Intuitive und Turing-Berechenbarkeit

Prof. Dr. Jasmin Blanchette

Lehr- und Forschungseinheit für
Theoretische Informatik

Stand: 13. Juni 2023
Folien ursprünglich von PD Dr. David Sabel

Intuitive Berechenbarkeit

Hintergrund

- Was **kann** mit einem Computerprogramm berechnet werden?
- Was **kann nicht** mit einem Computerprogramm berechnet werden?
- Aus der (Programmier-)Erfahrung:

  Man hat ein gewisses Gefühl dafür, was berechnet werden kann (und was nicht).

- Das ist der intuitive Begriff der Berechenbarkeit.
- Wie beweist man, dass etwas nicht berechnet werden kann?

  Diese Frage ist auch von praktischem Nutzen:
  Suche nach „passendem“ Algorithmus ist sinnlos.

Historie

- Berühmte Mathematiker/Informatiker versuchten in den 1930er Jahren den Begriff der Berechenbarkeit zu formalisieren.
- Dafür entwarfen sie verschiedene Modelle.
- Insbesondere sind zu nennen:
  - Alan Turing (Turingmaschine)
  - Alonzo Church (Lambda-Kalkül)

Berechenbare Funktion

Eine (partielle oder totale) Funktion $f : \mathbb{N}^k \to \mathbb{N}$ nennen wir **berechenbar**, wenn es einen Algorithmus einer modernen Programmiersprache gibt, der f berechnet.

- Bei Eingabe $(n_1, \ldots, n_k)$ stoppt der Algorithmus nach endlich vielen Berechnungsschritten und gibt den Wert von $f(n_1, \ldots, n_k)$ aus.
- Wenn $f(n_1, \ldots, n_k)$ undefiniert ist (f ist also eine partielle Funktion), dann stoppt der Algorithmus nicht.

Beispiele (1)

Der Algorithmus

Eingabe: Zahl $n \in \mathbb{N}$
Beginn
  solange *true* **tue**
    skip

berechnet $f_1 : \mathbb{N} \to \mathbb{N}$ mit $f_1(x) =$ undefiniert für alle x.

Der Algorithmus

Eingabe: Zahlen $n_1, n_2 \in \mathbb{N}$
Beginn
  $hilf := n_1 + n_2$;
  return *hilf*

berechnet die Funktion $f_2 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ mit $f_2(x, y) = x + y$.

f_1 und f_2 sind daher berechenbar.

Beispiele (2)

$$f_3(n) = \begin{cases} 1, & \text{falls } n \text{ ein Präfix der Ziffern der Dezimalzahl-} \\ & \text{darstellung von } \pi \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

Z.B. gilt

- $f_3(31) = 1$ und $f_3(314) = 1$
- $f_3(2) = 0$ und $f_3(315) = 0$

Ist f_3 berechenbar?

Beispiele (2)

$$f_3(n) = \begin{cases} 1, & \text{falls } n \text{ ein Präfix der Ziffern der Dezimalzahl-} \\ & \text{darstellung von } \pi \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

Z.B. gilt

- $f_3(31) = 1$ und $f_3(314) = 1$
- $f_3(2) = 0$ und $f_3(315) = 0$

Ist f_3 berechenbar?

Ja: Sei n eine k-stellige Zahl. Es gibt Algorithmen, die die ersten k Stellen von π berechnen. Danach kann man vergleichen.

Beispiele (3)

$$f_4(n) = \begin{cases} 1, & \text{falls } n \text{ ein Teilwort der Ziffern} \\ & \text{der Dezimalzahldarstellung von } \pi \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

Ist f_4 berechenbar?

Beispiele (3)

$$f_4(n) = \begin{cases} 1, & \text{falls } n \text{ ein Teilwort der Ziffern} \\ & \text{der Dezimalzahldarstellung von } \pi \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

Ist f_4 berechenbar?

Unklar. Es gibt die Vermutung, dass jede x-beliebige Ziffernfolge irgendwann als Folge in π auftaucht, aber die Frage ist bisher ungelöst.

Beispiele (4)

$$f_5(n) = \begin{cases} 1, & \text{falls die Dezimaldarstellung von } \pi \text{ das Wort } 3^n \\ & \text{als Teilwort enthält} \\ 0, & \text{sonst} \end{cases}$$

Ist f_5 berechenbar?

Beispiele (4)

$$f_5(n) = \begin{cases} 1, & \text{falls die Dezimaldarstellung von } \pi \text{ das Wort } 3^n \\ & \text{als Teilwort enthält} \\ 0, & \text{sonst} \end{cases}$$

Ist f_5 berechenbar?

Ja:

- Entweder: $f_5(n) = 1$ für alle n
- Oder: Es gibt n_0, sodass π 3^{n_0} als Teilwort hat, aber alle Teilworte 3^n mit $n > n_0$ nicht mehr besitzt.
  Dann ist $f_5(n) = \begin{cases} 1, \text{falls } n \leq n_0 \\ 0, \text{falls } n > n_0 \end{cases}$
- Egal, welcher Fall zutrifft, wir können für beide Fälle Algorithmen angeben, die f_5 berechnen.

Beachte: Unsere Definition von Berechenbarkeit ist nicht konstruktiv.
Wir müssen keinen Algorithmus liefern, sondern nur einen Beweis, dass einer existiert.

Beispiele (5)

$$f_6(n) = \begin{cases} 1, & \text{falls deterministische LBAs genau die gleichen} \\ & \text{Sprachen erkennen wie nichtdeterministische LBAs} \\ 0, & \text{sonst} \end{cases}$$

Ist f_6 berechenbar?

Beispiele (5)

$$f_6(n) = \begin{cases} 1, & \text{falls deterministische LBAs genau die gleichen} \\ & \text{Sprachen erkennen wie nichtdeterministische LBAs} \\ 0, & \text{sonst} \end{cases}$$

Ist f_6 berechenbar?

Ja: Entweder ist $f_6(x) = 1$ oder $f_6(x) = 0$.
Beide Funktionen sind berechenbar
(unabhängig davon, ob das 1. LBA-Problem eine positive oder negative Lösung hat).

Beispiele (6)

$$f^r(n) = \begin{cases} 1, & \text{falls } n \text{ ein Präfix der Ziffern der} \\ & \text{Dezimalzahldarstellung von } r \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

Ist f^r für jedes $r \in \mathbb{R}$ berechenbar?

Beispiele (6)

$$f^r(n) = \begin{cases} 1, & \text{falls } n \text{ ein Präfix der Ziffern der} \\ & \text{Dezimalzahldarstellung von } r \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

Ist f^r für jedes $r \in \mathbb{R}$ berechenbar?

Nein: Wir bräuchten genauso viele verschiedene Algorithmen wie es reelle Zahlen gibt.
Es gibt nur abzählbar viele Algorithmen einer Programmiersprache,
aber überabzählbar viele reelle Zahlen.

Churchsche These

Im folgenden untersuchen wir Modelle zur Berechenbarkeit

- Turingmaschinen (in dieser Stunde)
- WHILE-Programme (nur in FSK)
- GOTO-Programme (nur in FSK)
- μ-rekursive Funktionen (nur in FSK)

Resultat: Alle führen zum selben Begriff der Berechenbarkeit.

Churchsche These:

Die Klasse der Turingberechenbaren (äquivalent WHILE-berechenbaren, GOTO-berechenbaren, μ-rekursiven) Funktionen stimmt genau mit der Klasse der intuitiv berechenbaren Funktionen überein.

Die Churchsche These kann man nicht beweisen, da der Begriff „intuitiv berechenbar“ nicht formal gefasst werden kann.

Turingmaschinen: Wiederholung

- Eine Turingmaschine ist ein 7-Tupel $(Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit Zuständen Z, Eingabealphabet Σ, Bandalphabet $\Gamma \supset \Sigma$, Blank-Symbol $\square$, Startzustand z_0, Endzuständen $E \subseteq Z$ und Überführungsfunktion δ.
- DTM: $\delta : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$
- NTM: $\delta : Z \times \Gamma \to \mathcal{P}(Z \times \Gamma \times \{L, R, N\})$
- TM-Konfiguration $a_1 \cdots a_m z a_{m+1} \cdots a_n$, wobei $a_1 \cdots a_n \in \Gamma^*$ der entdeckte Teil des Bands ist, TM in Zustand z ist und der Schreib-Lesekopf unter a_{m+1} ist.

Turingberechenbarkeit

Definition (Turingberechenbarkeit)

Sei $bin(n)$ die Binärdarstellung von $n \in \mathbb{N}$.

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt Turingberechenbar, falls es eine (deterministische) Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $n_1, \ldots, n_k, m \in \mathbb{N}$ gilt:

$$f(n_1, \ldots, n_k) = m$$
$$\text{g.d.w.}$$
$$z_0 bin(n_1) \# \cdots \# bin(n_k) \vdash^* \square \cdots \square z_e bin(m) \square \cdots \square \text{ mit } z_e \in E$$

Eine Funktion $f : \Sigma^* \to \Sigma^*$ heißt Turingberechenbar, falls es eine (deterministische) Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $u, v \in \Sigma^*$ gilt:

$$f(u) = v \text{ g.d.w. } Start_M(u) \vdash^* \square \cdots \square z_e v \square \cdots \square \text{ mit } z_e \in E$$

Konsequenz

Eine Konsequenz der Definition ist:

Falls $f(n_1, \ldots, n_k)$ bzw. $f(u)$ undefiniert ist, dann kann die Maschine ewig laufen.

Beispiele

Nachfolgerfunktion

Die Funktion $f(x) = x + 1$ für alle $x \in \mathbb{N}$ ist Turingberechenbar.
Wir haben ein Beispiel bereits gesehen.

Beispiele

Nachfolgerfunktion

Die Funktion $f(x) = x + 1$ für alle $x \in \mathbb{N}$ ist Turingberechenbar.
Wir haben ein Beispiel bereits gesehen.

Identitätsfunktion

Die Funktion $f(x) = x$ für alle $x \in \mathbb{N}$ ist Turingberechenbar:
Für die Turingmaschine $M = (\{z_0\}, \{0, 1, \#\}, \{0, 1, \#\square\}, \delta, z_0, \square, \{z_0\})$ mit
$\delta(z_0, a) = (z_0, a, N)$ für alle $a \in \{0, 1, \#, \square\}$ gilt:
$z_0 bin(n) \vdash^* z_0 bin(n)$ für alle $n \in \mathbb{N}$.

Beispiele

Nachfolgerfunktion

Die Funktion $f(x) = x + 1$ für alle $x \in \mathbb{N}$ ist Turingberechenbar.
Wir haben ein Beispiel bereits gesehen.

Identitätsfunktion

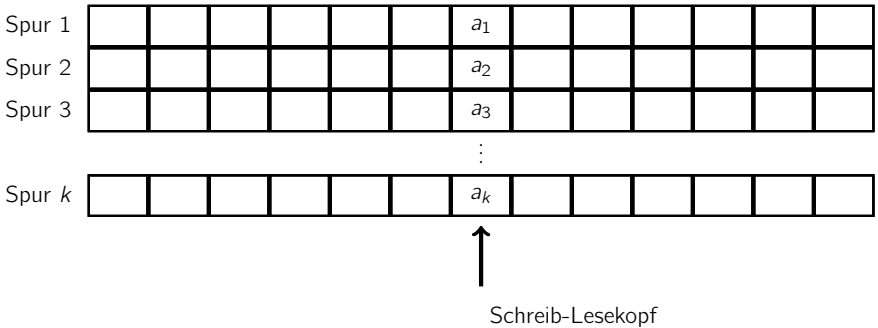
Die Funktion $f(x) = x$ für alle $x \in \mathbb{N}$ ist Turingberechenbar:
Für die Turingmaschine $M = (\{z_0\}, \{0, 1, \#\}, \{0, 1, \#, \square\}, \delta, z_0, \square, \{z_0\})$ mit $\delta(z_0, a) = (z_0, a, N)$ für alle $a \in \{0, 1, \#, \square\}$ gilt:
$z_0 bin(n) \vdash^* z_0 bin(n)$ für alle $n \in \mathbb{N}$.

Überall undefinierte Funktion

Die Funktion $f(x) = \perp$ für alle x ist Turingberechenbar, da die TM $M = (\{z_0\}, \{0, 1, \#\}, \{0, 1, \#, \square\}, \delta, z_0, \square, \emptyset)$ mit $\delta(z_0, a) = (z_0, a, N)$ für keine Eingabe akzeptiert (sondern stets in eine Endlosschleife geht).

Mehrspuren-Turingmaschinen (1)

Erweiterung: Das Band hat k Spuren:



Mehrspuren-Turingmaschinen (2)

Definition (Mehrspuren-Turingmaschine)

Eine k-**Spuren-Turingmaschine** ($k \in \mathbb{N}_{>0}$) ist ein 7-Tupel $(Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z ist eine endliche Menge von Zuständen,
- Σ ist das (endliche) Eingabealphabet,
- $\Gamma \supset \Sigma$ ist das (endliche) Bandalphabet,
- δ ist die Zustandsüberführungsfunktion:
  - für eine DTM: $\delta : Z \times \Gamma^k \to Z \times \Gamma^k \times \{L, R, N\}$
  - für eine NTM: $\delta : Z \times \Gamma^k \to \mathcal{P}(Z \times \Gamma^k \times \{L, R, N\})$,
- $z_0 \in Z$ ist der Startzustand,
- $\square \in \Gamma \setminus \Sigma$ ist das Blank-Symbol und
- $E \subseteq Z$ ist die Menge der Endzustände.

Berechenbarkeit: Ein- und Ausgabe auf der ersten Spur.

Mehrspuren-Turingmaschinen (3)

Berechenbarkeit: Ein- und Ausgabe auf der ersten Spur.

Satz

Jede Mehrspuren-Turingmaschine kann auf einer 1-Spur-Turingmaschine simuliert werden.

Mehrspuren-Turingmaschinen (3)

Berechenbarkeit: Ein- und Ausgabe auf der ersten Spur.

Satz

Jede Mehrspuren-Turingmaschine kann auf einer 1-Spur-Turingmaschine simuliert werden.

Beweis: Konstruktion der 1-Spur-TM:

- Verwende als Bandalphabet $\Gamma \cup \Gamma^k$.
- Eingabealphabet Σ und Blank-Symbol $\square$.
- Aus Eingabe w aus Σ^* erzeugt die TM die Mehrspurendarstellung: Ersetze $a \in \Sigma$ durch k-Tupel $(a, \square, \ldots, \square)$.
- Anschließend wird die Mehrspurenmaschine simuliert.
- Nach Akzeptanz der Mehrspurenmaschine: Erzeuge 1-Spur-Darstellung, d.h. ersetze alle $(a_1, \ldots, a_k)$ durch a_1. $\square$

Beispiel einer Mehrspurenmaschine

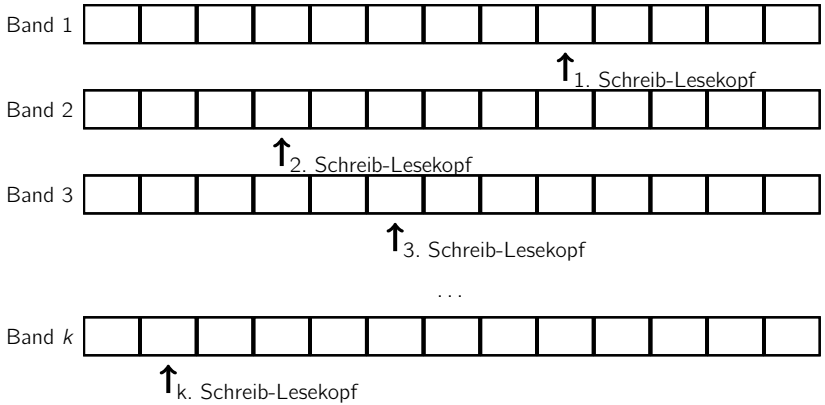
- TM, die $\{wcw \mid w \in \{a, b\}^+\}$ erkennt.
- Wir konzentrieren uns auf die 2-Spuren-Darstellung.

Idee:

- Eingabe wcw auf Spur 1 wird nur gelesen nicht verändert.
- Auf Spur 2 wird nur $\square$ und D zum Markieren von Zeichen auf Spur 1 verwendet.
- Markieren: Erst ein Zeichen im linken w, dann selbes Zeichen im rechten w.
- Zustände $Z = \{z_0, z_{1a}, z_{1b}, z_{2a}, z_{2b}, z_3, \ldots, z_9\}$
  $z_{1a}, z_{1b}, z_{2a}, z_{2b}$ „speichern“ das gelesene Zeichen (a oder b).
  - z_0 ist der Startzustand.
  - z_8 ist der einzige akzeptierende Zustand.
  - z_9 ist Müllzustand (zum Verwerfen).

Siehe Skript für die Details.

Mehrbandmaschinen



Schreib-Leseköpfe bewegen sich **unabhängig**.

Mehrbandmaschinen (2)

Definition (Mehrband-Turingmaschine)

Eine k-Band-Turingmaschine (für $k \in \mathbb{N}_{>0}$) ist ein 7-Tupel $(Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z ist eine endliche Menge von Zuständen,
- Σ ist das (endliche) Eingabealphabet,
- $\Gamma \supset \Sigma$ ist das (endliche) Bandalphabet,
- δ ist die Zustandsüberführungsfunktion
  - für eine DTM: $\delta : Z \times \Gamma^k \to Z \times \Gamma^k \times \{L, R, N\}^k$
  - für eine NTM: $\delta : Z \times \Gamma^k \to \mathcal{P}(Z \times \Gamma^k \times \{L, R, N\}^k)$,
- $z_0 \in Z$ ist der Startzustand,
- $\square \in \Gamma \setminus \Sigma$ ist das Blank-Symbol und
- $E \subseteq Z$ ist die Menge der Endzustände.

Berechenbarkeit: Ein- und Ausgabe auf dem ersten Band, anfangs alle anderen leer

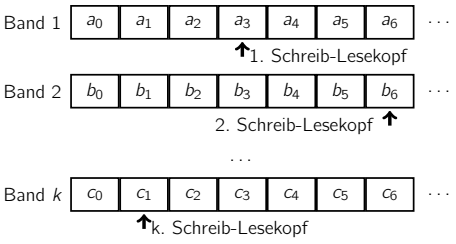
Mehrbandmaschinen (3)

Theorem

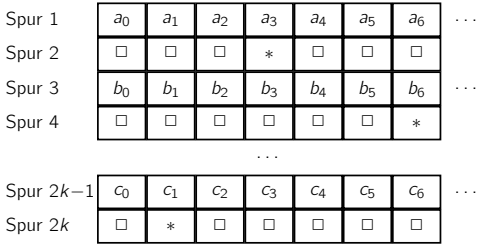
Jede Mehrband-TM kann von einer 1-Band-TM simuliert werden.

Beweis:

- Sei M eine k-Band-TM.
- Wir verwenden eine $2k$-Spuren-TM um M zu simulieren.



k-Band-TM



Simulation mit $2k$ Spuren

Mehrbandmaschinen (4)

- Eingabe $w \in \Sigma^*$ auf dem Eingabeband der Mehrband-Maschine
- 1-Band-Maschine erzeugt Darstellung in $2k$ Spuren.
- 1-Band-Maschine simuliert anschließend Berechnungsschritte.
  Simulation eines Berechnungsschrittes der Mehrband-TM:
  1. Lies den verwendeten Bandbereich von links nach rechts, um die Kopfpositionen zu finden.
  2. Speichere dabei alle k Bandinhalte an den Kopfpositionen durch Zustände.
  3. Führe Schritt der Mehrband-TM aus, durch Anpassen der Bandinhalte und der Kopfpositionen.
- Bei Akzeptanz der Mehrband-TM transformiert die 1-Band-TM die Spurendarstellung in die Darstellung der Ausgabe. □