# Formale Grundlagen der Informatik

# 8

## Berechenbarkeit

## Rekursiv aufzählbare und rekursive Mengen

# Algorithmus – intuitiver Begriff

- Ein **Algorithmus** ist eine Folge von Anweisungen, die **Eingabedaten** in **Ausgabedaten** überführt.
- Dabei muss für jede Eingabe eindeutig feststehen:
  - Welche Anweisung wird zuerst ausgeführt?
  - Welche Anweisung folgt auf eine gerade ausgeführte?
  - Wann ist der Algorithmus beendet?
- Der Algorithmus muss für alle (passenden) Eingabedaten die Ausgabedaten korrekt berechnen, ohne dass er angepasst werden muss.

➢ Ein Algorithmus *berechnet eine Funktion*.

# Berechenbarkeit

- Eine Funktion heißt (algorithmisch) **berechenbar**, wenn es einen Algorithmus gibt, der alle ihre Argumente korrekt in die zugehörigen Funktionswerte überführt.

➢ **Eine Funktion heißt berechenbar, wenn es ein C-/Java-/Python-/… Programm gibt, das diese Funktion berechnet.**

- *Herausforderungen:*
  1. Nachweis, dass Funktionen nicht berechenbar sind.
  2. Begriff der Berechenbarkeit ohne Bezug zu Programmiersprachen. (siehe z.B. das 10. Problem aus dem Vortrag von David Hilbert auf dem Internationalen Mathematikerkongress im Jahr 1900: *Man gebe ein Verfahren an, …*) ⟶ *Frage nach der Existenz eines Algorithmus*

# Einige Begriffe der Berechenbarkeit

- **WHILE-Berechenbarkeit**
  Eine Funktion ist berechenbar, wenn sie durch ein WHILE-Programm berechnet werden kann.
- **Partiell-rekursive Funktionen**
  Eine Funktion ist berechenbar, wenn sie mithilfe gewisser Schemata „rekursiv auf einfachste Funktionen zurückgeführt“ werden kann.
- **RAM-Berechenbarkeit**
  Eine Funktion ist berechenbar, wenn sie durch eine Registermaschine (eine einfache Abstraktion von Assemblersprachen) berechnet werden kann.
- **Turing-Berechenbarkeit**
  Eine Funktion heißt berechenbar, wenn sie durch eine Turing-Maschine berechnet werden kann.
- **…**

# Church-Turing-These

- Alle Berechenbarkeitsbegriffe sind formal definiert.
  Ihre **Äquivalenz** ist nachgewiesen.
- Das betrifft auch die sehr vielen Varianten in der Definition von Turing-Maschinen.

➢ **Church-Turing-These:**
***Die Menge der Turing-berechenbaren Funktionen stimmt mit der Menge der intuitiv berechenbaren Funktionen überein.***
  - kann natürlich nicht bewiesen werden
  - bisher alle anerkannten Definitionen der Berechenbarkeit äquivalent
  - ist allgemein anerkannt

# Turing-Berechenbarkeit – Idee

- Formales Modell für „schriftliches Rechnen“
  - z.B. schriftliche Addition, Division, Polynomdivision, …
  - Rechnen „auf kariertem Papier“
- Jede TM definiert einen Algorithmus.
- **Eingabedaten:** Eingabewort
- **Ausgabe** (Funktionswert): Bandinhalt nach dem Halten der TM.

# Turing-Berechenbarkeit – Definition

- Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, *, F)$ eine Turing-Maschine.
- Die von $M$ **induzierte Funktion** $f_M: \Sigma^* \to (\Gamma \setminus \{*\})^*$ ist wie folgt definiert:
  $f_M(w) = v$ gdw.
  1. $q_0 w \vdash^* v_1 p v_2$
  2. $p \in F$
  3. $*^i\, v *^j = v_1 v_2$ für Zahlen $i, j \geq 0$.
- Eine Funktion $f$ heißt **Turing-berechenbar**, wenn es eine Turing-Maschine $M$ gibt mit $f_M = f$.

➢ Die induzierte Funktion einer Turing-Maschine kann eine *partielle* Funktion sein.

# Turing-Berechenbarkeit – Beispiel

**Eingabe:** Dezimaldarstellung einer natürlichen Zahl

**Ausgabe:** Nachfolger dieser Zahl

Seien $\Sigma_{dec} = \{0,1,2,3,4,5,6,7,8,9\}$ und

$$M_{succ} = (\{q_0, +, f\}, \Sigma_{dec}, \Sigma_{dec} \cup \{*\}, \delta_{succ}, q_0, *, \{f\}).$$

| | * | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $q_0$ | $(+, *, \mathrm{L})$ | $(q_0, 0, \mathrm{R})$ | $(q_0, 1, \mathrm{R})$ | $(q_0, 2, \mathrm{R})$ | $(q_0, 3, \mathrm{R})$ | $(q_0, 4, \mathrm{R})$ | $(q_0, 5, \mathrm{R})$ | $(q_0, 6, \mathrm{R})$ | $(q_0, 7, \mathrm{R})$ | $(q_0, 8, \mathrm{R})$ | $(q_0, 9, \mathrm{R})$ |
| $+$ | $(f, 1, \mathrm{L})$ | $(f, 1, \mathrm{L})$ | $(f, 2, \mathrm{L})$ | $(f, 3, \mathrm{L})$ | $(f, 4, \mathrm{L})$ | $(f, 5, \mathrm{L})$ | $(f, 6, \mathrm{L})$ | $(f, 7, \mathrm{L})$ | $(f, 8, \mathrm{L})$ | $(f, 9, \mathrm{L})$ | $(+, 0, \mathrm{L})$ |

# Rekursive Mengen - Definition

- *Intuitiv:* Eine Menge ist entscheidbar, wenn ein Algorithmus entscheiden kann, ob seine Eingabe ein Element der Menge ist oder nicht.
- Sei $\Sigma$ ein Alphabet und $L \subseteq \Sigma^*$. Die **charakteristische Funktion** von $L$ ist die Funktion $\varphi_L : \Sigma^* \to \{0,1\}$ vermöge

totale Funktion!

$$\varphi_L(x) = \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{falls } x \notin L \end{cases}$$

- Eine Menge ist **rekursiv (entscheidbar)**, wenn ihre charakteristische Funktion Turing-berechenbar ist. $\longrightarrow$ TM hält bei jeder Eingabe!

➢ Auf dem Turing-Band muss nach dem Stoppen exakt 0 oder 1 stehen.

  ➢ Ggf. alles andere löschen vor dem Stoppen!

# Entscheidbare Menge – Beispiel

$$M' = (\{q_0, q_1, q_2, q_3, q_4, f\}, \{a, b\}, \{a, b, *\}, \delta, q_0, *, \{f\})$$
$$\cup \{r, \ell\}$$

| | $a$ | $b$ | $*$ |
|---|---|---|---|
| $q_0$ | $(q_1, *, \mathrm{R})$ | $(r, 0, \mathrm{R})$ | $\cancel{(f, *, \mathrm{R})} \rightarrow \boxed{(f, 1, \mathrm{R})}$ |
| $q_1$ | $(q_1, a, \mathrm{R})$ | $(q_1, b, \mathrm{R})$ | $(q_2, *, \mathrm{L})$ |
| $q_2$ | $(\ell, 0, \mathrm{L})$ | $(q_3, *, \mathrm{L})$ | $(\ell, 0, \mathrm{L})$ |
| $q_3$ | $(q_3, a, \mathrm{L})$ | $(q_3, b, \mathrm{L})$ | $(q_0, *, \mathrm{R})$ |

*Beobachtung:*
Wenn die TM $M$ stoppt, ist das Eingabeband leer.

$$L(M) = \{\, a^n b^n \mid n \geq 0 \,\}$$

# Entscheidbare Menge – Beispiel

$$M' = (\{q_0, q_1, q_2, q_3, q_4, r, \ell, f\}, \{a, b\}, \{a, b, *\}, \delta, q_0, *, \{f\})$$

| | $a$ | $b$ | $*$ |
|---|---|---|---|
| $q_0$ | $(q_1, *, \mathrm{R})$ | $(r, 0, \mathrm{R})$ | $(f, 1, \mathrm{R})$ |
| $q_1$ | $(q_1, a, \mathrm{R})$ | $(q_1, b, \mathrm{R})$ | $(q_2, *, \mathrm{L})$ |
| $q_2$ | $(\ell, 0, \mathrm{L})$ | $(q_3, *, \mathrm{L})$ | $(\ell, 0, \mathrm{L})$ |
| $q_3$ | $(q_3, a, \mathrm{L})$ | $(q_3, b, \mathrm{L})$ | $(q_0, *, \mathrm{R})$ |
| $r$ | $(r, *, \mathrm{R})$ | $(r, *, \mathrm{R})$ | $(f, *, \mathrm{R})$ |
| $\ell$ | $(\ell, *, \mathrm{L})$ | $(\ell, *, \mathrm{L})$ | $(f, *, \mathrm{L})$ |

*Bei Fehlersituation:*
*Band ist nicht leer!*

- In $r$ alles von links nach rechts löschen.
- In $\ell$ alles von rechts nach links löschen.

$M'$ **berechnet** $\varphi_L$ von $L = \{\, a^n b^n \mid n \geq 0 \,\}$ $\longrightarrow$ $M'$ **entscheidet** $L = \{\, a^n b^n \mid n \geq 0 \,\}$

# Rekursiv aufzählbare Mengen

- *Bisher:*
  - TM als Akzeptoren für Sprachen
  - TM als Algorithmen / zur Berechnung von Funktionen
- *Jetzt:* TM als **Generatoren** von Sprachen:
  1. TM beginnt auf leerem Eingabeband
  2. TM schreibt Wort für Wort die Elemente einer Sprache auf das Band, voneinander getrennt durch #

  ➢ TM **zählt** die Wörter der Sprache **auf**
  (in beliebiger Reihenfolge, ggf. auch wiederholt)

  ➢ die von der TM $M$ rekursiv aufgezählte Sprache $G(M)$

# Rekursiv aufzählbare Mengen

- Eine Menge $L$ heißt **rekursiv aufzählbar** wenn es eine TM $M$ gibt, so dass $G(M) = L$.

**Satz 8.1:** Zu jeder rekursiv aufzählbaren Menge $L$ kann eine Turing-Maschine konstruiert werden, die $L$ akzeptiert.

**Beweis:** Sei $M$ eine TM mit $G(M) = L$. Konstruieren eine TM $M'$ mit $L(M') = L$:

1. Erzeuge ein Trennsymbol \$ hinter dem Eingabewort $w$.
2. Simuliere $M$ hinter \$, bis ein Wort aus $G(M)$ von erzeugt ist.
3. Vergleiche das erzeugte Wort mit der Eingabe $w$.
   a) Bei Gleichheit halten (also die Eingabe $w$ akzeptieren).
   b) Bei Ungleichheit das erzeugte Wort löschen und zurück zu Schritt 2.: dort das nächste Wort aus $G(M)$ erzeugen.
   Bei endlicher Menge $G(M)$: Das zuletzt erzeugte Wort immer wieder erzeugen.

# Rekursiv aufzählbare Mengen

**Satz 8.1:** Zu jeder rekursiv aufzählbaren Menge $L$ kann eine Turing-Maschine konstruiert werden, die $L$ akzeptiert.

*Fortsetzung des Beweises:*

- Zustände von $M'$ sind Paare, erste Komponente ist der aktuelle Zustand von $M$, zweite Komponente steuert den Ablauf der Arbeit von $M'$.
- Falls $w \in G(M)$, dann $w \in L(M')$.
- Falls $w \notin G(M)$, dann wird $M'$ nie halten, also $w \notin L(M')$. ■

➢ Gilt die Umkehrung von Satz 8.1?

# Rekursiv aufzählbare Mengen

**Lemma 8.2:** Wenn $L \subseteq \Sigma^*$ rekursiv ist, dann ist $L$ rekursiv aufzählbar.

**Beweis:** Sei $M$ eine TM, die $L$ entscheidet, also mit $f_M = \varphi_L$.

- Sei $\Sigma = \{a_1, a_2, \dots, a_n\}$. **Kanonische Ordnung auf $\boldsymbol{\Sigma^*}$:**
  - Anordnung der Wörter aufsteigend der Länge nach;
  - bei gleicher Länge alphabetisch (bezogen auf eine Anordnung der Buchstaben in $\Sigma$).

*Z.B.* für $\Sigma = \{0,1\}$:
$\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots$

- Konstruieren eine TM $M'$ mit $G(M') = L$ wie folgt:

  Für alle Wörter $w$ in kanonischer Anordnung:

  Simuliere $M$ bei Eingabe $w$. Falls $f_M(w) = 1$, erzeuge $w$ in der Aufzählung. ■

➢ *Problem:* Funktioniert so nicht für Sprachen, die von einer TM akzeptiert werden, da $M$ nicht für jedes Wort halten muss.

# Kantors Anordnung für Paare positiver ganzer Zahlen

- Bekannt von der ersten Diagonalisierung, z.B. für Brüche
- Ordne die Paare der Form $(i, j)$ aufsteigend
  1. nach ihren Summen $i + j$ und
  2. bei gleicher Summe aufsteigend nach $i$.

  *also* (1,1), (1,2), (2,1), (1,3), (2,2), (3,1), (1,4), (2,3), (3,2), (4,1), (1,5), …

➢ Eine TM $M_{\mathrm{Paar}}$ kann alle Paare in dieser Anordnung aufzählen.

➢ Jedes Paar $(i, j)$ wird von $M_{\mathrm{Paar}}$ irgendwann („nach endlicher Zeit“) erzeugt.

# Rekursiv aufzählbare Mengen

**Satz 8.3:** Zu jeder TM $M$ kann eine TM $M'$ konstruiert werden, die die Menge $L(M)$ rekursiv aufzählt.

**Beweis:** Sei $L(M) \subseteq \Sigma^*$. Konstruieren TM $M'$ mit geeigneten Tupeln aus Zustände.

Solange die TM $M'$ nicht hält, wiederholt sie folgende Schritte:

1. Simuliere $M_{\text{Paar}}$, so dass das nächste Paar $(i, j)$ erzeugt wird
2. Erzeuge (hinter $(i, j)$ und einem Trennsymbol) das Wort $w_i$ in der kanonischen Anordnung auf $\Sigma$.
3. Simuliere die ersten $j$ Schritte von der TM $M$ auf $w_i$.
   Falls $M$ in dieser Simulation hält, dann gehe ganz nach links und erzeuge $w_i\#$.

# Rekursiv aufzählbare Mengen

**Satz 8.3:** Zu jeder TM $M$ kann eine TM $M'$ konstruiert werden, die die Menge $L(M)$ rekursiv aufzählt.

*Fortsetzung des Beweises:*

- Falls $w \in L(M)$, dann gibt es ein Paar $(i, j)$, so dass $w = w_i$ in kanonischer Anordnung und $M$ hält bei Eingabe $w_i$ nach $j$ Schritten.
  Dann gilt $w \in G(M')$.
- Falls $w \notin L(M)$ und $w = w_i$ in kanonischer Anordnung, dann wird $M$ nach keiner Anzahl von Schritten $j$ bei Eingabe $w_i$ halten.
  Daher gilt $w \notin G(M')$. $\blacksquare$

# Rekursiv aufzählbare Mengen

**Folgerung 8.4:** Eine Sprache wird genau dann von einer TM akzeptiert, wenn sie rekursiv aufzählbar ist. ■

- *Wir wissen außerdem:*
  Wenn eine Sprache rekursiv ist, dann ist sie rekursiv aufzählbar.
- Die Umkehrung gilt nicht (*siehe nächste Vorlesung: Unentscheidbarkeit*).
- *Aber es gilt:*

  Eine Sprache $L$ ist genau dann rekursiv, wenn sie und ihr Komplement rekursiv aufzählbar sind.

# Komplemente rekursiver Mengen

**Lemma 8.5:** Wenn eine Sprache $L$ rekursiv ist, dann ist ihr Komplement $\overline{L}$ rekursiv.

**Beweis:** Sei $L \subseteq \Sigma^*$ rekursiv.

Dann gibt es eine TM $M$ mit $f_M = \varphi_L$.

Sei $\overline{M}$ die TM mit Eingabealphabet $\Sigma$, die exakt wie $M$ arbeitet aber 0 ausgibt, wenn $M$ eine 1 ausgibt und 1 ausgibt, wenn $M$ eine 0 ausgibt.

Dann gilt $f_{\overline{M}} = \varphi_{\overline{L}}$ und somit ist $\overline{L}$ rekursiv. ∎

# Rekursive *vs.* rekursiv aufzählbare Mengen

**Satz 8.6:** Eine Sprache $L$ ist genau dann rekursiv, wenn sie und ihr Komplement rekursiv aufzählbar sind.

**Beweis:**

1. Sei $L$ rekursiv. Dann ist $\overline{L}$ rekursiv (Lemma 8.5).

   Dann sind $L$ und $\overline{L}$ rekursiv aufzählbar (Lemma 82).

2. Seien $L$ und $\overline{L}$ rekursiv aufzählbar. Dann gibt es Turing-Maschinen $M$ und $\overline{M}$ mit $L(M) = L$ und $L(\overline{M}) = \overline{L}$ (Folgerung 8.4).

   Konstruieren eine Turing-Maschine $N$, die $L$ entscheidet:

# Rekursive *vs.* rekursiv aufzählbare Mengen

Konstruieren eine Turing-Maschine $N$, die $L$ entscheidet:

i. Kopiere das Eingabewort $w$ hinter ein Trennzeichen.

ii. Wiederhole:

   1. Simuliere den nächsten Schritt von $M$ auf dem Eingabewort $w$.
      Falls $M$ in diesem Schritt hält, lösche den Bandinhalt, schreibe 1 und halte.
   2. Simuliere den nächsten Schritt von $\overline{M}$ auf der Kopie von $w$.
      Falls $\overline{M}$ in diesem Schritt hält, lösche den Bandinhalt, schreibe 0 und halte.

   Zustände von $N$ sind Tripel, 1. und 2. Komponente sind die aktuellen Zustände von $M$ bzw. $\overline{M}$, die 3. Komponente steuert den Ablauf.

Die TM $N$ berechnet somit die charakteristische Funktion $\varphi_L$ von $L$. ∎

# Sprachfamilien

- $\mathcal{L}$**(REC)** $= \{\, L \mid L$ ist rekursiv $\}$
- $\mathcal{L}$**(RE)** $= \{\, L \mid L$ ist rekursiv aufzählbar $\}$
  $= \{\, L \mid L = G(M)$ für eine TM $M \,\}$
  $= \{\, L \mid L = L(M)$ für eine TM $M \,\}$

**Lemma 8.7:** $\mathcal{L}$**(REG)** $\subseteq$ $\mathcal{L}$**(REC)**

**Beweis:** Sei $L \in \mathcal{L}(\mathrm{REG})$. Dann gibt es einen DEA $A$ mit $L(A) = L$.

Die TM $M_A$ simuliere $A$. Wenn $A$ akzeptiert, gibt $M_A$ eine 1 aus, sonst 0.

$\blacksquare$

# Hierarchiebeziehungen

- $\mathcal{L}$(REC) $= \{ L \mid L$ ist rekursiv $\}$
- $\mathcal{L}$(RE) $= \{ L \mid L$ ist rekursiv aufzählbar $\}$
  $= \{ L \mid L = G(M)$ für eine TM $M \}$
  $= \{ L \mid L = L(M)$ für eine TM $M \}$

**Folgerung 8.7:** $\mathcal{L}$(REG) $\subset$ $\mathcal{L}$(REC) $\subseteq$ $\mathcal{L}$(RE)

Teilmenge nach Lemma 8.7;
**echt** wegen $\{ a^n b^n \mid n \geq 0 \}$

Teilmenge nach Lemma 8.2;
*Echtheit siehe Vorlesung 9*