



# Automatentheorie und Formale Sprachen

– Chomsky-  
Hierarchie –

Prof. Dr. Michael Neitzke

# Pumping Lemma für kontextfreie Sprachen

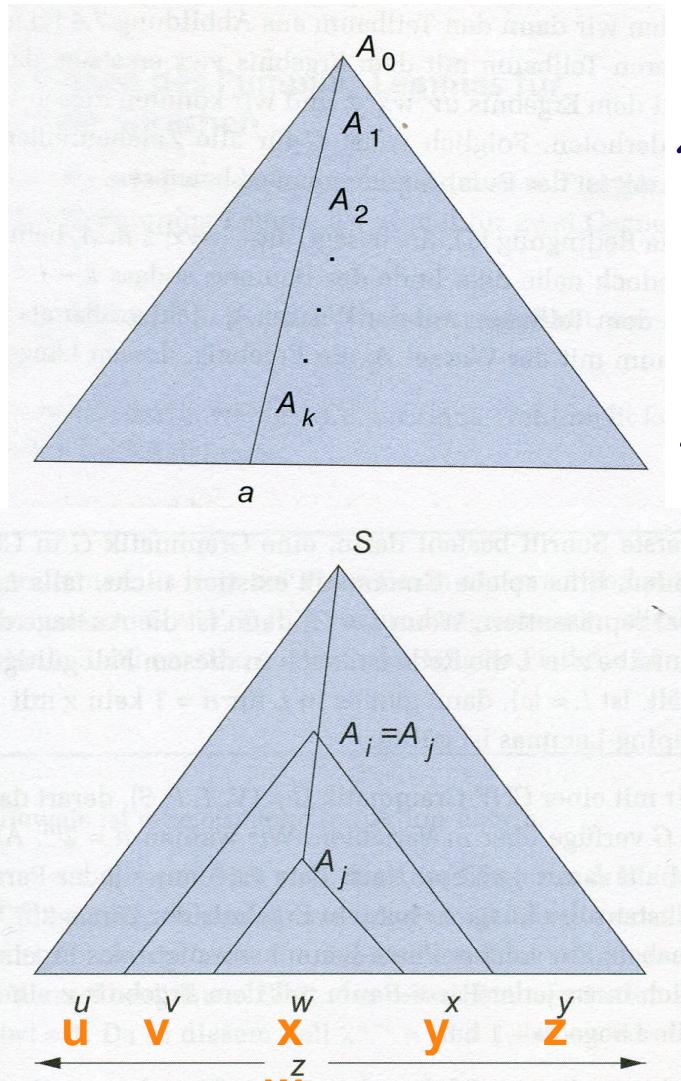
Sei  $L$  eine kontextfreie Sprache. Dann gibt es eine für  $L$  sprachspezifische Konstante  $n$ , für die gilt: Wenn  $w$  ein Wort aus  $L$  mit der Mindestlänge  $n$  ist, also  $|w| \geq n$ , dann kann man  $w$  in fünf Abschnitte  $uvxyz$  zerlegen, für die folgende Bedingungen erfüllt sind:

- 1)  $|vxy| \leq n$  d.h., dass die aufpumpbaren Abschnitte  $v$  und  $y$  sind (zusammen mit dem Mittelteil  $x$ ) höchstens so lang wie die Sprach-Konstante  $n$ .
- 2)  $vy \neq \epsilon$  d.h., mindestens einer der beiden aufpumpbaren Abschnitte ist nicht leer.
- 3) Alle Wörter  $uv^kxy^kz$ ,  $k \geq 0$  sind auch in  $L$  enthalten. Es werden also beide Abschnitte gleich oft aufgepumpt.

# Beweis des Pumping Lemmas für KFS

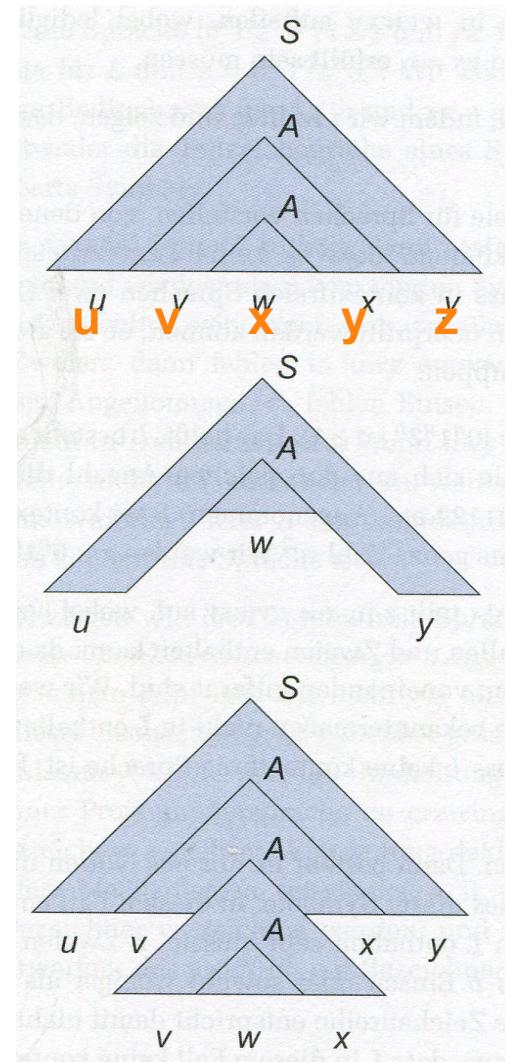
- Die Sprache sei durch eine Grammatik in CNF ( $A \rightarrow BC \mid a$ ) beschrieben.
- Die Grammatik habe **m** Variablen.
- Wir betrachten Wörter mit der Mindestlänge  **$n = 2^{m+1}$**
- Ein Parsebaum der Tiefe **m** kann maximal Wörter der Länge  **$2^{m-1}$**  abbilden. Für ein Wort mit der Mindestlänge  **$n = 2^{m+1}$**  muss er also mindestens die Tiefe  **$m+2$**  haben. Also gibt es einen Pfad aus  **$m+1$**  Variablen, sodass dort mindestens eine doppelt vorkommt.
  - Ein Pfad besteht aus einer Kette von Variablen (außer dem letzten Element), beginnend bei der Wurzel des Parsebaums

# „Aufpumpen“ beim Pumping Lemma für KFS



$S = A_0$

$A_1$   
 $A_2$   
 $\vdots$   
 $A_{m-1}$   
 $\vdots$   
 $A_m$



# Nicht-kontextfreie Sprachen

1)  $\{a^i b^i c^i \mid i \geq 1\}$

$$M \underline{\leq}^k x y^k z$$

2)  $\{a^i b^m c^i d^m \mid i, m \geq 1\}$

3)  $\{ww \mid w \in \{a,b\}^*\}$  „Kopiersprache“

Beweise durch Pumping Lemma:

1)  $w=a^n b^n c^n$  wegen  $|vxy| \leq n$  kann vxy nicht a's UND c's enthalten und auch der Fall „nur b's“ bleibt nicht in der Sprache

2)  $w=a^n b^n c^n d^n$  wegen  $|vxy| \leq n$  umfasst vxy nur ein oder zwei Symbole

3)  $w=a^n b^n a^n b^n$  (siehe nächste Folie)

3)  $\{ww \mid w \in \{a,b\}^*\}$ 

„Kopiersprache“

Fallunterscheidungen

4 SETs

1|2|3|4

3)  $w = a^n b^n a^n b^n$ wegen  $|vxy| \leq n$  umfasst vxy nur ein oder zwei

Symboleninhaltliche Teilwörter (SETs)

1. 2.

**Vielleicht findest du hier einen einfacheren Beweis?**

Analog &  
zur Übung

1. ein SET

2. zwei SETs

1. nur a's oder nur b's werden aufgezählt

(x)  $vxy = a^k b^l \quad (1 \leq k, l \leq n)$

(B)  $vxy = b^k a^l$

Unterscheidung der Reihenfolge a/b in vxy

(x1)  $u = a^{n-k} \quad z = b^{n-l} a^l b^u$

(x2)  $u = a^u b^u a^{n-k} \quad z = b^{n-l}$

Unterscheidung der Form der äußeren Teilwörter u und z

$$\begin{cases} (\alpha 1) \text{ oder } u = \epsilon \quad k = n \quad z = b^{n-l} a^u b^u \\ (\alpha 2) \text{ oder } u = a^u b^u a^{n-k} \quad z = \epsilon \quad l = n \end{cases} \quad \begin{array}{l} \text{enthalten} \\ \text{in } \alpha 1 / \alpha 2 \end{array}$$

analog (x1)

~~(x)~~  $v = \epsilon \quad [i=0 \text{ in } \alpha B]$

(x)  $v = a^i \quad x = a^i$

(x)  $v = a^i \quad x = a^{k-i} b^j$

(x)  $v = a^k b^l \quad x = b^i a^l$

Unterscheidung der Form der inneren Teilwörter v und x

(x)  $a^{n-k} a^{i-p} a^j (a^{k-i-j} b^l)^p b^{n-l} a^u b^u \xrightarrow{p>1} \not\approx b^{l+p+n-l} \neq b^u$

(x)  $a^{n-k} a^{i-p} a^{k-i-j} b^j (b^{l-j})^p b^{n-l} a^u b^u \xrightarrow{p=2} \not\approx b^{j+2(l-j)+n-l} \neq b^u \quad (j < l : xc1.1) \quad \not\approx a^{n-k+z_i+k-i} \neq a^u \quad (j = l : xc1.2)$

(x)  $a^{n-k} (a^k b^i)^p b^j (b^{l-j})^p a^u b^u \xrightarrow{p>1} \not\approx a^{n-k+k \cdot p} \neq a^u$

dieser Block auch vorne möglich

PL - Parameter  
Name  $\frac{P}{K}$   
(nicht  $\frac{P}{K}$ )

 $uv^p xy^p z$

# Abgeschlossenheit kontextfreier Sprachen

- Kontextfreie Sprachen sind abgeschlossen bezüglich
  - **Vereinigung**
  - **Verkettung**
  - **Hülle (unendliche Verkettung)**
  - **Spiegelung**
  - **Homomorphismus**
  - **Inversem Homomorphismus**
  - **Durchschnitt mit regulären Sprachen**
- Nicht abgeschlossen bezüglich
  - **Komplement**
  - **Durchschnitt**

# Kleine Übung zum Schnitt von Sprachen

$$R_1 = (01)^* + 1^*$$

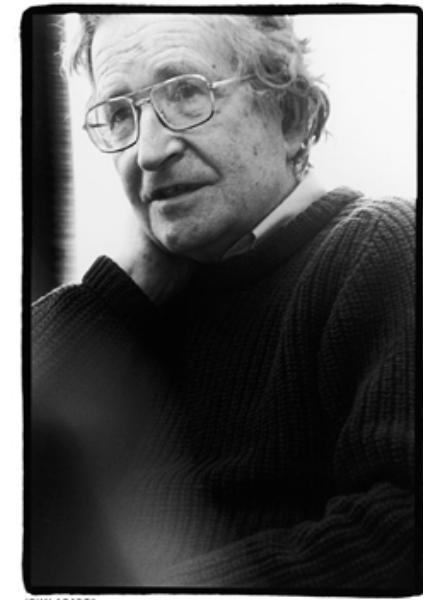
$$L_1 = \{ w \in \{0,1\}^k \mid \text{"Anzahl } 1 = \text{Anzahl } 0 \} \\ L_1 \cap R_1 = ?$$

# CH1: Chomsky-Hierarchie

- Hierarchie von Sprachklassen
- Typ 0 sind die aufzählbaren Sprachen (s.u.)
- Typ 1 sind die kontextsensitiven Sprachen (s.u.)
- Typ 2 sind die **kontextfreien** Sprachen
- Typ 3 sind die **regulären** Sprachen

# Chomsky

- Noam Chomsky
- Professor für Linguistik am MIT
- U. a. Forschungen zur Analyse und Erzeugung natürlicher Sprache
- Politisch aktiver Intellektueller
- Klassifizierung formaler Sprachen 1956
  - Vier Typen: Typ 0 bis Typ 3
  - Typ 0 ohne Einschränkung



JOHN SOARES

# Chomsky-Hierarchie mit Ergänzungen

Grammatik	Regeln	Sprachen	Automat	Abgeschlossenheit
Typ 0	$\alpha \rightarrow \beta$	aufzählbar	Turing Maschine	KSV*
Typ 1	$\alpha A\beta \rightarrow \alpha\gamma\beta$	kontextsensitiv	Linear platzbeschränkte, nicht-deterministische Turingmaschine	CKSV*
Typ 2	$A \rightarrow \gamma$	kontextfrei	Nichtdeterministischer Pushdown Automat	KV*
		LR(k)	Deterministischer Pushdown Automat	C
Typ 3	( $A \rightarrow Ba$ oder $A \rightarrow aB$ ) und $A \rightarrow a$	regulär	Endlicher Automat	CKSV*

A, B: Nichtterminale  
 C: Komplement  
 V: Vereinigung

a, b: Zeichenketten aus Terminalen  
 K: Verkettung  
 \*: Unendliche Verkettung

$\alpha, \beta, \gamma$ : beliebig  
 S: Schnitt

# Typ 0 und Typ 1 Grammatiken

## ■ Typ 0: Keine Einschränkungen

- Beispiel-Regel:

$$aXbbYZ \rightarrow ZcKc$$

## ■ Typ 1 Grammatiken

- Typ 1 monoton:

- Keine Regel hat links mehr Symbole als rechts

~~XbY → XZ  
3 > 2~~

- Typ 1 kontext-sensitiv:

- Nur ein Nonterminal darf ersetzt werden

$$aXbbYZ \rightarrow aXbbcWZ$$

- Beide Varianten sind äquivalent!

- Keine anschauliche Sprache bekannt, die nicht Typ 1 ist

- "Globale Korrelation (Typ 1) vs. lokale Unabhängigkeit (Typ 2)"

- Analogie: "Säugetier statt Katze" sagen entspricht einer zu allgemeinen Einordnung einer Grammatik

# Standardbeispiel für Typ 1 Grammatik: $a^n b^n c^n$

- I)  $S \rightarrow aSQ$
- II)  $S \rightarrow abc$
- III)  $bQc \rightarrow bbcc$
- IV)  $cQ \rightarrow Qc$

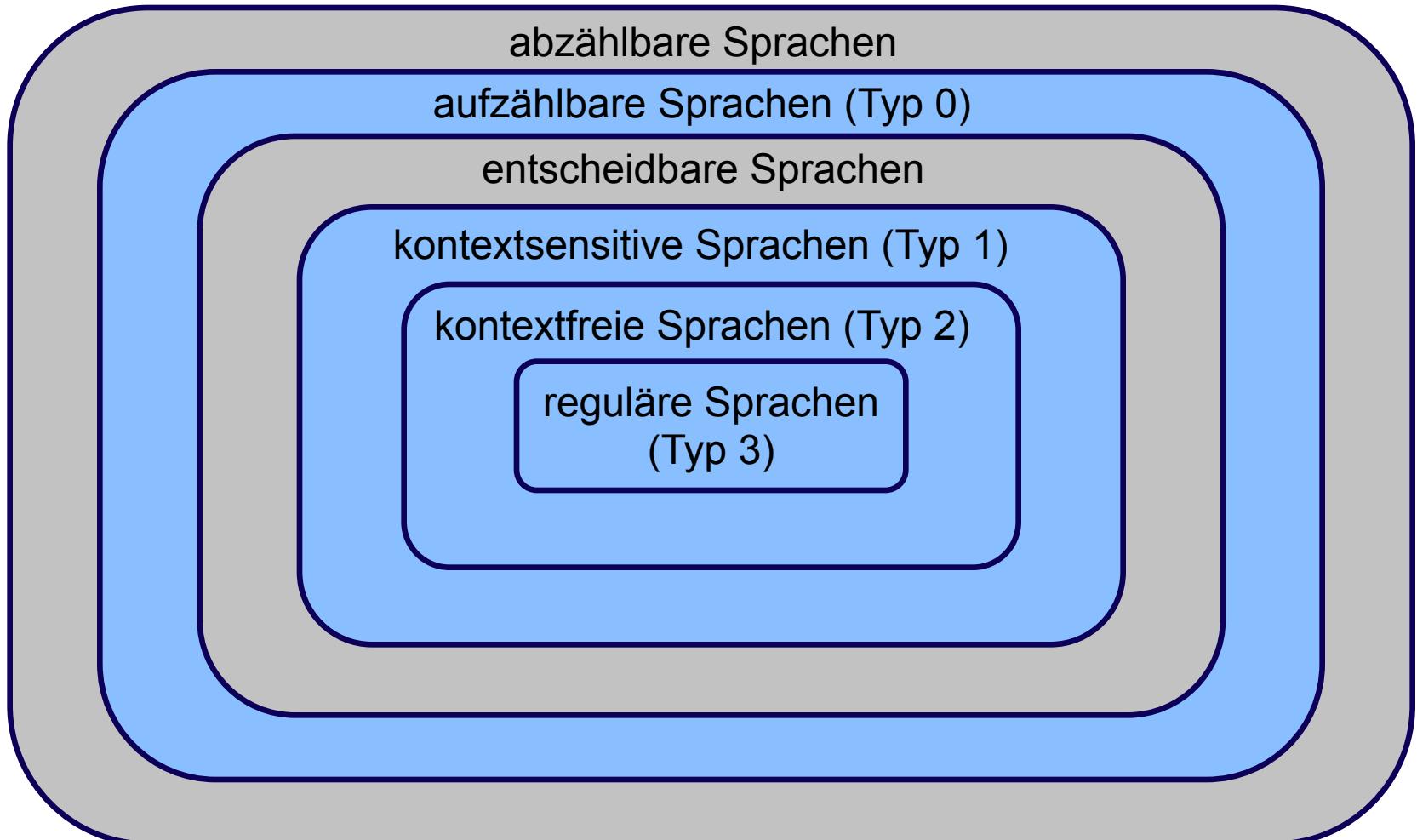
■ Übung: Leiten Sie **aaabbbccc** ab

■ Beschreiben Sie die allgemeine Reihenfolge der Regelanwendungen I-IV für  $a^n b^n c^n$

# "Typ 4"-Grammatiken

- "Finite Choice" Grammatiken
- Keine Nonterminale im rechten Teil einer Regel
- Also Startsymbol wird auf endliche Menge von alternativen Terminalsymbolfolgen abgebildet
  - Die Wörter der Sprache
- Praktisches Beispiel: Liste der Schlüsselwörter einer Programmiersprache

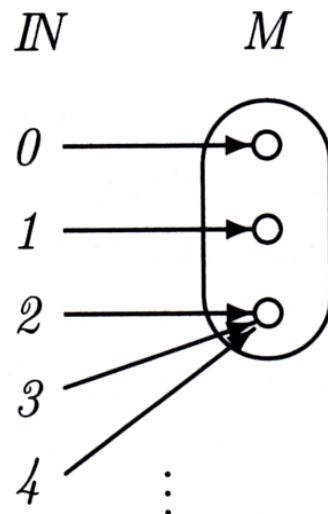
# Mengendiagramm für Sprachen



# Abzählbar

 **Definition:** Eine Menge  $M$  ist **abzählbar**, falls es eine Funktion  $f : \mathbb{N} \rightarrow M$  gibt, die surjektiv ist (d. h. alle Elemente aus  $M$  kommen als Bildelement vor), oder falls  $M = \emptyset$ . Ist eine Menge nicht abzählbar, so heißt sie **überabzählbar**.

 **Surjektiv:**

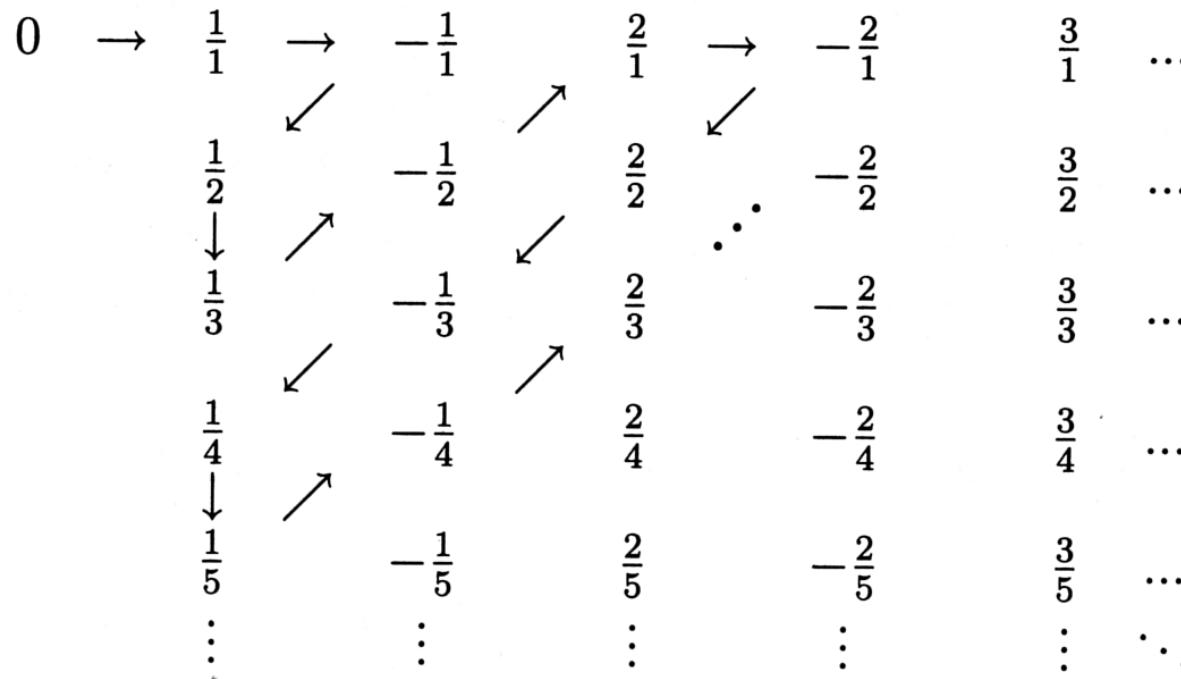


**Beispiel: Zweier-Potenzen**

$\mathbb{N}$	$M$
0	$1 = 2^0$
1	$2 = 2^1$
2	$4 = 2^2$
3	$8 = 2^3$
4	$16 = 2^4$
	$\vdots$

# Rationale Zahlen sind abzählbar

- In jedem Intervall unendlich viele rationale Zahlen
  - Dennoch abzählbar!
  - Cantorsches Diagonalverfahren (Georg Cantor, 1845-1918)



# Reelle Zahlen sind überabzählbar

## Widerspruchsbeweis

- Annahme: abzählbar
- Also:  $\mathbb{N} \rightarrow \mathbb{R}$
- Zahl in der Diagonalen wird in jeder Position verändert (z. B. 1 abziehen oder addieren).
- Diese Zahl müsste auch in irgendeiner Zeile stehen, z. B. in Zeile  $x$ .
- Aber das geht nicht, weil die Position  $(x, x)$  der Matrix einen anderen Wert aufweist als die neu erzeugte Zahl an der  $x$ -ten Stelle. Die neue Zahl ist von allen anderen verschieden.



Ende V10

# Alle formalen Sprachen sind abzählbar

 **Satz:** Die Menge aller Wörter  $\Sigma^*$  über einem Alphabet  $\Sigma$  ist abzählbar.

 **Beweis:** Wörter werden der Länge nach, dann lexikographisch geordnet

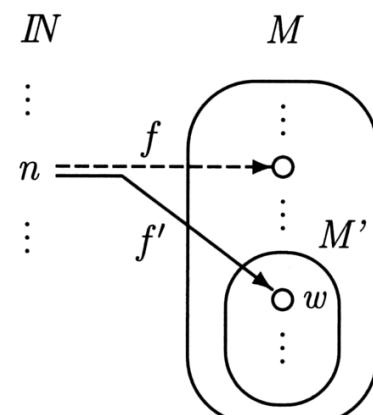
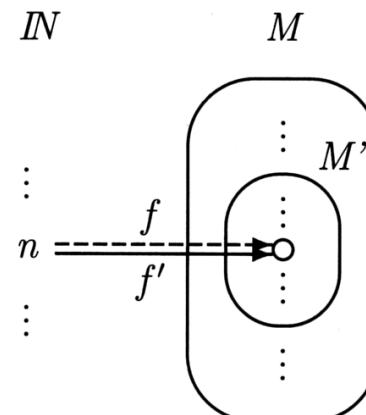
 **Satz:** Jede Teilmenge  $M'$  einer abzählbaren Menge  $M$  ist auch abzählbar.

 **Beweis:** Sei  $f : \mathbb{N} \rightarrow M$

  $f'(n) = f(n)$  falls  $f(n) \in M'$

  $f'(n) = w \in M'$  falls  $f(n) \notin M'$

 Wenn die gesamte Menge  $M$  abgedeckt ist, muss ja auch deren Teilmenge  $M'$  abgedeckt sein. Die Elemente aus  $\mathbb{N}$ , die ein Bildelement außerhalb von  $M'$  haben, werden auf ein beliebiges Bildelement aus  $M'$  abgelenkt.



# Aufzählbar

- **Definition:** Eine Menge  $M$  ist **aufzählbar** (auch rekursiv aufzählbar oder semi-entscheidbar), falls es eine surjektive Funktion  $f : \mathbb{N} \rightarrow M$  gibt, und einen **Algorithmus**, der es gestattet, für jedes  $n \in \mathbb{N}$  den Funktionswert  $f(n)$  zu berechnen, oder falls  $M = \emptyset$ .
- **Satz:** Die Menge aller Wörter  $\Sigma^*$  über einem Alphabet  $\Sigma$  ist aufzählbar.
- **Beweis:** Wörter können der Länge nach und dann lexikographisch geordnet erzeugt werden
- **Frage:** **Weshalb heißen aufzählbare Mengen auch semi-entscheidbar?**

# Entscheidbarkeit, Halteproblem

Eine **Eigenschaft** E einer Menge M ist eine (klar definierte) Teilmenge  $E \subseteq M$  genau der Elemente, welche die Eigenschaft besitzen.

Ein **Entscheidungsverfahren** ist ein Algorithmus, der für jedes Element einer Menge beantworten kann, ob es eine Eigenschaft hat oder nicht. Wenn es bewiesenermaßen kein solches Entscheidungsverfahren gibt, dann nennt man die Eigenschaft **unentscheidbar**.

Eine Eigenschaft auf einer Menge heißt **entscheidbar** (auch **rekursiv**, **rekursiv ableitbar**), wenn es ein **Entscheidungsverfahren** für sie gibt.

Das **Halteproblem** beschreibt die Frage, ob ein Algorithmus mit einer Eingabe terminiert. Alan Turing wies die Unentscheidbarkeit dieser Frage nach. Formaler ist das Halteproblem die Eigenschaft von Paaren von Algorithmus und Eingaben, dass der Algorithmus für die Eingabe terminiert, das heißt nur endlich lange rechnet.

M abzählbar :  $f: \mathbb{N} \rightarrow M$  surjektiv (oder  $M = \emptyset$ )

M aufzählbar :  $f: \mathbb{N} \rightarrow M$  surjektiv

und es gibt einen Algorithmus A mit  $n \xrightarrow{\quad A \quad} f(n)$  für  $n \in \mathbb{N}$   
(oder  $M = \emptyset$ )

entscheidbare Eigenschaft E einer Menge M ( $E \subseteq M$ )

$\Leftrightarrow$  Es gibt Algorithmus B mit

$$B(x) = \begin{cases} 1 & : x \in E \\ 0 & : x \in M \setminus E \end{cases}$$

$A(x)$  zählt die  
Eigenschaft E  
auf!

aufzählbar  $\hat{=}$  semi-entscheidbare

Algorithmus  $B(x)$ : Für ( $i = 1, i++$ ) wiederhole  
erfolgloser  
versuch eines Alg.  
aber.  $B$  terminiert nur für  $x \in E$

Wenn  $(A(i) = x)$   
dann  $B(x) = 1$

# Entscheidbarkeit, Halteproblem

Die Eigenschaft  $H_1 = \text{"Algorithmus A terminiert bei Eingabe 1"}$  ist auf der Menge  $M$  aller Algorithmen nicht entscheidbar. (Halteproblem)  
 $H_1$  ist abzählbar, aber nicht aufzählbar.

Sei  $g$  eine abzählende Funktion (Nummerierung) aller Algorithmen  
 $g: \mathbb{N} \rightarrow M$  und  $K$  ein (beliebiger) Algorithmus aus  $H_1$   
 $f: \mathbb{N} \rightarrow H_1$  mit  $f(n) = \begin{cases} K & : g(n) \notin H_1 \\ g(n) & : g(n) \in H_1 \end{cases}$  ist eine Abzählung von  $H_1$ !

# Entscheidbar

- **Definition:** Gegeben sei ein Alphabet  $\Sigma$ . Eine Sprache  $L \subseteq \Sigma^*$  heißt **entscheidbar**, falls es einen **abbrechenden Algorithmus**, Entscheidungsverfahren genannt, gibt, der für jedes  $w \in \Sigma^*$  feststellt, ob  $w \in L$  oder  $w \notin L$ .
- Genauer: Das Wortproblem ist entscheidbar
- Es gibt Sprachen, die aufzählbar, aber nicht entscheidbar sind:
  - $L \subset \text{ASCII}^*$ , daher aufzählbar
  - $L = \{xy \mid x \text{ ist ein Programm, } y \text{ ist eine Eingabe, und } x \text{ stoppt bei der Eingabe von } y \text{ nach endlich vielen Schritten}\}$
  - **Halteproblem !**

# Unentscheidbarkeit des Halteproblems

## Und dieser Beweis baut darauf auf

- Annahme: Entscheidbar, dann existiert Turingmaschine, die Matrix (unten) berechnet
- Turingmaschine bauen, die diese Turingmaschine (als Unterprogramm) enthält und für  $\omega_i$  genau das Gegenteil aus Feld  $(i,i)$  tut.

Nicht möglich, Widerspruch, denn diese Turingmaschine müsste ja irgendwo in der Matrix stehen, mit entgegengesetztem Verhalten

Turingmaschine  $T_1$  hält nicht bei Eingabe  $\omega_7$

	$\omega_1$	$\omega_2$	$\omega_3$	$\omega_4$	$\omega_5$	$\omega_6$	$\omega_7$
$T_1$	ja	ja	nein	ja	nein	ja	nein
$T_2$	nein	ja	nein	ja	ja	nein	ja
$T_3$	ja	ja	nein	nein	nein	ja	ja
$T_4$	ja	nein	nein	ja	ja	ja	nein
$T_5$	ja	ja	nein	ja	nein	nein	nein
$T_6$	nein	nein	ja	ja	ja	nein	ja
$T_7$	nein	nein	ja	ja	ja	nein	nein

# Vergleich von Grammatiken

■ Falsche Annahme: Je mächtiger, desto mehr Wörter

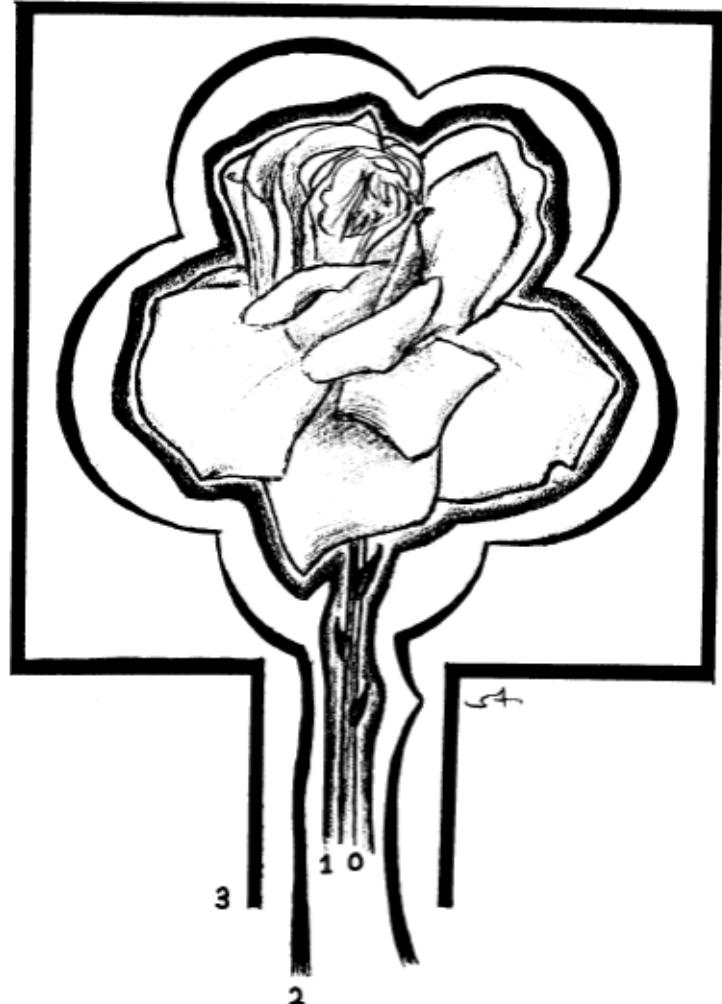
■ Umfangreichste Sprache für gegebenes Alphabet  $\Sigma$  kann durch reguläre Grammatik beschrieben werden:

$$S \rightarrow [\Sigma] S \mid \epsilon$$

■ Statt dessen: Grenze zwischen Wörtern in und außerhalb der Sprache hat einen komplexeren Verlauf *Bei wichtigeren Grammatiken*

# Analogie: Bild einer Rose

- Reguläre Sprache: Vertikale und horizontale Liniensegmente für Silhouette
- Kontextfreie Sprache: Beliebige Winkel für Liniensegmente plus Kreissegmente für Silhouette
- Kontextsensitive Sprache: Beliebiger Kurvenverlauf für die Silhouette
- Typ 0 Sprache: Perfektes Bild
- Rose selbst lässt sich nicht durch endliche Beschreibung vollständig erfassen



# CH1: Rückblick auf die Ziele

CH1	
Kenntnis	<ul style="list-style-type: none"><li>- Definitionen der rot hervorgehobenen Begriffe kennen</li><li>- Abschlusseigenschaften für kontextfreie Sprachen kennen</li></ul>
Verständnis	<ul style="list-style-type: none"><li>- Merkmale der verschiedenen Sprachklassen erklären können</li><li>- Das Pumping Lemma für kontextfreie Sprachen und seinen Einsatz in der Beweisführung im Detail erläutern können</li></ul>
Anwendung	
Analyse Synthese	<ul style="list-style-type: none"><li>- Eine vorgegebene Sprache hinsichtlich ihrer Sprachklassenzugehörigkeit beurteilen können</li><li>- Beweiskonzept für nicht kontextfreie Sprachen entwickeln können</li></ul>
Beurteilung	