Theoretische Informatik 3. Semester

Prof. Dr. Martin Plümicke

6. Dezember 2023

Inhaltsverzeichnis

5 Turingmaschine als Erkennungsautomat für Chomsky–1 und Chomsky–0 Sprachen

5.1 Turingmaschine als Erkennungsautomat für formale Sprachen

Beispiel (Turingmaschine zum Erkennen einer Sprache)**.** Zunächst geben wir einmal die Spezifikation für einen Automaten, der eine Sprache erkennt, an.

Deklarationen: $< \Sigma; * >$ Alphabet der Sprache mit der Operation *, die die Menge aller Wörter (jede beliebige Kombination aus Zeichen des Alphabets Σ) bildet.

Eingabe: $e \in \Sigma*$ (zu prüfendes Wort)
$L \subseteq \Sigma*$ (Sprache)

Ausgabe: $a \in \{true, false\}$

Nachbedingung: $a = (e \in L)$.

In Worten lässt sich die Spezifikation wie folgt zusammenfassen: Automaten, die eine Sprache erkennen, sollen als Eingabe eine beliebige Kombination von Zeichen akzeptieren und als Ausgabe angeben, ob die Eingabe in der vorgegebenen Sprache liegt.

Nun wollen wir das Alphabet $\Sigma = \{0, 1\}$ und die Sprache L, die alle Worte enthält, die zunächst hintereinander n-mal die 0 und dann n-mal die 1 haben ($L = \{0^n 1^n \mid n \geq 1\}$), betrachten.

Idee der Konstruktion einer Turingmaschine TM:

Am Anfang steht das Eingabewort w $\in$ {0,1 }* auf dem Band.

- TM ersetzt abwechselnd eine 0 durch ein a und eine 1 durch ein b

- Wird im gleichen Durchgang die letzte 0 und die letzte 1 ersetzt, ist das Wort akzeptiert

Folgende Zustände:

- TM beginnt im Startzustand q_0 und zeigt auf die am weitesten links stehende 0.
- Nach Ersetzen von 0 durch a Wechsel in Zustand q_1, beim Bewegen des LSK nach rechts werden alle 0 und b überlesen, bis 1 erreicht wird
- Dann wird diese durch b ersetzt, in den Zustand q_2 gewechselt
- Im Zustand q_2 geht es nach links zurück, bis auf ein a gestoßen wird, dann Rechtsschritt und Wechsel in den Zustand q_0
- Sind alle 0 und 1 durch a und b überschrieben, dann trifft TM bei Rechtsbewegung im Zustand q_0 auf ein b, dann Wechsel nach q_3
- Vom Zustand q_3 geht es unter b weiter nach rechts
- Wird in diesem Zustand auf B getroffen, dann Wechsel in Endzustand q_4

Wenn die Turingmaschine TM im Finalzustand q_4 anhält, ist das Wort in der Sprache L enthalten. Hält die Turingmaschine in einem anderen Zustand an, so ist das Wort nicht in der Sprache L enthalten.

Formal sieht die Turingmaschine dann wie folgt aus:

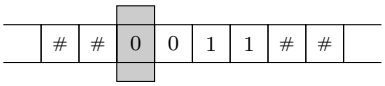
$$TM = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, a, b, \#\}, \delta, q_0, \#, \{q_4\})$$

mit der Übergangsfunktion δ:

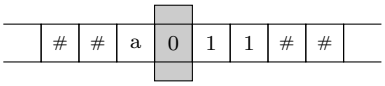
		Symbol				
		0	1	a	b	#
Zustand	q_0	(q_1, a, R)	-	-	(q_3, b, R)	-
	q_1	$(q_1, 0, R)$	(q_2, b, L)	-	(q_1, b, R)	-
	q_2	$(q_2, 0, L)$	-	(q_0, a, R)	(q_2, b, L)	-
	q_3	-	-	-	(q_3, b, R)	$(q_4, \#, L)$
	q_4	-	-	-	-	-

Wir rechnen die Turingmaschine nun mit einem Wort als Eingabe durch, das in der Sprache L enthalten ist:

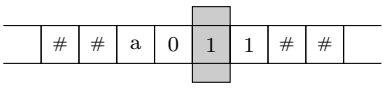
- Eingabe von 0011, Startzustand q_0:



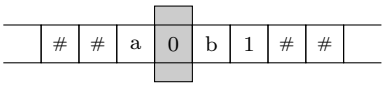
- $(q_0, 0) \mapsto (q_1, a, R)$:



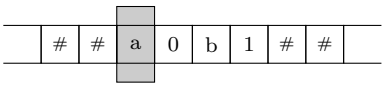
- $(q_1, 0) \mapsto (q_1, 0, R)$:



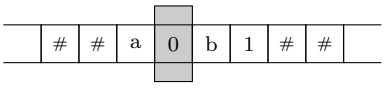
- $(q_1, 1) \mapsto (q_2, b, L)$:



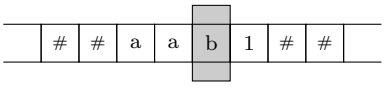
- $(q_2, 0) \mapsto (q_2, 0, L)$:



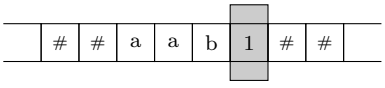
- $(q_2, a) \mapsto (q_0, a, R)$:



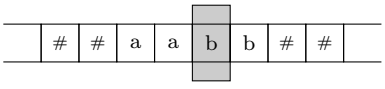
- $(q_0, 0) \mapsto (q_1, a, R)$:



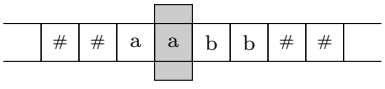
- $(q_1, b) \mapsto (q_1, b, R)$:



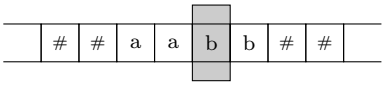
- $(q_1, 1) \mapsto (q_2, b, L)$:



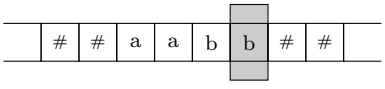
- $(q_2, b) \mapsto (q_2, b, L)$:



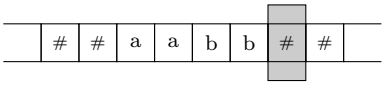
- $(q_2, a) \mapsto (q_0, a, R)$:



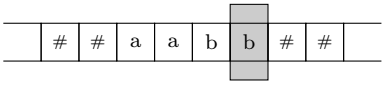
- $(q_0, b) \mapsto (q_3, b, R)$:



- $(q_3, b) \mapsto (q_3, b, R)$:



- $(q_3, \#) \mapsto (q_4, \#, L)$:

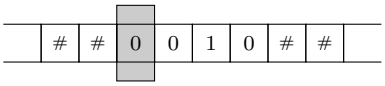


- TM hält an und befindet sich im Finalzustand q_4

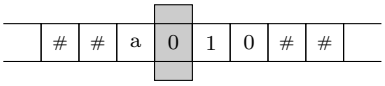
  $\Rightarrow$ Eingabe ist akzeptiert

Nun rechnen wir die Turingmaschine auch noch mit einer Eingabe durch, die nicht in der Sprache L enthalten ist.

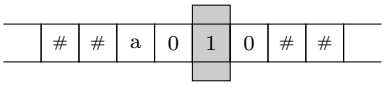
- Eingabe von 0010, Startzustand q_0:



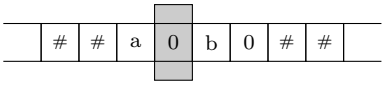
- $(q_0, 0) \mapsto (q_1, a, R)$:



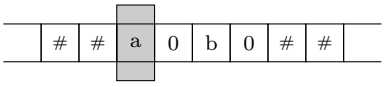
- $(q_1, 0) \mapsto (q_1, 0, R)$:



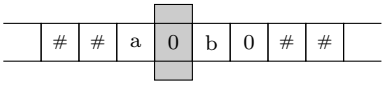
- $(q_1, 1) \mapsto (q_2, b, L)$:



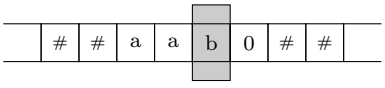
- $(q_2, 0) \mapsto (q_2, 0, L)$:



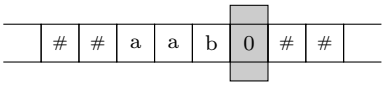
- $(q_2, a) \mapsto (q_0, a, R)$:



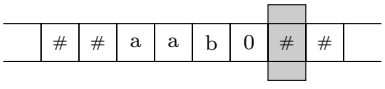
- $(q_0, 0) \mapsto (q_1, a, R)$:



- $(q_1, b) \mapsto (q_1, b, R)$:



- $(q_1, 0) \mapsto (q_1, 0, R)$:



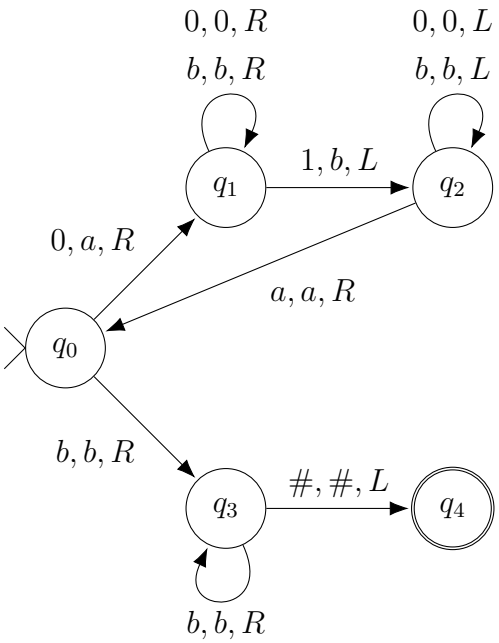
- $(q_1, \#) \mapsto \perp$

  $\Rightarrow$ TM hält an und $q_1 \notin F \Rightarrow$ Eingabe wird nicht akzeptiert

Turingmaschinen kann man auch in Diagrammdarstellungen visualisieren:

- Jeder Zustand wird durch einen Knoten symbolisiert.
- Der Startzustand erhält eine eingehende Kante ohne Quelle.
- Die Endzustände werden doppelt umrandet dargestellt.
- Gibt es zwischen zwei Zuständen q und q‘ einen Übergang mit δ (q, a) = (q‘, b, D), dann wird zwischen q und q‘ eine gerichtete Kante gezeichnet mit der Beschriftung „a, b, D“.

Als Beispiel betrachten wir die Turingmaschine aus Beispiel 5.1 visualisiert:



Im Folgenden betrachten wir die Sprachen, die Turingmaschinen *bearbeiten* können. Dabei werden wir feststellen, dass Turingmaschinen eine Klasse zwischen Chomsky–1 und Chomsky–0 akzeptieren können und die Chomsky–0 Sprachen *aufzählen* können.

5.2 Chomsky-1 und Chomsky-0 Sprachen

In diesem Abschnitt beschäftigen wir uns mit den Erkennungsautomaten zu den jeweiligen Sprachtypen. Wir haben bereits für Chomsky-3 die NEA/DEA und für Chomsky-2 die Kellerautomaten kennengelernt. Im Folgenden wenden wir uns nun Chomsky-1- und Chomsky-0-Sprachen zu.

Definition 5.1 (Abzählbare Menge)**.** Eine Menge M heißt abzählbar, falls es eine surjektive Funktion $f : \mathbb{N} \to M$ gibt oder falls $M = \emptyset$ ist.

Beispiel. Vergleiche Theo. Inf. 1:

- endliche Mengen
- $\mathbb{N}$ $\quad 0 \mapsto 0$
  $1 \mapsto 1$
  $2 \mapsto 2$
  $\dots$

- $\mathbb{Z}$

$0 \mapsto 0$

$1 \mapsto 1$

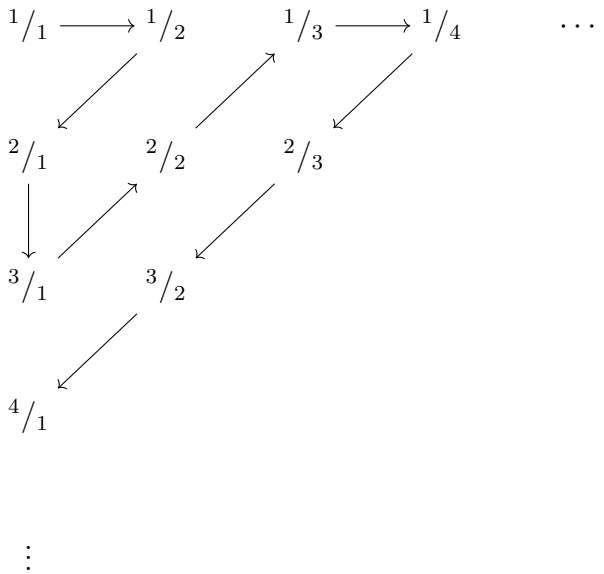
$2 \mapsto -1$

$3 \mapsto 2$

$4 \mapsto -2$

$\dots$

- $\mathbb{Q}$



Satz 5.2. Die Menge Σ^* über einem Alphabet Σ ist abzählbar.

Beweis. Zunächst definiert man eine totale Ordnung auf Σ (z.B. alphabetisch geordnet: $a < b < c < \dots < z$):

1. Man ordnet $w \in \Sigma$ der Länge nach
2. Wörter mit gleicher Länge ordnet man lexikographisch

Für jede Länge gibt es endlich viele Wörter. Man nummeriert sie durch und erhält so eine Abbildung $f : \mathbb{N} \to \Sigma^*$. □

Beispiel. $\Sigma = \{a, b\} \Rightarrow \Sigma^* = \{a, b, aa, ab, ba, bb, \dots\}$

Abbildung $f : \mathbb{N} \to \Sigma^*$:

$0 \mapsto \varepsilon$	$3 \mapsto aa$	$6 \mapsto bb$	$9 \mapsto aba$
$1 \mapsto a$	$4 \mapsto ab$	$7 \mapsto aaa$	$10 \mapsto abb$
$2 \mapsto b$	$5 \mapsto ba$	$8 \mapsto aab$	$\dots$

Satz 5.3. Jede Teilmenge $M' \subseteq M$ einer abzählbaren Menge ist selbst auch abzählbar.

Beweis. Sei $f : \mathbb{N} \to M$ eine surjektive Funktion. Dann sei $f' : \mathbb{N} \to M'$ definiert durch

$$f'(n) = \begin{cases} f(n) & \text{falls } f(n) \in M' \\ m \in M' \text{ beliebig} & \text{falls } f(n) \notin M' \end{cases}$$

f' ist surjektiv und M' damit abzählbar. □

Korollar 5.4. Sei $\mathcal{L} \subseteq \Sigma^*$ eine Sprache über dem Alphabet Σ, dann ist $\mathcal{L}$ abzählbar.

Definition 5.5 ((rekursiv) aufzählbar/semi-entscheidbar)**.** Eine Menge M heißt (rekursiv) aufzählbar/semi-entscheidbar, falls es eine rekursive und surjektive Funktion $f : \mathbb{N} \to M$ gibt oder $M = \emptyset$. (rekursiv $\hat{=}$ berechenbar/programmierbar, vgl. Definition 1.17)

Mit anderen Worten: Es gibt einen (rekursiven) Algorithmus, der $f(n)$ für jedes $n \in \mathbb{N}$ berechnet.

Da jede Sprache eine Menge ist, ist analog zu obigem auch eine *(rekursiv) aufzählbare Sprache* definiert.

Satz 5.6. Sei Σ ein Alphabet, dann ist Σ^* aufzählbar.

Beweis. Wir definieren eine Funktion, die Schrittweise alle Elemente von Σ^* erzeugt.

Sei $\Sigma = \{a_1, \ldots, a_n\}$ mit $a_1 < \cdots < a_n$, dann soll gelten:

$$\begin{array}{ll} f(0) = \varepsilon & \\ f(1) = a_1 & \\ \quad \vdots & \forall i \in \mathbb{N}: \\ & f(n+i+1) \text{ ist das in lexikografischer Ordnung auf } f(n+i) \\ f(n) = a_n & \text{folgende Element} \end{array}$$

□

Beispiel. $\Sigma = \{a, b, c\}$ mit $a < b < c$:

$$\begin{array}{llll} f(0) = \varepsilon & f(4) = aa & f(8) = bb & f(12) = cc \\ f(1) = a & f(5) = ab & f(9) = bc & f(13) = aaa \\ f(2) = b & f(6) = ac & f(10) = ca & f(14) = aab \\ f(3) = c & f(7) = ba & f(11) = cb & \ldots \end{array}$$

Anmerkung. Nicht jede $\mathcal{L} \subset \Sigma^*$ ist aufzählbar.

Definition 5.7 (entscheidbar)**.** Eine Sprache $\mathcal{L} \subseteq \Sigma^*$ über dem Alphabet Σ heißt entscheidbar, wenn es eine rekursive charakteristische Funktion $\chi : \Sigma^* \to \{true, false\}$ mit

$$\chi(w) = \begin{cases} true & \text{falls } w \in \mathcal{L} \\ false & \text{falls } w \notin \mathcal{L} \end{cases}$$

gibt.

Anmerkung. Es ist leicht zu sehen, dass jede endliche Sprache entscheidbar ist.

Satz 5.8. Jede entscheidbare Sprache ist aufzählbar.

Beweis. Sei f die Funktion, die Σ^* aufzählt (vgl. Satz 5.6). So gilt für jede entscheidbare Sprache $\mathcal{L} \subseteq \Sigma^*$:
Es gibt eine Funktion $g : \mathbb{N} \to \mathcal{L}$, die alle Elemente von f der Reihe nach durchgeht und mit χ überprüft, ob das Element in $\mathcal{L}$ enthalten ist. Elemente, die nicht enthalten sind, werden ausgelassen. So können mithilfe von χ alle Elemente der Sprache aufgezählt werden. □

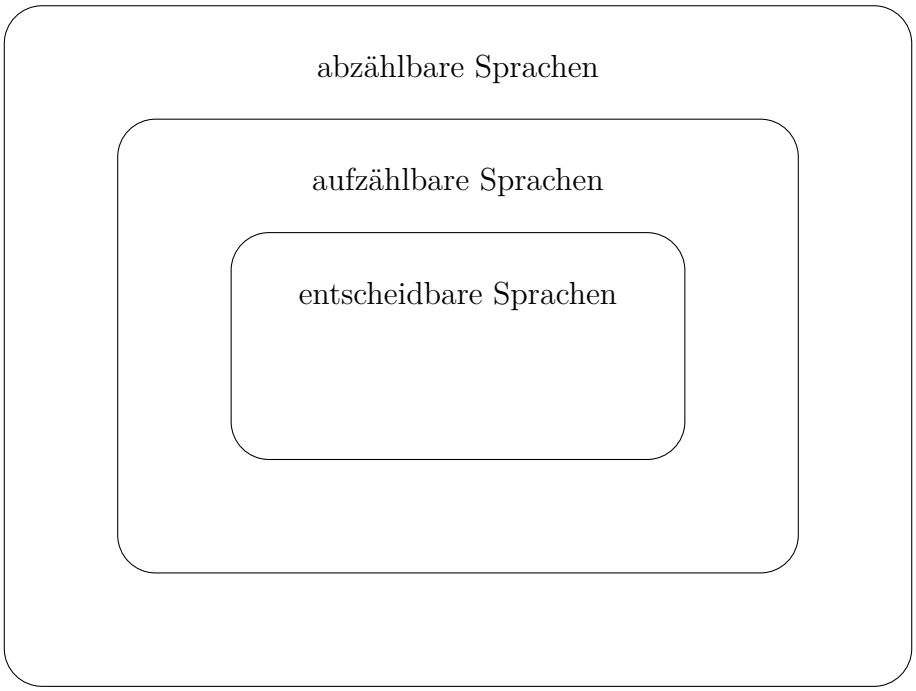


Abbildung 5.1: Zusammenhang zwischen entscheidbar, aufzählbar und abzählbar

Im Folgenden werden wir nun die Chomsky-Hierarchie ansehen und untersuchen, welche Sprachklassen welche Eigenschaften haben.

Satz 5.9. Eine Sprache ist genau dann von einer Grammatik erzeugbar, wenn sie rekursiv aufzählbar ist, d.h. alle von einer Grammatik erzeugbaren Sprachen können von einer Turingmaschine aufgezählt werden.

Beweis. Für den Beweis dieses Satzes benötigen wir die *Effektive Nummerierung* von $\mathbb{N}^2$ (s. Abb. 5.2).

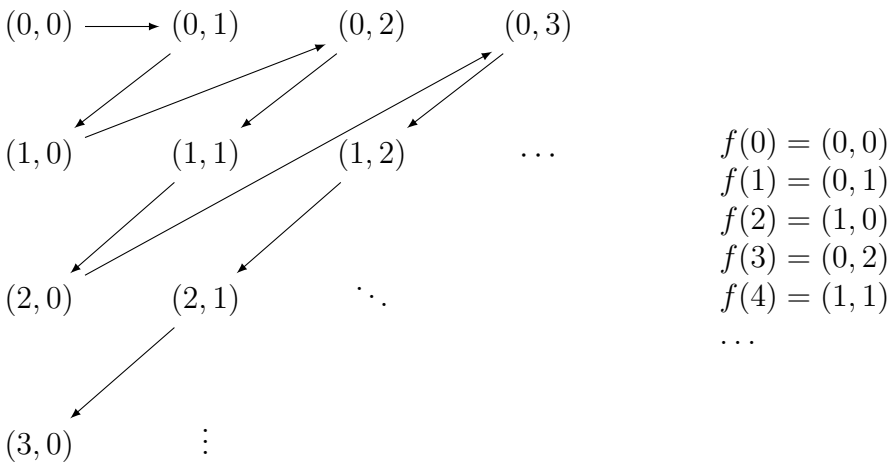


Abbildung 5.2: Effektive Nummerierung von $\mathbb{N}^2$

Man kann die Funktion f aufteilen in $f(n) = (l(n), r(n))$. Alle drei Funktionen f, l, r sind rekursiv. In diesem Beweis wird die effektive Nummerierung von $\mathbb{N}^2$ benötigt für:

$l(n)$: Auflistung aller Wörter die in $l(n)$ Schritten ableitbar sind.

$r(n)$: Das $r(n)$. Wort in der Liste von $l(n)$.

Sei $G = (N, \Sigma, \Pi, S)$ eine beliebige Grammatik. Wir konstruieren nun eine Turingmaschine, die eine total rekursive surjektive Funktion $\Phi : \mathbb{N} \to \mathcal{L}(G)$ berechnet, die $\mathcal{L}(G)$ aufzählt.

Sei $w \in \mathcal{L}(G)$ fest gewählt:

1. Berechne für die Eingabe n die Werte $l(n)$ und $r(n)$
2. Führe $l(n)$ Schritte durch $(0 \leq i \leq l(n))$:
   0. Schritt: Schreibe S auf das Band
   $i+1$. Schritt: Zu allen in Schritt i auf das Band geschriebenen Wörter schreibe sämtliche direkten Ableitungen auf das Band. Da Π endlich ist, sind dies endlich viele Neubildungen. Lösche alle im Schritt i geschriebenen Wörter.

3. Sind mindestens $r(n)+1$ Wörter entstanden, lösche alle Wörter bis auf das $(r(n)+1)$. Wort. Andernfalls gehe zu 5.

4. Besteht das Wort u auf dem Band nur aus Terminalen ($u \in \Sigma^*$ und $u \in \mathcal{L}(G)$, da aus G abgeleitet), so ist u das Ergebnis und die Turingmaschine hält an. Andernfalls gehe zu 5.

5. Schreibe w auf das Band und die Turingmaschine hält an

□

Bevor wir die Rückrichtung betrachten, geben wir ein Beispiel an:

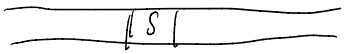
Beispiel. Als Beispiel für eine Turingmaschine, die aus einer Grammatik vom Typ-0 erzeugt wird, betrachten wir folgende Grammatik:

$G = (\{S, X, Y\}, \{0, 1\}, \Pi, S)$
$\Pi = \{$
$\quad S \to XY,$
$\quad X \to 1X \mid Y,$
$\quad Y \to 0Y \mid 0$
$\}$

Anmerkung. $\mathcal{L}(G) = \mathcal{L}(1^*000^*)$ ist eine Sprache vom Typ-3

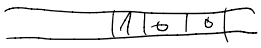
$w = 100$

$n > 0$: 1. $\ell(0) = 0$, $r(0) = 0$

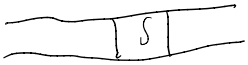
2. 

3. nichts zu löschen

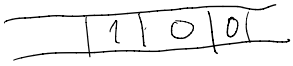
4. $S \notin \Sigma^*$

5. 

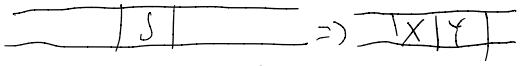
$n = 1$ 1. $\ell(1) = 0$, $r(1) = 1$

2. 

3. Es sind keine $r(1) + 1 = 2$ Worte entstanden

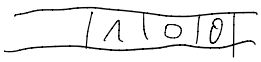
5. 

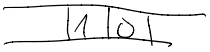
$n = 2$ 1. $\ell(2) = 1$, $r(2) = 0$

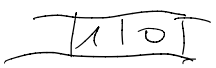
2. 

3. nichts zu löschen

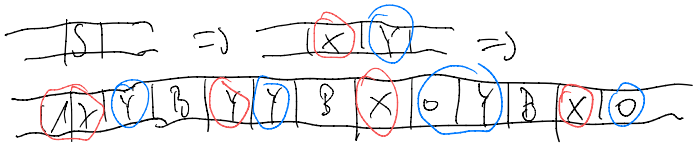
4. $XY \notin \Sigma^*$

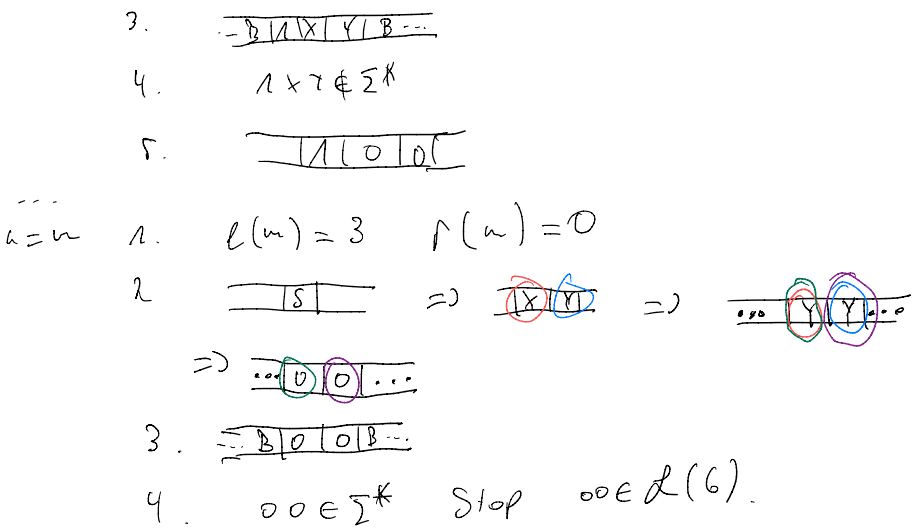
5. 

$n = 3$ 1. $\ell(3) = 0$, $r(3) = 2$ $\Rightarrow$ 

$n = 4$ 1. $\ell(4) = 1$, $r(4) = 1$ $\Rightarrow$ 

$n = 5$ 1. $\ell(5) = 2$, $r(5) = 0$

2. 



Rückrichtung:

Eine Menge $M \subseteq \Sigma^*$ wird durch eine Turingmaschine $TM = (Q, \Sigma_{TM}, \Gamma, \delta, q_0, B, \{q_E\})$ aufgezählt, d.h. TM berechnet eine surjektive Funktion $\Phi : I \to M$, wobei $I \subseteq \Sigma_{TM}^*$ mit $\Sigma_{TM} \subseteq \{0, \ldots, 9\}$. Weiterhin gilt: $\Gamma = \Sigma \cup \{B\} \cup \Sigma_{TM}$, $\Gamma \cap Q = \emptyset$ und nach der Berechnung befindet sich der LSK der TM immer auf dem ersten Symbol des Ergebnisses.

Im Folgenden wird die Grammatik G mit $\mathcal{L}(G) = M$ erzeugt:

$$G = (Q \dot\cup (\Gamma \setminus \Sigma) \dot\cup \{S, L, P, A\}, \Sigma, \Pi, S)$$

mit folgenden Produktionen:

(1) $S \to LAP$,
$A \to q_0$,
$A \to Ax, \qquad \forall x \in \Sigma_{TM}$

(Damit wird die Anfangskonfiguration Lq_0pP mit $p \in \Sigma_{TM}^*$ der TM erzeugt)

(2) Sei $Q' = Q \setminus \{q_E\}$, dann $\forall q \in Q'$:
$qx \to q'y$ für $\delta(q, x) = (q', y, N)$
$qx \to yq'$ für $\delta(q, x) = (q', y, R)$
$x'qx \to q'x'y$ $\forall x' \in \Gamma$ und $\delta(q, x) = (q', y, L)$

(Der LSK wird durch den Zustand q bzw. q' links von dem Zeichen repräsentiert.)

(3) $qP \to qBP \qquad \forall q \in Q'$

$Lq \to LBq \qquad \forall q \in Q'$

(Einfügen von Blanks am rechten und linken Rand; wie beim Band der TM)

(4) $Bq_E \to q_E$

$Lq_E \to q_E$

$q_E x \to x q_E \qquad \forall x \in \Sigma$

$q_E B \to q_E$

$q_E P \to \varepsilon$

(Zeichenketten der Form $LB \dots Bq_E wB \dots BP$ mit $w \in M$ werden auf w reduziert $\to B, L, P, q_E$ löschen)

Beispiel. Sei folgende Turingmaschine gegeben:

$TM = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{1\}, \{a, 1, B\}, \delta, q_0, B, \{q_5\})$

$\delta = \{$

$(q_0, 1) \mapsto (q_1, a, L)$

$(q_1, B) \mapsto (q_2, a, R)$

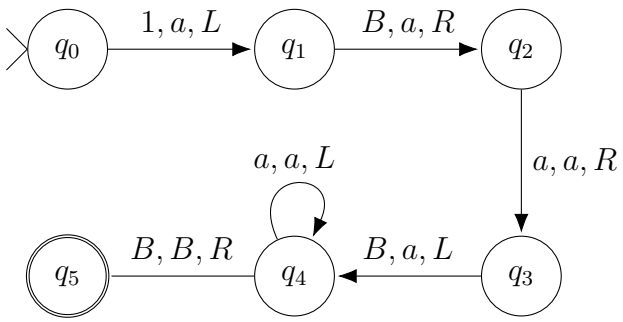
$(q_2, a) \mapsto (q_3, a, R)$

$(q_3, B) \mapsto (q_4, a, L)$

$(q_4, a) \mapsto (q_4, a, L)$

$(q_4, B) \mapsto (q_5, B, R)$

$\}$



Die Turingmaschine zählt nur das Wort aaa auf.
Umwandlung in eine Grammatik:

$G = (\{q_0, q_1, q_2, q_3, q_4, q_5\} \cup \{1, B\} \cup \{S, L, P, A\}, \{\, a \,\}, \Pi, S)$

$\Pi = \{$

$S \to LAP$ (1)
$A \to q_0$ (2)
$A \to A1$ (3)
} (1)

$q_1 B \to a q_2$ (4)
$q_2 a \to a q_3$ (5)
$q_4 B \to B q_5$ (6)
} R

$a q_0 1 \to q_1 a a$ (7)
$1 q_0 1 \to q_1 1 a$ (8)
$B q_0 1 \to q_1 B a$ (9)
$a q_3 B \to q_4 a a$ (10)
$1 q_3 B \to q_4 1 a$ (11)
$B q_3 B \to q_4 B a$ (12)
$a q_4 a \to q_4 a a$ (13)
$1 q_4 a \to q_4 1 a$ (14)
$B q_4 a \to q_4 B a$ (15)
} L } (2)

$q_i P \to q_i B P \quad i \in \{0 \dots 4\}$ (16)
$L q_i \to L B q_i \quad i \in \{0 \dots 4\}$ (17)
} (3)

$B q_5 \to q_5$ (18)
$L q_5 \to q_5$ (19)
$q_5 a \to a q_5$ (20)
$q_5 B \to q_5$ (21)
$q_5 P \to \varepsilon$ (22)
} (4)

$\}$

Ableitung von S:

$$S \xrightarrow{(1)} LAP \xrightarrow{(3)} LA1P \xrightarrow{(2)} Lq_0 1P \xrightarrow{(17)} LBq_0 1P \xrightarrow{(9)} Lq_1 BaP \xrightarrow{(4)} Laq_2 aP \xrightarrow{(5)}$$
$$Laaq_3 P \xrightarrow{(16)} Laaq_3 BP \xrightarrow{(10)} Laq_4 aaP \xrightarrow{(13)} Lq_4 aaaP \xrightarrow{(17)} LBq_4 aaaP \xrightarrow{(15)}$$
$$Lq_4 BaaaP \xrightarrow{(6)} LBq_5 aaaP \xrightarrow{(18)} Lq_5 aaaP \xrightarrow{(19)} q_5 aaaP \xrightarrow{(20)^3} aaaq_5 P \xrightarrow{(22)} aaa$$

Satz 5.10 (Zusammenhang zu Chomsky–0 Grammatiken)**.** Für jede rekursive Funktion f existiert eine Chomsky–0 Grammatik, deren Ableitungen der Berechnung der Funktion entspricht.

Beweis. Nach dem Hauptsatz dem Algorithmentheorie (Satz 1.25) gibt es eine Turingmaschine

$$TM = (Q, \{\,0, 1\,\}, \{\,0, 1, \#\,\}, \delta, q_0, \#, F)$$

die f berechnet. Sei weiter

$$TM' = (Q', \{\,0, 1\,\}, \{\,0', 1', 0, 1, \#\,\}, \delta', q_0', \#, F')$$

eine erweiterte Turingmaschine, die TM entspricht und am Ende alle Zeichen 0 auf dem Band durch $0'$ und alle Zeichen 1 durch $1'$ ersetzt, und den LSK auf das Zeichen ganz links verschiebt. Es ist offensichtlich, dass TM und TM' äquivalente Funktionen berechnen.

Im Beweis von Satz 5.9 wird aus einer gegebenen Turingmaschine, die eine Sprache aufzählt, eine Chomsky–0 Grammatik

$$G = (Q' \dot\cup \{\,0, 1, \#\,\} \dot\cup \{S, L, P, A\}, \{\,0', 1'\,\}, \Pi, S)$$

der Sprache angegeben. Die Ableitungen dieser Grammatik entsprechen also der Berechnungen der Funktion f. □

Satz 5.11 (Satz von Chomsky)**.** Jede Sprache einer nicht-verkürzenden Grammatik (Typ 1b) ist rekursiv-entscheidbar.

Beweis. Sei $\mathcal{L} \subseteq \Sigma^*$ eine Sprache einer nicht-verkürzenden Grammatik $G = (N, \Sigma, \Pi, S)$. Wir geben eine Turingmaschine an, die $\mathcal{L}$ entscheidet:

1. w steht als Eingabe auf dem Band
2. Länge von w wird bestimmt
3. Alle Ableitungen aus S bis zu einer Länge von $|w| + 1$ werden erzeugt (terminiert, weil G nicht-verkürzend ist)
4. a) Wenn w im 3. Schritt erzeugt wurde, gibt die Turingmaschine 1 ($true$) aus

   b) Wenn w im 3. Schritt nicht erzeugt wurde, dann wird w auch durch weitere Ableitungen nicht mehr erzeugt (weil G nicht-verkürzend ist) und die Turingmaschine gibt 0 ($false$) aus

□

Satz 5.12. Nicht jede rekursiv-entscheidbare Sprache ist vom Typ 1b.

Beweis. Sei $G_0, G_1, \ldots$ eine effektive Nummerierung aller nicht-verkürzender Grammatiken (surjektive rekursive Abbildung: $\mathbb{N} \to \{G \mid G \text{ ist Grammatik vom Typ 1b}\}$) über einem Alphabet Σ. Dann sei $\mathcal{L}_0, \mathcal{L}_1, \ldots$ eine effektive Nummerierung aller entsprechenden Sprachen vom Typ 1b. Sei nun:

$$\mathcal{L} = \{w \mid w \in \Sigma^* \wedge w \notin \mathcal{L}_{|w|}\}$$

Wir zeigen nun, dass $\mathcal{L}$ zwar rekursiv-entscheidbar, aber nicht vom Typ 1b ist.

1. $\mathcal{L}$ ist rekursiv-entscheidbar:
   Aus $|w|$ folgt, dass $\mathcal{L}_{|w|}$ bestimmbar ist, da $\mathcal{L}_i$ effektiv nummeriert ist. $\mathcal{L}_{|w|}$ ist eine Sprache vom Typ 1b, somit ist nach Satz 5.11 $\mathcal{L}_{|w|}$ rekursiv entscheidbar. Wir können also entscheiden, ob $w \in \mathcal{L}_{|w|}$ liegt, somit ist auch $\mathcal{L}$ rekursiv-entscheidbar, weil auch $w \notin \mathcal{L}_{|w|}$ entscheidbar ist.

2. $\mathcal{L}$ ist nicht vom Typ 1b:
   Wir zeigen, dass es kein i gibt mit $\mathcal{L} = \mathcal{L}_i$:
   Angenommen es gäbe ein i mit $\mathcal{L} = \mathcal{L}_i$, dann existiert ein w mit $|w| = i$. Nun gibt es 2 Möglichkeiten:

   a) $w \in \mathcal{L}$: Dann gilt nach Definition von $\mathcal{L}$: $w \notin \mathcal{L}_i$

   b) $w \notin \mathcal{L}$: Dann gilt nach Definition von $\mathcal{L}$: $w \in \mathcal{L}_i$

   Beide Fälle sind ein Widerspruch zur Annahme $\mathcal{L} = \mathcal{L}_i$

□

Satz 5.13. Die Menge der Sprachen vom Typ 1a und Typ 1b sind identisch.

Beweis. siehe z.B. [Winter, Satz 4.7] □

Daraus ergibt sich, dass die Sätze 5.11 und 5.12 analog auch für Sprachen kontextsensitiver Grammatiken (Typ 1a) gültig sind.

Satz 5.14. Jede Sprache einer kontextsensitiven Grammatik ist rekursiv-entscheidbar.

Korollar 5.15. Nicht jede rekursiv-entscheidbare Sprache ist Sprache einer kontextsensitiven Grammatik (Typ 1a). (analog zu Satz 5.12)

Definition 5.16 (linear-beschränkte Turingmaschine (LBTM))**.** Eine Turingmaschine heißt linear beschränkt, wenn sie nur den Speicherplatz benötigt, den die Eingabe belegt hat.

Beispiel. $\mathcal{L} = \{0^n 1^n 2^n\}$

Wir geben nun eine Turingmaschine an, die $\mathcal{L}$ erkennt. Idee:

1. Lösche die erste 0
2. Überschreibe die letzte 1 mit 2
3. Lösche die letzten beiden 2er

Anmerkung. Bisher haben wir nur nicht zwischen deterministischen und nichtdeterministischen Turingmaschinen unterschieden, weil die Menge der berechenbaren Funktionen sich für allgemeine Turingmaschinen nicht unterscheiden. Bei LBTMs ist dies bisher nicht bekannt. Deshalb kann folgender Satz nur für nicht-deterministische LBTMs bewiesen werden.

Satz 5.17 (Kuroda, Weber 1964)**.** Eine Sprache ist genau dann kontextsensitiv, wenn sie von einer nicht-deterministischen linear-beschränkten Turingmaschine akzeptiert wird.

Beweis. Sei $G = (N, \Sigma, \Pi, S)$ eine nicht-verkürzende Grammatik. Wir konstruieren nun eine nicht-deterministische LBTM:

$$T = (\{q_0, \ldots, q_E\}, \Sigma, \Gamma = \Sigma \cup N \cup \{B, *\}, \delta, q_0, B, \{q_E\})$$

Die Konstruktion der Relation δ wird durch die folgenden Schritte beschrieben:

1. Steht $w \in (\Sigma \cup N)^*$ auf dem Band, dann gehe zu 2., sonst zu 6.
2. Steht für $(p \to q) \in \Pi$ wqw' auf dem Band, gehe zu 3., sonst zu 5. (nicht-deterministisch bei der Bestimmung der Produktion)
3. q wird durch $p*^{|q|-|p|}$ ersetzt (z.B. $q \to p * **$, wenn $|q| - |p| = 3$)
4. $*$ werden gelöscht und die anderen Bandsymbole zusammengeschoben. Neues Wort auf dem Band wird mit w bezeichnet. Gehe zu 2.
5. Ist $w = S$, ersetze S durch 1 ($true$), gehe zu q_E und stoppe.
6. Das Band von T wird gelöscht und mit 0 ($false$) ersetzt. Gehe zu q_E und stoppe.

T ist beschränkt, da G nicht-verkürzend ist, d.h für $(p \to q) \in \Pi$ gilt $|q| - |p|$. Nach Konstruktion akzeptiert T die Sprache $\mathcal{L}(G)$.

Rückrichtung:

Prinzipiell wird ähnlich vorgegangen wie im Beweis von Satz 5.9, allerdings ist bei der Konstruktion der Grammatik zu beachten, dass sie nicht-verkürzend wird.

Für den konkreten Beweis wird auf [Winter, Satz 4.13] verwiesen. □

Beispiel. Sei $G = (\{S, B\}, \{0, 1, 2\}, \Pi, S)$ mit

$$\Pi = \{$$
$$\quad S \to 0SB2 \mid 012,$$
$$\quad 2B \to B2,$$
$$\quad 1B \to 11$$
$$\}$$

gegeben.

Ablauf der LBTM mit der Eingabe 001122:

001122	
$001B22$	
$0012B2$	$0012B2$
$00122B$	$0S^{**}B2$
$0S^{**}2B$	$0SB2$
$0S2B$ ↯	S^{***}
	S ✓

$\mathcal{L} = \{0^n 1^n 2^n\}$

Zusammenhang der Sprachtypen und der Chomsky-Hierarchie mit den Automaten

Chomsky-Sprachtyp	Beschreibung		Automat	Algorithmus	Tool
Typ-0	allgemein	$\overset{5.9}{\Leftrightarrow}$	TM (nur aufzählbar, nicht entscheidbar)		
	$\cup$ (5.12, 5.15)				
	entscheidbare Sprachen	$\overset{\text{per Definition}}{\Leftrightarrow}$	TM		
Typ-1	nicht-verkürzend		NDLBTM		
	$\Updownarrow$				
	kontextsensitiv	$\overset{5.13/5.17}{\Leftrightarrow}$	NDLBTM		
Typ-2	kontextfrei	$\overset{4.7}{\Leftrightarrow}$	NPDA		
			$\cup$		
			DPDA	LR(k) mit k $\geq$1	
			$\Leftrightarrow$		
				$\cup$	
				LALR	JAY/JAOOY
				$\cup$	
				SLR	
Typ-3	regulär	$\overset{3.13}{\Leftrightarrow}$	DEA $\overset{3.13}{\Leftrightarrow}$ NEA		JLEX