



Technische  
Universität  
Braunschweig



# Theoretische Informatik 2

Arne Schmidt

# Kapitel 3 – Berechen- und Entscheidbarkeit

# Kapitel 3.1 – Berechenbarkeit

# Beispiel 3.1

Sei  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  (für bel.  $\Sigma_1^*, \Sigma_2^*$ ) mit  $f(w) = \text{undef}$  für all  $w \in \Sigma_1^*$ .

Wie berechnet man  $f$ ?

Was dürfte ein Algorithmus ausgeben? Wenn er etwas ausgibt, wäre das ein definiertes Verhalten!

Wir müssen erst definieren, was „berechenbar“ bedeutet!

# Berechenbarkeit

Intuitiv: Eine (partielle) Funktion  $f: \Sigma_1^* \rightarrow_p \Sigma_2^*$  ist **berechenbar**, wenn es einen Algorithmus gibt, der mit Eingabe  $w \in \Sigma_1^*$

- nach endlich vielen Schritten akzeptiert und  $f(w)$  ausgibt, falls  $f(w)$  definiert ist
- nicht anhält oder nicht akzeptiert, falls  $f(w)$  nicht definiert ist.

Umgekehrt, berechnet jeder (deterministische) Algorithmus eine partielle Funktion.

# Beispiel 3.1

Sei  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  (für bel.  $\Sigma_1^*, \Sigma_2^*$ ) mit  $f(w) = \text{undef}$  für all  $w \in \Sigma_1^*$ .

Ein möglicher Algorithmus:

```
while true do  
    continue;  
end while
```

## Beispiel 3.2

Sei  $f_\pi: \mathbb{N} \rightarrow \mathbb{N}$  mit

$$f_\pi(n) = \begin{cases} 1, & \text{falls } n \text{ ein Prafix der Dezimaldarstellung von } \pi \text{ ist.} \\ 0, & \text{sonst} \end{cases}$$

Beispiele:

$$f_\pi(314) = 1, \quad f_\pi(5) = 0, \quad f_\pi(141) = 0$$

Ist diese Funktion berechenbar?

Ja! Sei  $n$  die Eingabe mit  $k$  vielen Ziffern. Dann:

- Approximiere  $\pi$  auf  $k - 1$  Nachkommastellen genau
- Vergleiche die Ziffern.
- Gib 1 zuruck, falls Vergleich erfolgreich, 0 sonst.

## Beispiel 3.3

Sei  $g_\pi: \mathbb{N} \rightarrow \mathbb{N}$  mit

$$g_\pi(n) = \begin{cases} 1, & \text{falls } n \text{ ein Infix der Dezimaldarstellung von } \pi \text{ ist.} \\ 0, & \text{sonst} \end{cases}$$

Beispiele:

$$g_\pi(314) = 1, \quad g_\pi(5) = 1, \quad g_\pi(141) = 1$$

Ist diese Funktion berechenbar?

Unbekannt!

Der Trick von eben funktioniert nicht mehr: Wir wissen nicht, wie lange wir approximieren müssen. Wenn es ein Infix ist, terminieren wir, wenn nicht, kennen wir keine Abbruchbedingung!

## Beispiel 3.4

$$\text{Sei } f_{P_{NP}}: \{0,1\}^* \rightarrow \{0,1\}^* \text{ mit}$$
$$f_{P_{NP}}(w) = \begin{cases} 0, & \text{falls } P = NP. \\ 1, & \text{falls } P \neq NP. \end{cases}$$

Ist diese Funktion berechenbar?

Ja!

Wir fordern nur, dass es einen Algorithmus gibt, nicht, dass wir ihn kennen!

Betrachte folgende zwei Algorithmen

- Gib immer 0 zurück.
- Gib immer 1 zurück.

Einer der beiden berechnet  $f_{P_{NP}}$ . Wir wissen nur nicht welcher.

P und NP definieren Klassen von Problemen, die auf einer DTM bzw. NTM in polynomieller Zeit gelöst werden können.

Bisher ungelöst, ob  $P = NP$  oder  $P \neq NP$ .

# Berechenbarkeit II

Intuitiv wollen wir eine (partielle) Funktion  $f: \Sigma_1^* \rightarrow_p \Sigma_2^*$  **berechenbar** nennen, wenn es einen Algorithmus gibt, der eine Eingabe  $w \in \Sigma_1^*$  nimmt, und

- nach endlich vielen Schritten akzeptiert und  $f(w)$  ausgibt, falls  $f(w)$  definiert ist
- nicht anhält oder nicht akzeptiert, falls  $f(w)$  nicht definiert ist.

Umgekehrt, berechnet jeder (deterministische) Algorithmus eine partielle Funktion.

Wir nennen eine Funktion **effektiv berechenbar**, wenn man den Algorithmus, der die Funktion berechnet, konkret angeben kann.

Analog nennt man ein Entscheidungsproblem **effektiv entscheidbar**, wenn man den Entscheidungsalgorithmus für das Problem kennt. (nächste VL)

## Beispiel 3.5

Sei  $h_\pi: \mathbb{N} \rightarrow \mathbb{N}$  mit

$$h_\pi(n) = \begin{cases} 1, & \text{falls } 5 \dots 5 \text{ (} n \text{ Mal die 5) ein Infix der Dezimaldarstellung von } \pi \text{ ist.} \\ 0, & \text{sonst} \end{cases}$$

Ist diese Funktion berechenbar?

Intuitiv: Nicht bekannt wie Beispiel 3.3.

Aber, es gibt zwei Fälle:

- Jedes Wort aus  $\{5\}^*$  kommt in  $\pi$  als Infix vor. Gib dann einfach immer 1 zurück.
- $\{5\}^{n_0}$  mit  $n_0 \in \mathbb{N}$  ist das längste Wort. Dann ist der Algorithmus wie folgt:
  - Falls  $n \leq n_0$ , gib 1 zurück.
  - Falls  $n > n_0$ , gib 0 zurück.



## Beispiel 3.7

### Definition 3.6

Eine Menge  $M$  ist **abzählbar**, wenn sie entweder leer ist oder eine surjektive Funktion  $f: \mathbb{N} \rightarrow M$  gibt.

Sei  $f_a: \mathbb{N} \rightarrow \mathbb{N}$  mit  $a \in \mathbb{R}$  und

$$f_a(n) = \begin{cases} 1, & \text{falls } n \text{ ein } \mathbf{Pr\ddot{a}fix} \text{ der Dezimaldarstellung von } a \text{ ist.} \\ 0, & \text{sonst} \end{cases}$$

Ist diese Funktion für jedes  $a$  berechenbar?

Nein!

Wir werden gleich sehen:

- Es gibt abzählbar viele Algorithmen, aber
- überabzählbar viele Reelle zahlen

Damit muss ein  $a$  existieren, sodass  $f_a$  nicht berechenbar ist!

## Beispiel 3.2 (reprise)

Sei  $f_\pi: \mathbb{N} \rightarrow \mathbb{N}$  mit

$$f_\pi(n) = \begin{cases} 1, & \text{falls } n \text{ ein Präfix der Dezimaldarstellung von } \pi \text{ ist.} \\ 0, & \text{sonst} \end{cases}$$

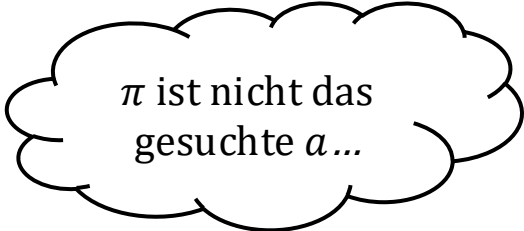
Beispiele:

$$f_\pi(314) = 1, \quad f_\pi(5) = 0, \quad f_\pi(141) = 0$$

Ist diese Funktion berechenbar?

Ja! Sei  $n$  die Eingabe mit  $k$  vielen Ziffern. Dann:

- Approximiere  $\pi$  auf  $k - 1$  Stellen genau
- Vergleiche die Ziffern.
- Gib 1 zurück, falls Vergleich erfolgreich, 0 sonst.



$\pi$  ist nicht das  
gesuchte  $a \dots$

# Berechenbarkeit

## Definition 3.8

Sei  $f : \Sigma_1^* \rightarrow_p \Sigma_2^*$  eine partielle Funktion. Wir nennen  $f$  **(Turing-)berechenbar**, wenn es eine deterministische Turing-Maschine  $M = (Q, \Sigma_1, \Gamma, q_0, \delta, Q_F)$  gibt, so dass für jede Eingabe  $w \in \Sigma_1^*$  gilt, dass

$$f(w) = w' \in \Sigma_2^* \iff q_0 w \rightarrow^* \dots \sqcup \sqcup q_f w' \sqcup \sqcup \dots,$$

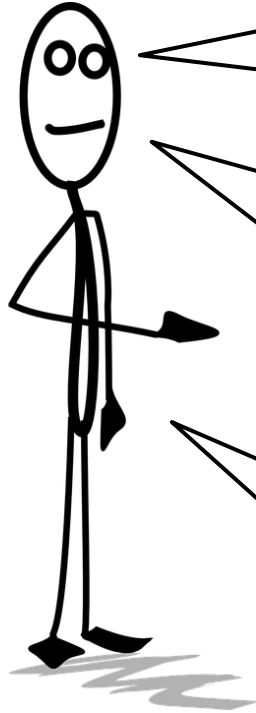
wobei  $q_f \in Q_F$ . Hierbei nehmen wir an, dass  $\Sigma_2 \subseteq \Gamma$  im Bandalphabet ist, und  $\sqcup \notin \Sigma_2$ .

Für partielle Funktionen  $f : \mathbb{N}^k \rightarrow_p \mathbb{N}$ , sagen wir dass  $f$  Turing-berechenbar ist, falls für jede Eingabe  $n_1, \dots, n_k \in \mathbb{N}$  gilt, dass

$$f(n_1, \dots, n_k) = n \in \mathbb{N} \iff q_0 \text{bin}(n_1) \# \dots \# \text{bin}(n_k) \rightarrow^* \dots \sqcup \sqcup q_f \text{bin}(n) \sqcup \sqcup \dots$$

wobei  $q_f \in Q_F$  und  $\text{bin}(n)$  die Binärdarstellung (ohne führende Nullen) von  $n$  ist.

## Bemerkung 3.9



DTMs sind hier keine Einschränkung: Zu jeder NTM existiert eine DTM!

DTMs sind allerdings natürlicher für Funktionen.

Vereinfachte Annahme: TMs ändern weder Zustand noch Kopf, sobald akzeptierender Zustand erreicht wurde.

→ Die Turing Maschine **hält**.

Anderfalls **bleibt** sie **stecken**, falls keine passende Transition in einem nicht-akzeptierenden Zustand existiert.

Für einen undefinierten Wert darf die TM keine Konfiguration wie in der Definition erreichen. D.h. sie muss

- Stecken bleiben (geht nicht für DTMs)
- Unendlich lange loopen

# Menge der Turing-Maschinen

## Beispiel 3.10

Die Funktionen  $f, f_\pi, f_{PNP}, h_\pi$  sind turing berechenbar.

Aber warum ist  $f_a$  nicht für jedes  $a \in \mathbb{R}$  turing-berechenbar?

## Lemma 3.11

Es gibt abzählbar viele Turing-Maschinen.

# Menge der Turing-Maschinen (Beweis)

## Lemma 3.11

Es gibt abzählbar viele Turing-Maschinen.

Beweis: Wir betrachten (det.) Turing-Maschinen der Form  $M = (Q, \Sigma, \Gamma, q_0, \delta, Q_F)$  mit

$$Q = \{q_0, \dots, q_k\},$$

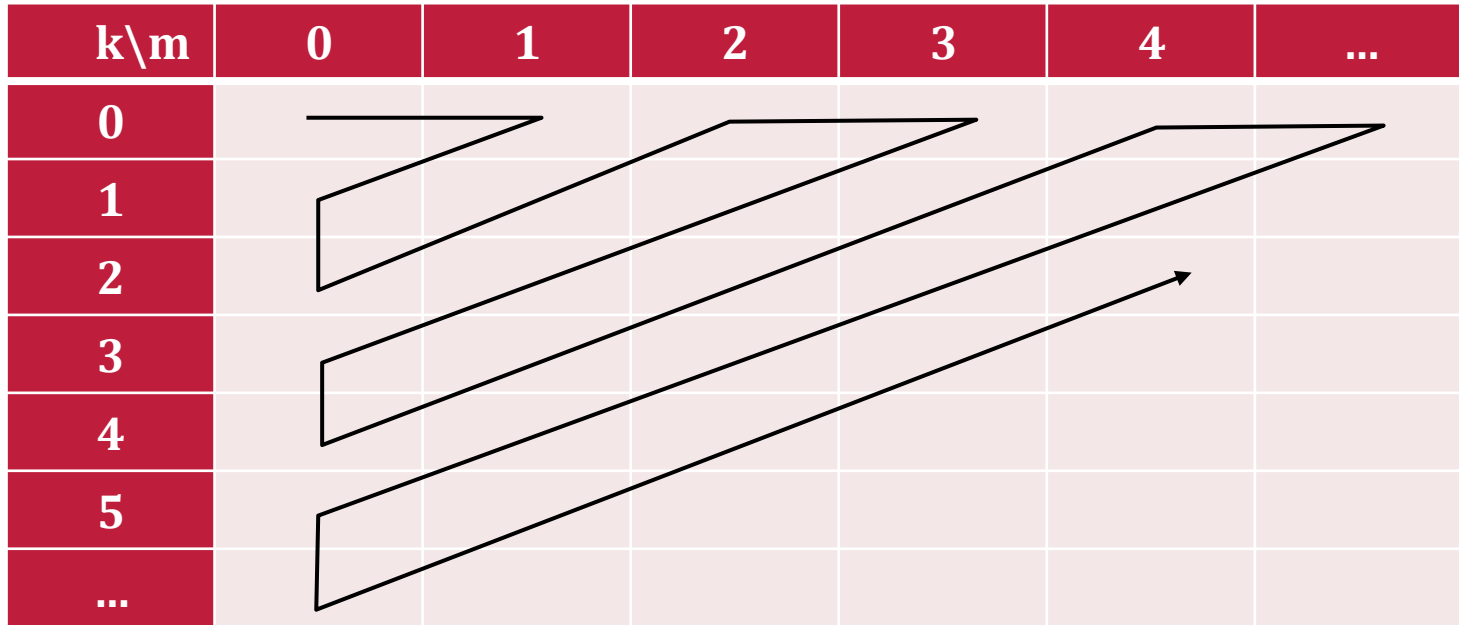
$$\Gamma = \Sigma \cup \{a_0, \dots, a_m, \$, \_ \}$$

Fixiert man  $m$  und  $k$ , gibt es nur endlich viele Turing-Maschinen. Die Anzahl an Transitionen kann durch folgenden Wert beschränkt werden:

$$((k + 1)(|\Sigma| + m + 3) \cdot 3)^{(k+1)(|\Sigma|+m+3)}$$

Nutze nun das Cantor'sche Diagonalverfahren, um alle Turing-Maschinen aufzuzählen.

# Diagonalverfahren



Zähle  $k$  und  $m$  auf, dann für jedes Tupel alle Turing Maschinen.

# Nicht-berechnbare Funktionen

## Lemma 3.12

Es seien  $\Sigma_1, \Sigma_2$  beliebige Alphabete. Es gibt nicht-berechenbare Funktionen  $f: \Sigma_1^* \rightarrow \Sigma_2^*$ .

Beweis:

Seien  $f_0, f_1, \dots$  alle berechenbaren Funktionen.

Definiere

$$f(n) := \begin{cases} 0, & \text{falls } f_n(n) = \text{undef} \\ f_n(n) + 1, & \text{sonst} \end{cases}$$

Annahme,  $f$  ist berechenbar.

Dann existiert  $m \in \mathbb{N}$  mit  $f = f_m$ .

Allerdings ist

$f(m) = f_m(m) + 1$ , falls definiert oder

$f(m) = 0$  für  $f_m(m) = \text{undef}$ .

Also können die beiden nicht gleich sein!

Alle berechenbaren Funktionen

$n$	$f$	$f_0$	$f_1$	$f_2$	$f_3$	$f_4$	...
0	0	<i>undef</i>	2	4	0	<i>undef</i>	...
1	5	1	4	100	0	1	...
2	106	0	5	105	0	4	...
3	1	2	<i>undef</i>	0	0	9	...
4	17	3	3	115	0	16	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$