

# Theoretische Informatik

Zusammenfassung

SoSe2024

## Inhaltsverzeichnis

<b>1</b>	<b>Allgemein</b>	<b>3</b>
1.1	Alphabete und Wörter . . . . .	3
1.2	Grammatiken . . . . .	3
<b>2</b>	<b>Chomsky-Hierarchie</b>	<b>4</b>
2.1	Typ 0 ( $\mathcal{L}_0$ ) - Phrasenstrukturgrammatiken . . . . .	4
2.2	Typ 1 ( $\mathcal{L}_1$ ) - Kontextsensitive Grammatiken . . . . .	4
2.3	Typ 2 ( $\mathcal{L}_2$ ) - Kontextfreie Grammatiken . . . . .	4
2.4	Typ 3 ( $\mathcal{L}_3$ ) - Reguläre Grammatik . . . . .	4
<b>3</b>	<b>Deterministischer Endlicher Automat (DEA)</b>	<b>5</b>
<b>4</b>	<b>Nicht-deterministischer Endlicher Automat (NEA)</b>	<b>6</b>
<b>5</b>	<b>Äquivalenz von DEA und NEA</b>	<b>7</b>
5.1	Satz von Rabin und Scott . . . . .	7
5.2	Potenzmengenkonstruktion ( $\text{NEA} \rightarrow \text{DEA}$ ) . . . . .	7
<b>6</b>	<b>Regex</b>	<b>8</b>
6.1	Satz von Kleene . . . . .	8
<b>7</b>	<b>Pumping Lemma</b>	<b>9</b>
<b>8</b>	<b>Satz von Myhill und Nerode</b>	<b>10</b>
<b>9</b>	<b>Minimalautomaten</b>	<b>12</b>
9.1	Table-Filling-Algorithmus . . . . .	12
<b>10</b>	<b>Kontextfreie Sprachen (<math>\mathcal{L}_2</math>)</b>	<b>13</b>
10.1	Chomsky Normalform (CNF) . . . . .	13
10.2	Greibach Normalform . . . . .	13

10.3 Konvertierung . . . . .	13
10.3.1 CNF $\rightarrow$ Greibach . . . . .	13
<b>11 Kellerautomaten</b>	<b>14</b>
11.1 CFG zu Kellerautomat . . . . .	15
<b>12 CYK-Algorithmus</b>	<b>16</b>
<b>13 Turing-Maschine</b>	<b>17</b>
13.1 Linear beschränkte Turing-Maschine . . . . .	18
<b>14 Satz von Kuroda</b>	<b>18</b>
<b>15 Berechnungskomplexität</b>	<b>19</b>
15.1 Zeitkomplexität . . . . .	19
15.2 Platzkomplexität . . . . .	19
<b>16 Komplexitätstheorie</b>	<b>19</b>
16.1 Zeitkomplexität . . . . .	19
16.2 Gödelisierung . . . . .	19

# 1 Allgemein

## 1.1 Alphabete und Wörter

- Ein Alphabet  $\Sigma$  ist eine endliche Menge unterscheidbarer Symbole
- Element  $\sigma \in \Sigma$  ist ein Zeichen des Alphabets  $\Sigma$
- Jedes Element  $\omega \in \Sigma^*$  ist ein Wort über  $\Sigma$
- $\varepsilon$  = Leeres Wort
- $\Sigma^*$ : Menge aller Wörter über  $\Sigma$
- $\Sigma^+$ : Menge aller Wörter über  $\Sigma$  mit mind. 1 Element
- $|\omega|$ : Länge eines Wortes ( $|\varepsilon| = 0$ )

## 1.2 Grammatiken

Eine Grammatik  $G$  ist ein 4-Tupel  $(V, \Sigma, P, S)$ :

- $V$ : endliche Menge an Nicht-Terminal-Symbolen
- $\Sigma$ : endliche Menge an Terminal-Symbolen ( $V \cap \Sigma = \emptyset$ )
- $P$ : endliche Menge an Produktionsregeln
- $S$ : Startsymbol ( $S \in V$ )

## 2 Chomsky-Hierarchie

### 2.1 Typ 0 ( $\mathcal{L}_0$ ) - Phrasenstrukturgrammatiken

- Beliebige Kombination aus T- und NT-Symbolen

### 2.2 Typ 1 ( $\mathcal{L}_1$ ) - Kontextsensitive Grammatiken

- $|l| \leq |r|$
- Länge des Wortes steigt
- $S \rightarrow \varepsilon$  erlaubt, wenn S auf **keiner** rechten Seite einer Regel steht!

**Beispiel:**

$$S \rightarrow S' \mid \varepsilon$$

$$S' \rightarrow aS'Bc \mid abc$$

$$cB \rightarrow Bc$$

$$bB \rightarrow bb$$

Das Nichtterminal S' braucht man nur, damit die Bedingung der Sonderregel erfüllt ist. Das Nichtterminal B wird mal zur Satzform Bc und mal zu bb, je nachdem ob B im **Kontext** c oder b steht.

### 2.3 Typ 2 ( $\mathcal{L}_2$ ) - Kontextfreie Grammatiken

Beim Ableiten in Typ-1-Grammatiken muss man immer aufpassen, dass das Nichtterminal auch im richtigen Kontext steht. Das Erzeugen von Sätzen ist viel leichter, wenn die Grammatik kontextfrei ist.

Eine Grammatik G ist vom Typ 2, wenn sie vom Typ 1 ist und zusätzlich auf der linken Seite jeder Regel genau **ein** Nichtterminal steht!

- $l \in V$
- $X \rightarrow \varepsilon$  immer erlaubt

### 2.4 Typ 3 ( $\mathcal{L}_3$ ) - Reguläre Grammatik

Eine Grammatik G ist vom Typ 3, wenn sie vom Typ 2 ist und zusätzlich folgende Regeln hat:

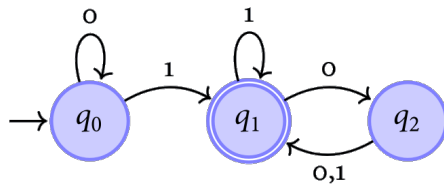
- $A \rightarrow b$
- $A \rightarrow bC$
- $A \rightarrow \varepsilon$

### 3 Deterministischer Endlicher Automat (DEA)

Eine DEA  $M$  ist ein 5-Tupel  $(Q, \Sigma, \delta, q_0, F)$ :

- $Q$ : endliche Zustandsmenge
- $\Sigma$ : endliches Alphabet
- $\delta: Q \times \Sigma \rightarrow Q$  Übergangsfunktionen
- $q_0$ : Startzustand
- $F$ : Menge der akzeptierten Endzustände

Beispiel:



- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- $q_0 = q_0$
- $F = q_2$
- $\delta$ :

$$\delta(q_0, 0) = q_0$$

$$\delta(q_0, 1) = q_1$$

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

$$\delta(q_1, 1) = q_1$$

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

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

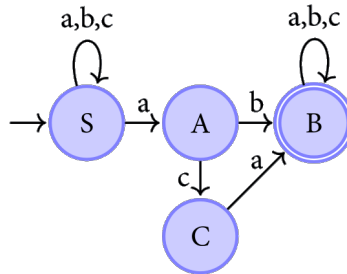
## 4 Nicht-deterministischer Endlicher Automat (NEA)

Eine NEA  $M$  ist ein 5-Tupel  $(Q, \Sigma, \delta, q_0, F)$ :

- $Q$ : endliche Zustandsmenge
- $\Sigma$ : endliches Alphabet
- $\delta: Q \times \Sigma \rightarrow Q$  Übergangsfunktionen
- $q_0$ : Menge der Startzustände
- $F$ : Menge der akzeptierten Endzustände

Beispiel:

$S \rightarrow aS \mid bS \mid cS \mid aA$   
 $A \rightarrow bB \mid cC$   
 $B \rightarrow aB \mid bB \mid cB \mid \varepsilon$   
 $c \rightarrow aB$



## 5 Äquivalenz von DEA und NEA

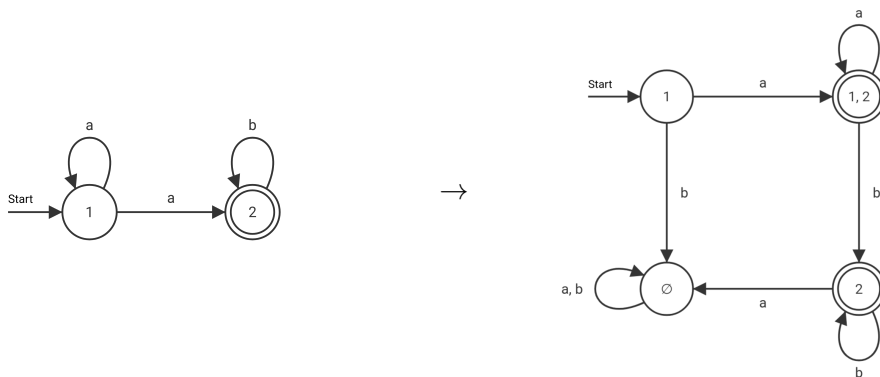
### 5.1 Satz von Rabin und Scott

Jede von einem NEA akzeptierte Sprache L ist auch von einem DEA akzeptierbar.

### 5.2 Potenzmengenkonstruktion (NEA $\rightarrow$ DEA)

Potenzmenge ist die Menge aller Teilmengen

Beispiel:



1. Potenzmenge bilden,  $\mathcal{P}\{1, 2\} = \{\{\emptyset\}, \{1\}, \{2\}, \{1, 2\}\}$
2. Jede Teilmenge ist Zustand des DEA
3. Wohin kommt man von von jedem NEA Zustand?
  - $\delta(\{1\}, a) = 1 \wedge 2 \rightarrow \{1, 2\}$
  - $\delta(\{1\}, b) = \{\emptyset\}$
  - $\delta(\{2\}, a) = \{\emptyset\}$
  - $\delta(\{2\}, b) = \{2\}$
  - $\delta(\{2\}, b) = \{2\}$
  - $\delta(\{1, 2\}, a) = \delta(\{1\}, a) \cup \delta(\{2\}, a) = 1 \wedge 2 \cup \emptyset = \{1, 2\}$
  - $\delta(\{\emptyset\}, a) \wedge \delta(\{\emptyset\}, b) = \emptyset \rightarrow \{\emptyset\}$
4. Startzustand bleibt gleich
5. Jede Teilmenge, in der der ursprüngliche Endzustand vorkommt, wird wieder zum Endzustand,  $F = 2 \rightarrow F' = \{\{2\}, \{1, 2\}\}$

## 6 Regex

- `.` : Beliebiges einzelnes Zeichen außer Zeilenumbruch
- `*` : Null oder mehr Wiederholungen
- `+` : Eine oder mehr Wiederholungen
- `{n}` : Genau n Wiederholungen
- `{n,}` : Mindestens n Wiederholungen
- `{n, m}` : Zwischen n und m Wiederholungen
- `[ ]` : Zeichenklasse (z.B. `[a-z]` für alle Kleinbuchstaben)
- `[ ^ ]` : Negierte Zeichenklasse

Email-Adressen: `[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}`

### 6.1 Satz von Kleene

Die Menge der durch reguläre Ausdrücke (Regex) beschreibbaren Sprachen ist genau die Menge der regulären Sprachen.

→ Alle endlichen Sprachen sind durch reguläre Ausdrücke beschreibbar



## 7 Pumping Lemma

- Wenn  $L$  erkennbar ist, dann existiert ein  $p \in \mathbb{N}$
- sodass für alle Wörter  $w \in L$  mit  $|w| \geq p$  gilt:
- Es gibt eine Zerlegung  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq p$ ,
- sodass für alle  $i \in \mathbb{N}$  gilt:
- $xy^iz \in L$

Beispiele:

$$L = \{w \in \{a, b\}^* \mid |w|_a > |w|_b\}$$

Also eine Sprache, die aus beliebig vielen a und b besteht, aber immer mehr a als b hat.

Sei  $p \in \mathbb{N}$  gegeben

Da unser Wort  $|w| \geq p$  sein muss, bietet sich  $a^{p+1}b^p$  an.

Da  $|xy| \leq p$  gilt, besteht  $xy$  also nur aus a.

Mit der Zerlegung in  $w = xy^iz$  und  $i = 0$ , hätte man  $|xz|_a \leq |xz|_b$ , also weniger a als b.

$$L = \{a^{n^2} \mid n \in \mathbb{N}\}$$

Sei  $p \in \mathbb{N}$  gegeben

Da unser Wort  $|w| \geq p$  sein muss, bietet sich  $w = a^{p^2}$  an.

Sei Zerlegung  $w = xy^iz$  mit  $|xy| \leq p$  gegeben.

Erstes Wort ist  $p^2 = |xyz|$

Nächst größere Wort ist  $(p+1)^2 = p^2 + 2p + 1$

Setze  $i=2$ , betrachte  $xy^2z$ :

$$|xyz| < |xy^2z|$$

$$\begin{aligned} |xy^2z| &= |xyz| + |y| \\ &\leq p^2 + p \quad (\text{Da } y \text{ nach } |xy| \leq p \text{ höchsten Länge } p \text{ hat}) \end{aligned}$$

$$p^2 + p < p^2 + 2p + 1$$

Somit  $xy^2z \notin L$

## 8 Satz von Myhill und Nerode

Eine Sprache  $L$  ist genau dann regulär, wenn der Index  $R_L$  endlich ist, also wenn es endlich viele Äquivalenzklassen gibt!

$$xR_Ly \Leftrightarrow [\forall w \in \Sigma^* : xw \in L \Leftrightarrow yw \in L]$$

### Beispiele:

$$L = \{w \in \{a, b\}^* \mid w \text{ enthält gerade Anzahl von a's}\}$$

→ Beliebige Kombination aus a's und b's, aber immer eine gerade Anzahl an a's.

1. Äquivalenzklassen bestimmen
  - Wörter mit gerade Anzahl an a's
  - Wörter mit ungerade Anzahl an a's
2. Endliche Anzahl an Äquivalenzklassen und somit regulär

$$L = \{w \in \{0, 1\}^* \mid w \text{ enthält die Unterfolge 01}\}$$

1. Äquivalenzklassen bestimmen
  - $[\varepsilon]$  Wörter, die noch keine 0 enthalten
  - $[0]$  Wörter, die mindesten eine 0 enthalten, aber noch keine 01
  - $[1*]$  Wörter, die mindesten eine 1 enthalten, aber keine 01
  - $[01]$  Wörter, die mindestens ein 01 enthalten haben
2. Endliche Anzahl an Äquivalenzklassen und somit regulär

$$L = \{a^n b^n \mid n \geq 0\}$$

1. Zeigen dass gilt:  $xz \in L \Leftrightarrow yz \in L$
2. Betrachten wir Wörter der Forma  $a^i$  und  $a^j$  mit  $i \neq j$ . Sei  $i < j$ , dann:
  - Wenn  $z = b^i$ , dann ist  $a^i b^i \in L$
  - Wenn  $z = b^i$ , dann ist  $a^j b^i \notin L$ , weil  $i < j$
3. Da es immer möglich ist, ein  $z$  zu finden, das  $a^i$  und  $a^j$  unterschiedlich behandelt, sind  $a^i$  und  $a^j$  nicht äquivalent!
4. Für jedes  $n \geq 0$  ist  $a^n$  in einer eigenen Äquivalenzklasse
5. Dies bedeutet, dass es unendlich viele unterschiedliche Äquivalenzklassen gibt, nämlich eine für jedes  $n$ !
6. Folgt:  $L$  ist nicht regulär

$$L = \{a^p \mid p \text{ ist eine Primzahl} \}$$

1. Zeigen dass gilt:  $xz \in L \Leftrightarrow yz \in L$
2. Betrachten wir Wörter der Forma  $a^i$  und  $a^j$  mit  $i \neq j$ . Sei  $i$  eine Primzahl und  $j$  nicht, dann:
  - $a^i \in L$  (weil  $i$  eine Primzahl ist)
  - $a^j \notin L$  (weil  $j$  keine Primzahl ist)
3. Für jedes  $n \geq 2$ , das eine Primzahl ist, und jedes  $m$ , das keine Primzahl ist:  
 $a^n \equiv_L a^m \Leftrightarrow n = m$
4. Dies bedeutet, dass es unendlich viele unterschiedliche Äquivalenzklassen gibt, nämlich eine für jede Primzahl
5. Folgt:  $L$  ist nicht regulär


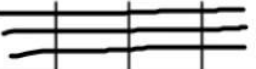

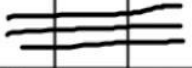

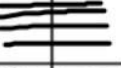



## 9 Minimalautomaten

### 9.1 Table-Filling-Algorithmus

Nur für DEAs!

1. Eventuell nicht erreichbare Zustände entfernen
2. Tabelle aus Zuständen erstellen  $\{q, q'\}, q \neq q'$
3. Zustandspaar markieren, bei dem immer **ein** Zustand ein Endzustand ist
4. Wiederhole solange, bis keine Markierungen mehr dazu kommen
  - Für jedes unmarkierte Paar  $\{q, q'\}$  und jeden Buchstaben  $a \in \Sigma$ 
    - Wenn  $\{\delta(q, a), \delta(q', a)\}$  markiert ist, dann das Paar  $\{q, q'\}$  selbst markieren
5. Verschmelze unmarkierte Zustandspaare

Beispiel Tabelle  $\{q, q'\}, q \neq q'$ :

z0					
z1					
z2					
z3					
z4					
	z0	z1	z2	z3	z4

## 10 Kontextfreie Sprachen ( $\mathcal{L}2$ )

### 10.1 Chomsky Normalform (CNF)

Regeln müssen folgende Formen haben:

- $A \rightarrow BC$
- $A \rightarrow a$
- $S \rightarrow \varepsilon$

### 10.2 Greibach Normalform

Eine  $\varepsilon$ -freie, kontextfrei Grammatik mit folgenden Regeln:

- $A \rightarrow aB_1B_2B_3\dots B_k$
- $k \geq 0$

### 10.3 Konvertierung

#### 10.3.1 CNF $\rightarrow$ Greibach

Beispiel:

$S \rightarrow AB \mid BC$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

- Die 3 letzteren Regeln sind bereits in der Greibach Normalform, da  $A \rightarrow a = A \rightarrow a\varepsilon$
- $S \rightarrow AB$  ist nicht in der GNF. Wir müssen sie umschreiben, damit es ein Terminal als erstes Symbol hat
- Wir wissen, dass:  $A \rightarrow a$
- Deshalb kann  $AB$  umgeschrieben werden als:  $AB \rightarrow aB$
- $S$  wird also zu:  $S \rightarrow aB \mid BC$
- Das selbe mit  $B$ , da  $B \rightarrow b$ :
- $S$  wird also zu:  $S \rightarrow aB \mid bC$

Konvertierte Greibach Normalform:

$S \rightarrow aB \mid bC$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

## 11 Kellerautomaten

Ein Kellerautomat (PDA)  $M$  ist ein 6-Tupel  $(Q, \Sigma, \Gamma, \delta, q_0, \#)$ :

- $Q$ : endliche Zustandsmenge
- $\Sigma$ : endliches Bandalphabet
- $\Gamma$ : endliches Kelleralphabet
- $\delta$ : Übergangsfunktionen
- $q_0$ : Startzustand ( $q_0 \in Q$ )
- $\#$ : Ursprüngliches Kellersymbol ( $q_0 \in \Gamma$ )

Akzeptanz:

- Kein akzeptierender Endzustand!
- Akzeptanzkriterien für Wörter  $x \in |\Sigma|^*$ :
  1. Wort komplett gelesen
  2. Keller (Stack leer)

Nicht-Determinismus:

- Mehrere simultane Übergänge möglich
- Spontane Übergänge ( $a = \varepsilon$ ) möglich

Konfiguration eines PDA gegeben durch 3-Tupel  $(Q, \Sigma^*, \Gamma^*)$ :

- $q \in Q$ : Momentaner Zustand
- $w' \in \Sigma^*$ : Noch zu lesender Anteil der Eingabe
- $\gamma \in \Gamma^*$ : Aktueller Kellerinhalt

Übergangsfunktion:

- $\delta(q, a, A) \ni (q', B_1 B_2 \dots B_k)$
- Wenn Automat in Zustand  $q$  ist, das Symbol  $a$  liest und  $A$  oben auf Stack liegt, wechselt er in Zustand  $q'$  und ersetzt das  $A$  auf dem Stack durch  $B_1 B_2 \dots B_k$

## 11.1 CFG zu Kellerautomat

Beispiel:

$$S \rightarrow ASbb \mid bT$$

$$T \rightarrow Tba \mid Sb \mid \varepsilon$$

Kellerautomat besteht aus nur einem Zustand!

Übergangsfunktionen:

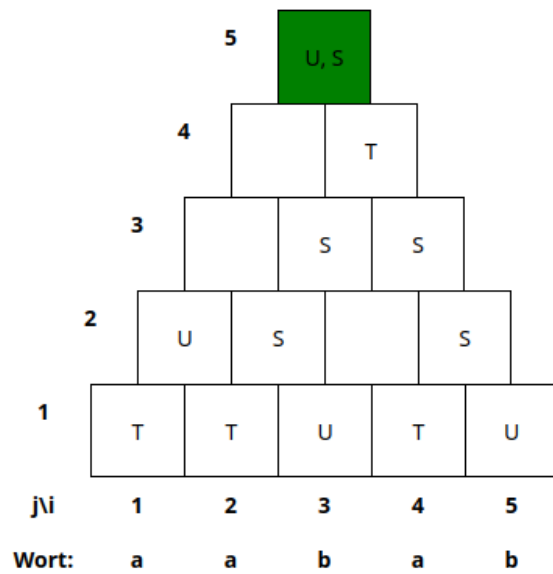
- $\delta(q_0, \varepsilon, S) = (q_0, aSbb)$
- $\delta(q_0, \varepsilon, S) = (q_0, bT)$
- $\delta(q_0, \varepsilon, T) = (q_0, Tba)$
- $\delta(q_0, \varepsilon, T) = (q_0, Sb)$
- $\delta(q_0, \varepsilon, T) = (q_0, \varepsilon)$
- $\delta(q_0, a, a) = (q_0, \varepsilon)$
- $\delta(q_0, b, b) = (q_0, \varepsilon)$

## 12 CYK-Algorithmus

Beispiel:

- $S \rightarrow ST \mid TU \mid US$
- $T \rightarrow SS \mid a$
- $U \rightarrow TT \mid b$

Wort: **aabab**



Nur wenn S (Startsymbol) ganz oben in der Pyramide steht, wird das Wort akzeptiert!



## 13 Turing-Maschine

Eine Turing-Maschine  $M$  ist ein 7-Tupel  $(Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ :

- $Q$ : endliche Zustandsmenge
- $\Sigma$ : endliches Eingabealphabet
- $\Gamma$ : endliches Arbeitsalphabet ( $\Sigma \subset \Gamma$ )
- $\delta$ : Übergangsfunktionen
- $q_0$ : Startzustand ( $q_0 \in Q$ )
- $\square$ : Blank-Symbol ( $\square \in \Gamma - \Sigma$ )
- $F$ : Menge der Endzustände

Übergangsfunktionen:

- Deterministisch:  $\delta(q, a) = \delta(q', b, x)$ 
  1.  $M$  befindet sich in Zustand  $q$  und liest  $a$  vom Band
  2.  $M$  geht in Zustand  $q'$  über und ersetzt das  $a$  mit einem  $b$
  3.  $M$  führt Kopfbewegung  $x \in \{l, n, r\}$
- Nicht-deterministisch:  $\delta(q, a) \ni \delta(q', b, x)$

Konfiguration einer TM ist ein Wort  $k \in \Gamma^* Q \Gamma^*$ :

$$k = \alpha q \beta$$

$$k = \alpha_1 \dots \alpha_m q \beta_1 \dots \beta_n$$

- $q$ : Aktueller Zustand
- $\alpha$ : Wort links des Schreib/Lese-Kopfes
- $\beta$ : Wort rechts des Kopfes

Startkonfiguration  $q_0 \vec{w}$ :

- $w \in \Sigma^*$  steht auf Band
- $M$  in Zustand  $q_0$
- S/L-Kopf steht auf erstem Buchstaben von  $w$
- Restliches Band mit Blanks  $\square$  befüllt

### 13.1 Linear beschränkte Turing-Maschine

Eine Turing Maschine  $M$  ist linear beschränkt, wenn  $|\vec{w}| < \infty$ , also endlich ist.

## 14 Satz von Kuroda

Die von linear beschränkten, nicht-deterministischen Turing-Maschinen akzeptierbaren Sprachen sind genau die kontextsensitiven Sprachen  $\mathcal{L}_1$

## 15 Berechnungskomplexität

### 15.1 Zeitkomplexität

$T_M(\vec{x}) \equiv$  Anzahl der Schritte des Automaten mit Eingabe  $\vec{x} \in \Sigma^*$

### 15.2 Platzkomplexität

$S_M(\vec{x}) \equiv$  Anzahl *verschiedener* Zellen, die der Automat bei Eingabe  $\vec{x} \in \Sigma^*$  besucht

## 16 Komplexitätstheorie

### 16.1 Zeitkomplexität

- **P**: Von einer Deterministischen Turing-Maschine in polyzeit Lösbar
- **NP**: Von einer Nicht-deterministischen Turing-Maschine in polyzeit Lösbar

### 16.2 Gödelisierung

Beispiel:

Angenommen, wir haben ein formales System mit den Symbolen  $\{a, b, c\}$  denen wir die Zahlen  $\{1, 2, 3\}$  zugeordnet haben. Eine Sequenz der Symbole 'a, b, c' wird dann wie folgt codiert:

- $a \rightarrow 1$
- $b \rightarrow 2$
- $c \rightarrow 3$

Die Sequenz 'a, b, c' wird zu der Zahl  $2^1 \times 2^2 \times 5^3 = 2 \times 9 \times 125 = 2250$  codiert.