$\{q_2\}$
$\{q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_3\}$
$\{q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$

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

$a, b \in \mathbb{N}$, $a + b \in \mathbb{N}$
$a - b \notin \mathbb{N}$
$\uparrow$
im Allgemeinem

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 raeguleren Sprachen ist abgeschlossen unter Vereinigung, Konkatenation und Kleenescher Stern

□

$\bigcup$ Vereinigung

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

<skitze>

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

<skitze>

Kleen'scher Stern

L regulaer, dann L^* regulaer

<skitze>

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
$\{0\}\ \{1\}\ \{0\}$
$\{0,1\}\ \{\epsilon, 0, 00, 000, ...\}$
$\{0, 00, 000, ..., 1, 10, 100, ...\}$

Bedeutung von String ist Sprache

Syntax und Semantik von REs

	RE Syntax	L(RE) Semantik
1.	a fuer $a \in \Sigma$	$\{a\}$
2.	ϵ	$\{\epsilon\}$
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) $\xrightarrow{Praezidenzen}$ (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

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

$\Sigma = c_1 \cup c_2 \cup ... \cup c_n$ 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 Σ
Dann existiert eine Zahl $p \in N$ (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) $xy^i z \in L$ 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:
Beziechne 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 $xy^i z, i \geq 0$

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 $\rightarrow$ $xyyz$: (Riehenfolge ist

falsch)

Wiederspruch 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

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

cmd ::= AssiCmd
cmd ::= IfCmd
cmd ::= WhileCmd

WhileCmd ::= while expr do cmd endwhile
IfCmd ::= if expr then cmd else cmd endif
IfCmd ::= if expr then cmd endif
AssiCmd ::= ident := expr
expr ::= ...

IfCmd ::= if expr then cmd optElse endif
optElse ::= ϵ
optElse ::= else cmd

IfCmd ::= if expr then cmd endif	Produktion

nicht-terminal Symbole
terminal Symbole

μDeutsch

Satz ::= Subjekt Praedikat Objekt .
Subjekt ::= Vogel
Subjekt ::= Katze
Objekt ::= Subjekt
Praedikat ::= frisst

Satz
$\Rightarrow$ Subjekt Praedikat Objekt .

$\Rightarrow$ Vogel Praedikat Objekt .

$\Rightarrow$ Vogel Praedikat Subjekt .

$\Rightarrow$ Vogel Praedikat Katze .

$\Rightarrow$ Vogel frisst Katze .

Eine Kontextfreie Grammatik ist ein 4-Tuple (V, Σ, R, S) mit

1. V : Alphabet der NTS (nicht-terminal symbole)

2. Σ : 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 $uAv \Rightarrow uwv$
$\downarrow$
"leitet her"

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

oder es gibt eine Folge

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

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

Dann ist

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

Beispiel

$L = \{0^n \# 1^n \mid n \geq 0\}$ nicht regulaer

$\Sigma = \{0, 1, \#\}$
$A \to 0A1$
$A \to \#$

A
$\Rightarrow$
$0A1$
$\Rightarrow$
$00A11$
$\Rightarrow$
...

$= \{\#, 0\#1, 00\#11, 000\#111, ...\}$

---

7 Siebte Woche

7.1 Turing Maschinen

Satz

Jede regulaere Sprache ist Kontextfrei

□

Beweis-Idee

DFA:

<skitze>

$R_1 \to 0R_1 \mid 1R_2$
$R_2 \to 0R_3 \mid 1R_2 \mid \epsilon$
$R_3 \to 0R_2 \mid 1R_2$

$\overline{L} = \Sigma^* - L$

<skitze>

Turing Maschine

Alan Turing (1912 - 1954)
$\hookrightarrow$ ACM Turing Award

<skitze>

Turing-Maschine 7-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$

1. Q : endliche Menge von Zustaenden
   $|Q| \geq 2$ (mind. acc und rej)

2. Σ : Eingabealphabet, Blank $\sqcup \notin \Sigma$ (gehoert nicht zur Eingabe)

3. Γ : Bandalphabet, $\sqcup \in \Gamma$, $\Sigma \subset \Gamma$

4. $\delta : (Q - \{q_{acc}, q_{rej}\}) \times \Gamma \xrightarrow{total} Q \times \Gamma \times \{L, R\}$

5. $q_0 \in Q$: Startzustand

6. $q_{acc} \in Q$: akzeptierende Zustand

$$q_{acc} \neq q_{rej}$$

7. $q_{rej} \in Q$: verwerfender Zustand

---

8 Achte Woche

8.1 Turing Erkennbarkeit / Entscheidbarkeit

TM (Turing Maschine) M akzeptiert (verwirft) Eingabe $w \in \Sigma^*$, falls eine Folge von Konfigurationen existiert $C_1, C_2, ..., C_k$ mit

1. C_1 ist die Startkonfiguration
2. C_i liefert C_{i+1} gemaess Bewegungen gemaess δ
3. C_k ist akzeptierend (verwerfend)

□

Beispiel

TM, die $A = \{O^{2^n} \mid n \geq 0\}$, also
$A = \{0, 00, 0000, 00000000, ...\}$

$\Sigma = \{0\}$
$\Gamma = \{0, \sqcup, x\}$
$Q = \{q_1, ..., q_5, q_{acc}, q_{rej}\}$
$Q' = Q - \{q_{acc}, q_{rej}\}$

$\delta : Q' \times \Gamma \xrightarrow{total} Q \times \Gamma \times \{L, R\}$

Sei M eine TM ueber Σ

$L(M) = \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}$ ist die von M erkannte Sprache

M partitioniert Σ^*

(1) Solange M laeuft, wissen wir nicht ob zu anhalten wird, oder weiterlaufen wird

Sei $w \in acc$. Dann haelt M an. Dann weiss man, dass $w \in acc$.
Sei $w \in rej$. Analog. Sei $w \in \infty$. Dann laeuft M ∞ lange.
Mit (1) weiss man nie, dass $w \in \infty$

Super waere eine Maschine H, die immer haelt und sagt, ob eine Maschine M auf eine Eingabe w haelt oder nicht

Sei M eine TM ueber Σ
$L(M) = \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}$ ist die von M erkannte Sprache

Sprache L heisst Turing-erkennbar (rekursiv-aufzaehlbar, semi-entscheidbar), falls eine TM gibt, die L erkennt

Sprache L heisst Turing-entscheidbar (rekursiv, entscheidbar), falls eine TM gibt, die L erkennt und auf jeder Eingabe haelt

Beispiel: M_2 Entscheider, M_2^∞ nicht (aber Erkenner)

$L(M_2) = \{0^{2^n} \mid n \geq 0\}$
$L(M_2^\infty) = L(M_2)$

Wenn Sprache Turing-entscheidbar, dann erst recht erkennbar.

---

9 Neunte Woche

9.1 Pruefung

...

---

10 Zehnte Woche

10.1 Varianten von TM

$L_1 = \{w\#w \mid w \in \{0,1\}^*\}$

01##01
01#01

$L_2 = \{ww \mid w \in \{0,1\}^*\}$

␣010110
␣x1011x
␣xx01xx

$\sum_{k=0}^{n} 3^k$

Haskell: $sum[3i \mid i \leftarrow [0..12]]$

Varianten von TM

1. Mehrspurmaschine

$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}$

2. Mehrbandmaschine

3. Nichtdeterministische TM lässt sich simulieren

- λ-Kalkuel 1936 Church
- TM 1936 Turing

$(\lambda x.x + 1)(z + 3)$
$\rightarrow (z + 3) + 1$

- Rekursive Funktionen
- Goto
- While

Church-Turing-These
Jede im intuitiven Sinne berechenbare Funktion lässt sich durch eine TM berechnen.

□

Codierungen, Beispiel TM

$\Sigma_u = \{0, 1, ,, R, L, A, J, B, C, Q, D, W\}$

$|\Sigma_u| = 12$

$B0^s1^{(g-1)-s},C0^c,\underbrace{Q(A|J|0^q,)}_{\text{Startzustand}}\, D\, \underbrace{trans_0...trans_1}_{\delta} *$

$s = |\Sigma|$
$g = |\Gamma|$
$c = |C|$
$C = Q - \{q_{acc}, q_{rej}\}$

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{< M > \leftarrow \text{Codierung von } M}$

$*\underbrace{sym_0,...,sym_{w-1},\sqcup}_{< w > \text{ Wort}}$

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{< M, w >}$