Technische Universität Braunschweig

Theoretische Informatik 2

Arne Schmidt

Kapitel 3 – Berechen- und Entscheidbarkeit

Kapitel 3.1 – Berechenbarkeit

Beispiel 3.1

Sei $f: \Sigma_1^* \to \Sigma_2^*$ (für bel. Σ_1^*, Σ_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 $f: \Sigma_1^* \to_p \Sigma_2^*$ 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 $f: \Sigma_1^* \to \Sigma_2^*$ (für bel. Σ_1^*, Σ_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 $f_\pi: \mathbb{N} \to \mathbb{N}$ mit

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

Beispiele:

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

Ist diese Funktion berechenbar?

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

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

Beispiel 3.3

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

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

Beispiele:

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

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

Sei $f_{PNP}: \{0,1\}^* \to \{0,1\}^*$ mit

$$f_{PNP}(w) = \begin{cases} 0, & \text{falls P} = \text{NP}. \\ 1, & \text{falls P} \neq \text{NP}. \end{cases}$$

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 ≠ NP.

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_{PNP}. Wir wissen nur nicht welcher.

Berechenbarkeit II

Intuitiv wollen wir eine (partielle) Funktion $f: \Sigma_1^* \to_p \Sigma_2^*$ **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 nenn man ein Entscheidungsproblem **effektiv entscheidbar**, wenn man den Entscheidungsalgorithmus für das Problem kennt.

Beispiel 3.5

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

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

Ist diese Funktion berechenbar?

Intuitiv: Nicht bekannt wie Beispiel 3.3.
Aber, es gibt zwei Fälle:

- Jedes Wort aus $\{5\}^*$ kommt in π 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.


Okay, gibt es jetzt auch eine nicht-berechenbare Funktion?

Beispiel 3.7

Definition 3.6
Eine Menge M ist **abzählbar**, wenn sie entweder leer ist oder eine surjektive Funktion $f: \mathbb{N} \to M$ gibt.

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

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

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 $f_\pi: \mathbb{N} \to \mathbb{N}$ mit

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

Beispiele:

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

Ist diese Funktion berechenbar?

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

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

Berechenbarkeit

Definition 3.8
Sei $f : \Sigma_1^* \to_p \Sigma_2^*$ 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

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

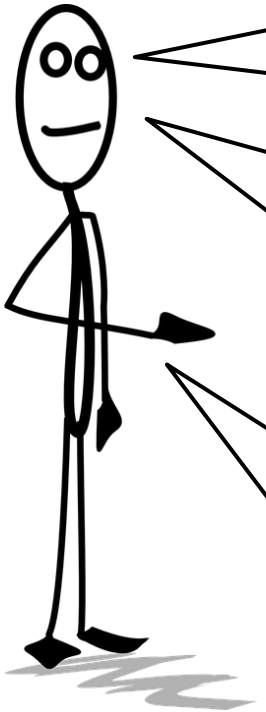
wobei $q_f \in Q_F$. Hierbei nehmen wir an, dass $\Sigma_2 \subseteq \Gamma$ im Bandalphabet ist, und $\sqcup \notin \Sigma_2$.

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

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

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
- In einer Konfiguration stehen bleiben, die nicht obiger Beschreibung entspricht.

Menge der Turing-Maschinen

Beispiel 3.10
Die Funktionen f, f_π, f_{PNP}, h_π 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 Turing-Maschinen der Form $M = (Q, \Sigma, \Gamma, q_0, \delta, Q_F)$ mit

$$Q = \{q_0, \dots, q_k\},$$
$$\Gamma = \Sigma \dot{\cup} \{a_0, \dots, a_m, \$, \llcorner\}$$

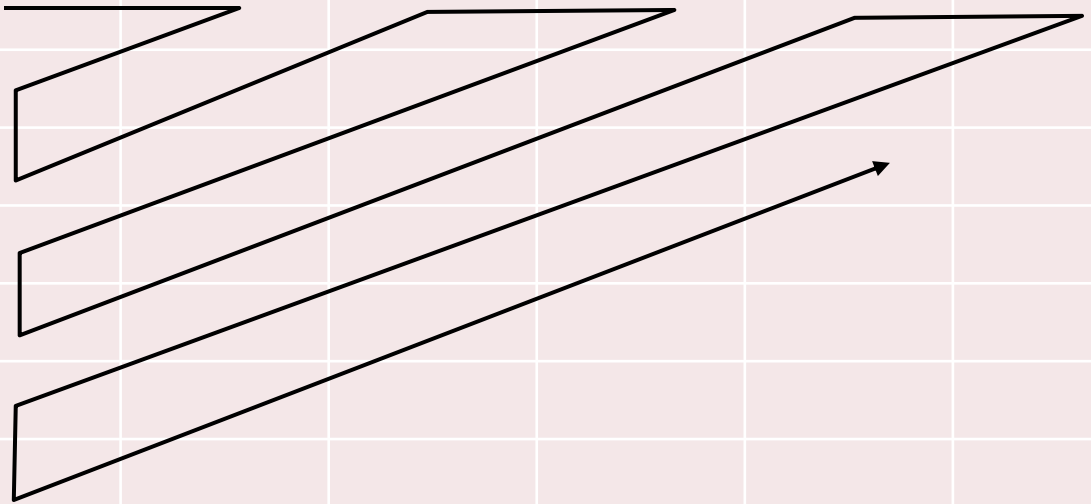
Fixiert man m und k, gibt es nur endlich viele Turing-Maschinen. Die Anzahl an Transitionen kann durch folgenden Wert beschränkt werden:

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

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

Diagonalverfahren

k\m	0	1	2	3	4	...
0						
1						
2						
3						
4						
5						
...						



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

Nicht-berechnbare Funktionen

Lemma 3.12
Es seien Σ_1, Σ_2 beliebige Alphabete. Es gibt nicht-berechenbare Funktionen $f: \Sigma_1^* \to \Sigma_2^*$.

Beweis:
Seien $f_0, f_1, \ldots$ 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	*undef*	2	4	0	*undef*	...
1	5	1	4	100	0	1	...
2	106	0	5	105	0	4	...
3	1	2	*undef*	0	0	9	...
4	17	3	3	115	0	16	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Kapitel 3.2 – Entscheidbarkeit

Entscheidbarkeit

Okay, es gibt berechenbare und nicht-berechenbare Funktionen.

Wie sieht das mit Sprachen aus?

Können wir immer entscheiden, ob ein Wort zu einer Sprache gehört?

(Semi-)Entscheidbarkeit

Definition 3.14

Eine Menge $A \subseteq \Sigma^*$ ist **entscheidbar**, wenn die totale charakteristische Funktion χ_A von A mit

$$x_A\colon \Sigma^* \to \{0,1\}$$
$$w \mapsto \begin{cases} 1, & \text{falls } w \in A \\ 0, & \text{sonst} \end{cases}$$

berechenbar ist.

Eine Menge A ist **semi-entscheidbar**, wenn die partielle charakteristische Funktion χ'_A von A mit

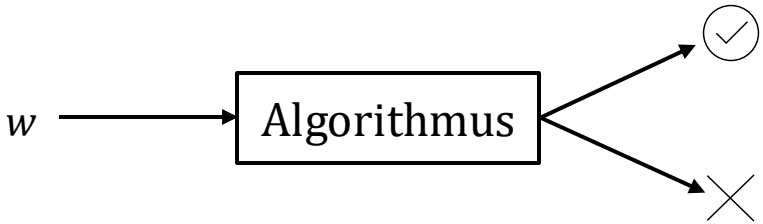
$$x'_A\colon \Sigma^* \to_p \{1\}$$
$$w \mapsto \begin{cases} 1, & \text{falls } w \in A \\ \text{undef}, & \text{sonst} \end{cases}$$

berechenbar ist.

Wortproblem

Sprachen werden in der Literatur oft mit Entscheidungsproblemen identifiziert:

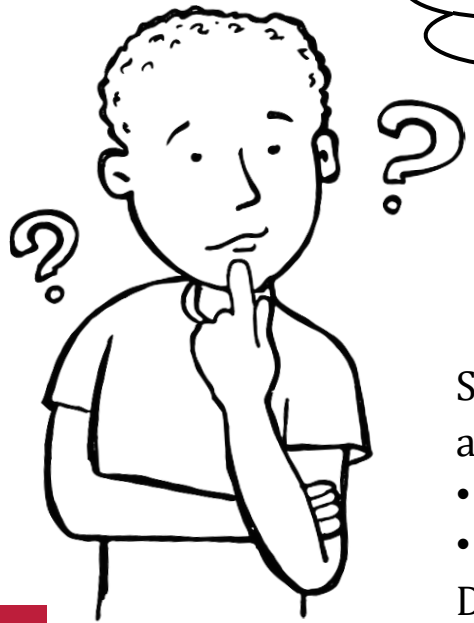
Wortproblem zu A:
Gegeben: Wort $w \in \Sigma^*$
Frage: Ist w ein Element von A?



Entscheidbar

Semi-Entscheidbar

Zusammenhang zu Turing-Maschinen



Turing-Maschinen sind zunächst nur Semi-Entscheider!

Können wir TMs etwas anpassen, dass sie auch „Entscheider“ werden können?

Statt $(Q, \Sigma, \Gamma, q_0, \delta, Q_F)$ benutze die Form $\left(Q, \Sigma, \Gamma, q_0, \delta, q_{acc}, q_{rej}\right)$, also jeweils:

- Ein eindeutiger **akzeptierender Zustand** q_{acc}.
- Ein eindeutiger **abweisender Zustand** q_{rej}.

Diese nennen wir auch **Haltezustände**.

Haltezustände

Diese Definition von TMs ist nicht schwächer!

Diese TMs **halten** in einer akzeptierenden oder abweisenden Konfiguration.

Diese TMs können also:

1. Nach endlich vielen Schritten das Wort akzeptieren.
2. Nach endlich vielen Schritten das Wort abweisen.
3. Unendlich lange laufen, ohne eine Haltekonfiguration zu erreichen – sie loopen.

Bemerkung zu NTMs:
Wir können auch in Zuständen stecken bleiben!
Es reicht aber, wenn es einen Pfad zum akzeptierenden Zustand gibt, um das Wort zu akzeptieren.

Entscheider

Proposition 3.16
Eine Menge $A \subseteq \Sigma^*$ ist genau dann **semi-entscheidbar**, wenn es eine Turing-Maschine M mit $A = \mathcal{L}(M)$ gibt.

Definition 3.17
Wir nennen eine Turing-Maschine $M = (Q, \Sigma, \Gamma, q_0, \delta, q_{acc}, q_{rej})$ **total** oder einen **Entscheider**, wenn jede Berechnung von M zu jeder Eingabe x nach endlich vielen Schritten hält.

Proposition 3.18
Eine Menge $A \subseteq \Sigma^*$ ist genau dann **entscheidbar**, wenn es einen Entscheider M mit $A = \mathcal{L}(M)$ gibt.

Entscheidbare Mengen

Wir sehen später: Das ist ein unentscheidbare Problem!

Bemerkung 3.19

Bereits bekannte Theoreme und Lemma gelten auch für Entscheider:

- Zu jedem nicht-deterministischen Entscheider M existiert ein deterministischer Entscheider M' mit $\mathcal{L}(M) = \mathcal{L}(M')$.
- Zu jedem Mehr-Band-Entscheider M_k existiert ein Ein-Band-Entscheider M' mit $\mathcal{L}(M_k) = \mathcal{L}(M')$.
- Zu jedem Entscheider $M_{\leftrightarrow}$ mit beidseitig unendlichem band existiert ein Entscheider M' mir recths unendlichem Band und $\mathcal{L}(M_{\leftrightarrow}) = \mathcal{L}(M')$.

Hilfssatz für entscheidbare Sprachen

Lemma 3.20
Jede konstextsensitive Sprache $\mathcal{L}(G)$ ist entscheidbar.

Beweis: Siehe Kapitel 1.

Hilfssatz für entscheidbare Sprachen

Theorem 3.21
Eine Sprache $A \subseteq \Sigma^*$ ist genau dann entscheidbar, wenn A und $\bar{A}$ semi-entscheidbar sind.

"⇒": Klar.
"⇐":
Sei M_A eine TM (Semi-Entscheider) für A, und $M_{\bar{A}}$ eine TM für $\bar{A}$.
Betrachte folgenden Algorithmus:

Eingabe: $w \in \Sigma^*$
For i = 1, 2, 3…
  if (M_A akzeptiert Eingabe w in höchstens i Schritten) then return 1

  if ($M_{\bar{A}}$ akzeptiert Eingabe w in höchstens i Schritten) then return 0

Dies lässt sich als TM / Entscheider verpacken!

Zusammenhang der Terminologie

Rekursiv aufzählbare Sprachen

Charakteristische Funktion

semi-entscheidbar

Gibt es konkrete unentscheidbare Sprachen?