

Grundlagen der Theoretischen Informatik

Wintersemester 2024/25

Prof. Dr. Heribert Vollmer

Institut für Theoretische Informatik
Leibniz Universität Hannover

Link für Kurzfragen:



URL: <https://pingo.coactum.de/events/739935>

Inhalt

Sprachen und Grammatiken

Die Chomsky-Hierarchie

Reguläre (Typ-3-) Sprachen

Endliche Automaten

Nichtdeterministische endliche
Automaten

Endliche Automaten und
Typ-3-Grammatiken

Das Pumping Lemma für
reguläre Sprachen

Kontextfreie (Typ-2-) Sprachen

Kellerautomaten

Das Pumping-Lemma für
kontextfreie Sprachen

Typ-1- und Typ-0-Sprachen

Der intuitive Berechenbarkeitsbegriff

Berechenbarkeit durch Maschinen

Turing-Berechenbarkeit

Mehrband-Maschinen

Berechenbarkeit in
Programmiersprachen

Die Programmiersprache LOOP

Die Programmiersprache WHILE

Die Church'sche These

Entscheidbarkeit und Aufzählbarkeit

Unentscheidbare Probleme

Das Halteproblem

Der Satz von Rice

Der intuitive Berechenbarkeitsbegriff

Berechenbarkeit

Eine Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **berechenbar**, falls es einen Algorithmus gibt, der f berechnet, d. h. gestartet mit Eingabe $(n_1, \dots, n_k) \in \mathbb{N}^k$ hält der Algorithmus nach endlich vielen Schritten mit Ausgabe $f(n_1, \dots, n_k)$.

Wir fordern nicht, dass f total sein muss, d. h. für gewisse $(n_1, \dots, n_k) \in \mathbb{N}^k$ darf $f(n_1, \dots, n_k)$ undefiniert sein. In diesem Fall soll der Algorithmus nicht stoppen (Endlosschleife).

Ziel: Präzisierung des Berechenbarkeitsbegriffs, d.h. des Begriffs **Algorithmus**.

Nur so ist es möglich, zu beweisen, dass eine Funktion **nicht** berechenbar ist.

Beispiel 1

$f_1: \mathbb{N} \rightarrow \mathbb{N}$

$$f_1(n) = \begin{cases} 1, & \text{falls } n \text{ ein Anfangsabschnitt der} \\ & \text{Nachkommastellen von } \pi \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

$$\pi = 3,14159265\ldots$$

$$f_1(1) =$$

$$f_1(2) = 0 = f_1(3)$$

$$f_1(14) =$$

$$= f_1(14159267)$$

$$f_1(141) = 1$$

Näherungsverfahren zur Berechnung von π

Ahoi: f_1 ist berechenbar

Beispiel 2

3, 101001000100001 ...

$$f_2(n) = \begin{cases} 1, & \text{falls } n \text{ irgendwo in den} \\ & \text{Nachkommastellen von } \pi \text{ vorkommt} \\ 0, & \text{sonst} \\ 1 & (\text{undef}) \end{cases}$$

$\pi = 3, 14159265 \dots$

$$f_2(2) = 1 \qquad f_2(2) = 0$$

$$\rightarrow f_2(265)$$

$$f_2(999) = 9$$

Berechenbarkeit von π
ist unbekannt.

f_2' ist berechenbar!

Beispiel 3

$$f_3(n) = \begin{cases} 1, & \text{falls 7 in den Nachkommastellen von } \pi \text{ irgendwo mindestens } n\text{-mal hintereinander vorkommt} \\ 0, & \text{sonst} \end{cases}$$

$f_3(100000)$ ist s.d.m. $\underbrace{777 \dots 7}_{100000 \text{ Stich}}$ in π vorkommt.

f_3 ist beachbar.

Pall 1:

Es kommen bei Blöcken

$77\dots 7$ in π vor.

Aber $f_3(w) = 1$ f.a. w.n.w.,
also beachbar.

Pall 2:

f_3 gibt maximale Blocklänge m , also

$$f_3(w) = \begin{cases} 1, & \text{falls } w \leq m \\ 0, & \text{sonst} \end{cases}$$

Aber f_3 ist f_3 beachbar.

Beispiel 4

$$f_4(n) = \begin{cases} 1, & \text{falls die Antwort auf das LBA-Problem „ja“ ist} \\ 0, & \text{sonst} \end{cases}$$

$f_4(u) = 1$ f.a. nein oder $f_4(u) = 0$ f.a. nein.

Turing-Berechenbarkeit

Definition

Eine Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, falls es eine DTM M gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt:

$$f(n_1, \dots, n_k) = m \Rightarrow$$

M mit Eingabe $\text{bin}(n_1)\#\dots\#\text{bin}(n_k)$

hält mit $\square \dots \square \text{bin}(m) \square \dots \square$

auf dem Arbeitsband.

Kopf von M auf
ersten Zeichen von
 $\text{bin}(m)$

stoppt auf einem
Zeichen von
 $\text{bin}(m)$

$$f(n_1, \dots, n_k) \text{ undefined} \Rightarrow$$

M mit Eingabe $\text{bin}(n_1)\#\text{bin}(n_2)\#\dots\#\text{bin}(n_k)$

stoppt nicht.

$\text{bin}(n)$ für $n \in \mathbb{N}$ bezeichnet die Binärdarstellung von n ohne
führende Nullen.

Bemerkung

Das Eingabealphabet einer TM, die eine Funktion über \mathbb{N} im obigen Sinne berechnet, ist stets $\{0, 1, \#\}$.

Beispiele:

1. $s: \mathbb{N} \rightarrow \mathbb{N}, s(n) = n+1$

ist Turing-berechenbar.

$$f(1) = 10$$

2. $f: \mathbb{N} \rightarrow \mathbb{N}, f(n) = 2n$

$$f(11) = 11$$

Polyade Tm berechnet f:

$$z_0 a \rightarrow z_0 a R \quad \text{für } a \in \{0, 1\}$$

$$z_0 \square \rightarrow z_1 0 L$$

$$z_1 a \rightarrow z_1 a L \quad - / \! \! \! -$$

$$z_1 \square \rightarrow z_0 \square R$$

Definition

Eine Funktion $f: \Sigma^* \rightarrow \Delta^*$ heißt **Turing-berechenbar**, falls es DTM M gibt, sodass für alle $x \in \Sigma^*$ und $y \in \Delta^*$ gilt:

$$f(x) = y \Rightarrow$$

M mit Eingabe x

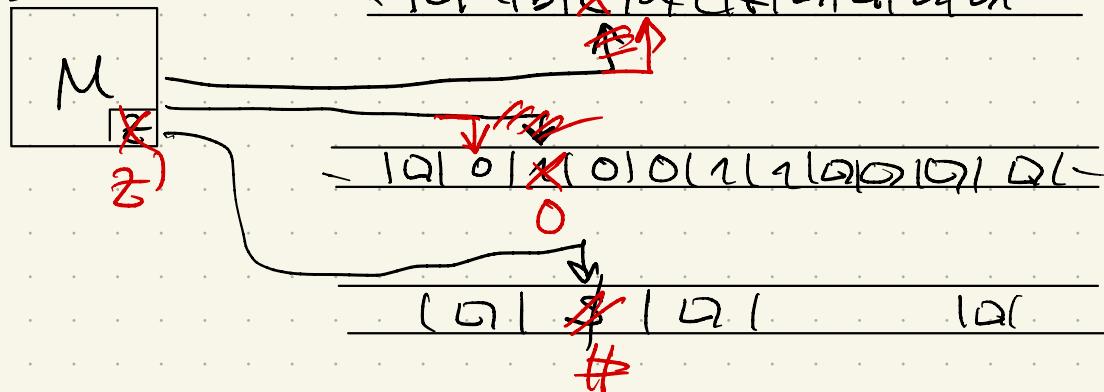
hält mit $\square \dots \square y \square \dots \square$ auf dem Arbeitsband.

$$f(x) \text{ undefiniert} \Rightarrow$$

M mit Eingabe x stoppt nicht.

Mehrband-Maschinen

k -Band-TM
 $(k=3)$



$$\mathcal{S}(z, c, r, \$) = (z, b, \emptyset, \#, R, \sqcup, \times)$$

$$2 \times T^k \quad 8 \times T^k \times \{L, N, R\}^k$$

Definition

Eine k -Band-DTM ist ein 7-Tupel

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E),$$

wobei für die einzelnen Komponenten gilt:

- $Z, \Sigma, \Gamma, z_0, \square$ und E sind wie bei einer 1-Band-DTM definiert.
- $\delta: \underbrace{Z}_{(i)} \times \underbrace{\Gamma^k}_{(ii)} \rightarrow \underbrace{Z}_{(iii)} \times \underbrace{\Gamma^k}_{(iv)} \times \underbrace{\{L, R, N\}^k}_{(v)}$ mit
 - (i) aktueller Zustand
 - (ii) gelesene Zeichen auf den k Bändern
 - (iii) neuer Zustand
 - (iv) geschriebene Zeichen auf den k Bändern
 - (v) Kopfbewegungen auf den k Bändern

Beispiel: $L = \{w\#w \mid w \in \{0,1\}^*\}$

2-Band TM under L accepts!

M_1 :

-- 10|1|0|0| # | 1|0|0|0| -
↑
01|1|0|0|0| -
↑

M_1' :

- 10|1|0|0| # | 0|1|0|0| ~~0|1|0|0|~~ | 0|1|0|0| -

Arbeitsweise

Die Eingabe steht zunächst auf Band 1. Die Bänder 2 bis k sind zunächst leer.

Die Maschine führt einzelne Schritte durch, analog zu gewöhnlichen DTMn.

Akzeptierte Sprache: Das Eingabewort x wird akzeptiert gdw. M erreicht irgendwann einen Endzustand.

Berechnete Funktion: $f(n_1, \dots, n_k) = m$ gdw. M mit Eingabe $\text{bin}(n_1)\# \dots \#\text{bin}(n_k)$ erreicht irgendwann einen Endzustand mit $\text{bin}(m)$ auf Band 1.

(Berechnung von Funktionen $f: \Sigma^* \rightarrow \Delta^*$ analog.)

Beispiel

Folgende 2-Band-Turingmaschine akzeptiert $\{w\#w \mid w \in \{0, 1\}^*\}$:

$$M = (\{z_0, z_1, z_2, z_e\}, \{0, 1, \#\}, \{0, 1, \#, \square\}, \delta, z_0, \square, \{z_e\}),$$

wobei für die Überführungsfunktion gilt:

$$\begin{array}{l l l} z_0 0 \square & \rightarrow & z_0 00 RR \\ z_0 1 \square & \rightarrow & z_0 11 RR \end{array} \quad \left. \begin{array}{l} 0, 1 \text{ auf Band 1 werden auf Band} \\ 2 \text{ kopiert} \end{array} \right\}$$

$$z_0 \# \square \rightarrow z_1 \# \square NL \quad \left. \begin{array}{l} \# \text{ auf Band 1} \Rightarrow \text{Zustand } z_1 \end{array} \right\}$$

$$z_0 \square \square \rightarrow z_0 \square \square NN \quad \left. \begin{array}{l} \text{Endlosschleife, falls kein } \# \text{ gefunden wird} \end{array} \right\}$$

$$\begin{array}{l l l} z_1 \# 0 & \rightarrow & z_1 \# 0 NL \\ z_1 \# 1 & \rightarrow & z_1 \# 1 NL \end{array} \quad \left. \begin{array}{l} \text{Kopf auf Band 2 nach links} \\ \text{Kopf auf Band 1 bleibt auf } \# \end{array} \right\}$$

Beispiel (Fortsetzung)

| | | |
|--|---|--|
| $z_1 \# \square \rightarrow z_2 \# \square RR$ | $\left. \begin{array}{l} \\ \end{array} \right\}$ | \square auf Band 2 \Rightarrow Zustand z_2 |
| $z_2 00 \rightarrow z_2 00 RR$ | $\left. \begin{array}{l} \\ \end{array} \right\}$ | auf beiden Bändern nach rechts gehen, |
| $z_2 11 \rightarrow z_2 11 RR$ | $\left. \begin{array}{l} \\ \end{array} \right\}$ | solange gleiche Zeichen gefunden werden |
| $z_2 01 \rightarrow z_2 01 NN$ | $\left. \begin{array}{l} \\ \end{array} \right\}$ | verschiedene Zeichen \Rightarrow Endlos- |
| $z_2 10 \rightarrow z_2 10 NN$ | $\left. \begin{array}{l} \\ \end{array} \right\}$ | schleife |
| $z_2 \square \square \rightarrow z_e \square \square NN$ | $\left. \begin{array}{l} \\ \end{array} \right\}$ | Alles gleich, daher fertig |

Beispiel (Fortsetzung)

$$\left. \begin{array}{l} z_2 0 \square \rightarrow z_2 0 \square NN \\ z_2 1 \square \rightarrow z_2 1 \square NN \\ z_2 \square 0 \rightarrow z_2 \square 0 NN \\ z_2 \square 1 \rightarrow z_2 \square 1 NN \end{array} \right\} \text{unterschiedliche Länge} \Rightarrow \text{Endlosschleife}$$

$$\left. \begin{array}{l} z_2 \# 0 \rightarrow z_2 \# 0 NN \\ z_2 \# 1 \rightarrow z_2 \# 1 NN \\ z_2 \# \square \rightarrow z_2 \# \square NN \end{array} \right\} \text{Endlosschleife, falls zweites \# gefunden wird}$$

Satz

Sei $k > 1$. Zu jeder **k -Band**-DTM M gibt es eine (**1-Band**-)DTM M' , sodass $L(M) = L(M')$ bzw. dass M und M' dieselbe **Funktion** berechnen.

Beweisidee:

Sei $M = (Q, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine k -Band-Maschine.

Wir speichern auf dem Band von M' hintereinander die Inhalte der k Bänder von M , getrennt durch ein spezielles Trennsymbol.
Wir markieren die Positionen der k Köpfe von M .

Simulation eines Schrittes von M : Aktualisierung der gelesenen Zeichen sowie der Kopfpositionen an k Stellen auf dem Band von M' .

Falls notwendig: Bereich für Bänder von M auf dem Band von M' vergrößern.

1-Band nach k-Band

Sei M eine 1-Band-TM. Dann bezeichnet $M(i, k)$ ($1 \leq i \leq k$), die k -Band-TM, die auf Band i genau die Aktion ausführt, die M auf seinem Band ausführt, und die Bänder $1, \dots, i-1, i+1, \dots, k$ unverändert lässt. Ist also z. B. in M

$\delta(z, a) = (z', b, X)$ mit $X \in \{L, N, R\}$, so ergibt sich für $M(2, 4)$:

$$\delta(z, c_1, a, c_3, c_4) = (z', c_1, b, c_3, c_4, N, X, N, N)$$

für alle c_1, c_3 und c_4 aus dem Arbeitsalphabet von M
($=$ Arbeitsalpahbet von $M(2, 4)$).

Schreibweise: $M(i)$ statt $M(i, k)$, falls k aus dem Kontext klar.

Spezielle Maschinen

„Band := Band + 1“

„Band i := Band i + 1“

„Band i := Band i – 1“ (hier: $0 - 1 = 0$)

„Band i := 0“

„Band i := Band j“

Hintereinanderschaltung von Turingmaschinen

Seien $M_i = (Z_i, \Sigma, \Gamma_i, \delta_i, z_{0,i}, \square, E_i)$ mit $i = 1, 2$ zwei DTMn mit o. B. d. A. $Z_1 \cap Z_2 = \emptyset$.

Wir definieren daraus die neue Turingmaschine

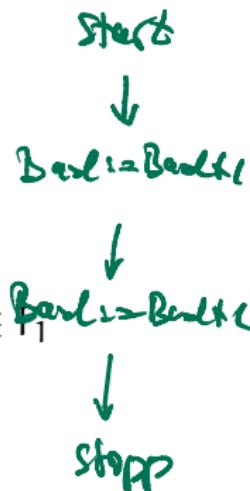
$$M = (Z_1 \cup Z_2, \Sigma, \Gamma_1 \cup \Gamma_2, \delta, z_{0,1}, \square, E_2),$$

wobei:

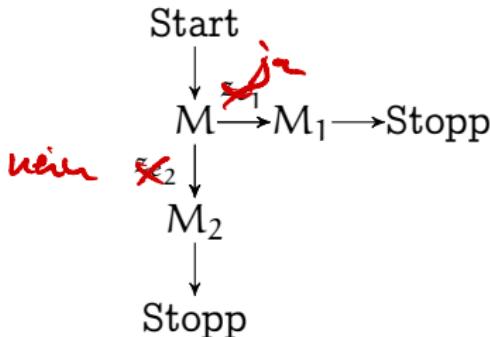
$$\delta(z, a) = \begin{cases} \delta_1(z, a), & \text{falls } z \in Z_1 \setminus E_1 \text{ und } a \in \Gamma_1 \\ \delta_2(z, a), & \text{falls } z \in Z_2 \text{ und } a \in \Gamma_2 \\ (z_{0,2}, a, N), & \text{falls } z \in E_1 \text{ und } a \in \Gamma_1 \end{cases}$$

Bezeichnungen für M : „ $M_1; M_2$ “ oder

$\text{Start} \rightarrow M_1 \rightarrow M_2 \rightarrow \text{Stopp}$. Dies lässt sich analog definieren für mehr als zwei Maschinen.



Bedingte Verzweigungen



bezeichnet die Turingmaschine, die zuerst M simuliert und vom Endzustand z_{e_1} von M nach M_1 und vom Endzustand z_{e_2} von M nach M_2 übergeht.

Bezeichnung: „IF M THEN M_1 ELSE M_2 “, falls $z_{e_1} = \text{ja}$ und $z_{e_2} = \text{nein}$.

Test auf Null

Definiere $M = (\{z_0, z_1, ja, nein\}, \Sigma, \Gamma, \delta, z_0, \square, \{ja, nein\})$ mit

- ▶ $\Sigma \supseteq \{0, 1\}$
- ▶ $\Gamma \supseteq \{0, 1, \square\}$
- ▶ für die Überführungsfunktion δ gilt:

$$\delta(z_0, a) = (nein, a, N) \text{ für } a \in \Gamma \setminus \{0\}$$

$$\delta(z_0, 0) = (z_1, 0, R)$$

$$\delta(z_1, \square) = (ja, \square, L)$$

$$\delta(z_1, a) = (nein, a, L) \text{ für } a \in \Gamma \setminus \cancel{\{0\}} \setminus \{\square\}$$

Bezeichnung für M : „Band = 0?“.

Schreibweise: „Band $i = 0?$ “ statt „Band = 0? (i)“.

Schleifen

Sei nun M eine beliebige Turingmaschine. „WHILE Band $i \neq 0$ DO M “ bezeichnet dann die Turingmaschine

