Berechenbarkeit und Komplexität
8. Vorlesung

Prof. Dr. Dietrich Kuske

FG Automaten und Logik, TU Ilmenau

Sommersemester 2023

Satz von Rice

Das nächste Resultat zeigt, daß jede Eigenschaft der von einer Turing-Maschine berechneten Funktion unentscheidbar ist.

Das bedeutet, es gibt keine Methode, mit der man für alle Turing-Maschinen verläßliche Aussagen über die von ihnen berechneten Funktionen machen kann.

Lemma

Sei $\mathcal{R}$ die Klasse aller Turing-berechenbaren Funktionen $\{0,1\}^* \to \{0,1\}^*$, Ω die nirgendwo definierte Funktion und sei $\mathcal{S} \subseteq \mathcal{R}$ mit $\Omega \in \mathcal{S}$ und $\mathcal{S} \neq \mathcal{R}$. Dann ist die Sprache

$$C(\mathcal{S}) = \{w \in L_{TM} \mid \varphi_w \in \mathcal{S}\}$$

unentscheidbar.

Beweis:
Da $\mathcal{S} \neq \mathcal{R}$ gilt, gibt es eine Funktion $q \in \mathcal{R} \setminus \mathcal{S}$ (insbes. $q \neq \Omega$). Sei Q eine TM, die die Funktion q berechnet.

Für $w \in \{0,1\}^*$ definieren wir eine Funktion $g_w \colon \{0,1\}^* \to \{0,1\}^*$ wie folgt:

$$g_w(y) = \begin{cases} q(y) & \text{falls } w \in L_{TM} \text{ und } \varphi_w(\varepsilon), q(y) \text{ sind definiert} \\ \text{undef.} & \text{sonst.} \end{cases}$$

Es gilt

$$g_w = \begin{cases} q & \text{falls } w \in L_{TM} \text{ und } M_w \text{ hält bei Eingabe von } \varepsilon \\ & \text{d.h. } w \in H_0 \ (H_0 \text{ ist das Halteproblem bei leerer Eingabe}) \\ \Omega & \text{sonst} \end{cases}$$

Die folgende TM $M(w)$ berechnet die Funktion g_w:

- Gilt $w \notin L_{TM}$, so besteht $M(w)$ nur aus einer Endlosschleife.
- Gilt $w \in L_{TM}$, so
  - ignoriert $M(w)$ die Eingabe y zunächst und simuliert M_w auf dem leeren Band.
  - Falls diese Simulation schließlich hält, so simuliert $M(w)$ die Maschine Q auf y.

Die totale Funktion $f: \{0,1\}^* \to \{0,1\}^*$ mit

$$f(w) = \text{Kode der Maschine } M(w)$$

ist berechenbar (Beweis ist ähnlich zu dem des Lemmas auf Folie 7.25). Es gilt also insbesondere $M(w) = M_{f(w)}$ und damit $g_w = \varphi_{f(w)}$ für alle $w \in \{0,1\}^*$.

Dann gilt:

$$\begin{aligned} w \in H_0 &\iff g_w = q \\ &\iff \varphi_{f(w)} \notin \mathcal{S} \\ &\iff f(w) \notin C(\mathcal{S}) \end{aligned}$$

Es gilt also $w \in \overline{H_0} \iff f(w) \in C(\mathcal{S})$, d.h. $\overline{H_0} \leq C(\mathcal{S})$.

Da H_0 nach dem Satz auf Folie 7.27 unentscheidbar ist, sind auch das Komplement $\overline{H_0}$ und somit $C(\mathcal{S})$ unentscheidbar. □

Satz von Rice

Sei $\mathcal{R}$ die Klasse aller Turing-berechenbaren Funktionen und sei $\mathcal{S} \subseteq \mathcal{R}$ mit $\mathcal{S} \neq \emptyset$ und $\mathcal{S} \neq \mathcal{R}$.
Dann ist die Sprache

$$C(\mathcal{S}) = \{w \in L_{TM} \mid \varphi_w \in \mathcal{S}\}$$

unentscheidbar.

Beweis:
Fall 1 $\Omega \in \mathcal{S}$: Dann ist $C(\mathcal{S})$ nach dem Lemma auf Folie 8.2 unentscheidbar.
Fall 2 $\Omega \notin \mathcal{S}$: Dann gilt $\Omega \in \mathcal{R} \setminus \mathcal{S}$ und $\mathcal{R} \setminus \mathcal{S} \neq \mathcal{R}$. Nach dem Lemma auf Folie 8.2 ist $C(\mathcal{R} \setminus \mathcal{S})$ also unentscheidbar und damit auch $C(\mathcal{S}) = L_{TM} \setminus C(\mathcal{R} \setminus \mathcal{S})$. □

Konsequenzen aus dem Satz von Rice

Folgende Probleme sind unentscheidbar:

- Konstante Funktion: $\{w \in L_{TM} \mid \varphi_w \text{ ist konstante Funktion}\}$
- Identität: $\{w \in L_{TM} \mid \varphi_w \text{ ist Identitätsfunktion}\}$
- Totale Funktion: $\{w \in L_{TM} \mid \varphi_w \text{ ist totale Funktion}\}$
- Überall undefinierte Funktion: $\{w \in L_{TM} \mid \varphi_w = \Omega\}$

Der Satz von Rice erlaubt es, Unentscheidbarkeitsresultate für die Eigenschaften der von einer Turing-Maschine berechneten Funktion zu zeigen, nicht jedoch für andere Eigenschaften einer Turing-Maschine (wie z.B. die Anzahl ihrer Zustände oder ob sie für irgendeine Eingabe hält - denn das Halten ist keine Eigenschaft der berechneten Funktion).

Konsequenz des Satzes von Rice für die Verifikation von Programmen: Kein Programm kann automatisch nichttriviale Eigenschaften von Software überprüfen.

Semi-Entscheidbarkeit

Das Halteproblem bei leerer Eingabe H_0 ist also unentscheidbar: Mit keinem Algorithmus kann festgestellt werden, ob die Maschine M_w bei leerer Eingabe anhalten wird.

Aber natürlich kann ich nach endlicher Zeit feststellen, daß die Maschine M_w bei leerer Eingaben anhält: Ich simuliere das Verhalten von M_w bei leerer Eingabe.

Das Halteproblem bei leerer Eingabe H_0 ist also „halb-“ oder „semi-entscheidbar“.

Analoges gilt für viele weitere Probleme (aber nicht für alle). Wir werden jetzt diese Eigenschaft formalisieren und dann ausführlich untersuchen.

Definition

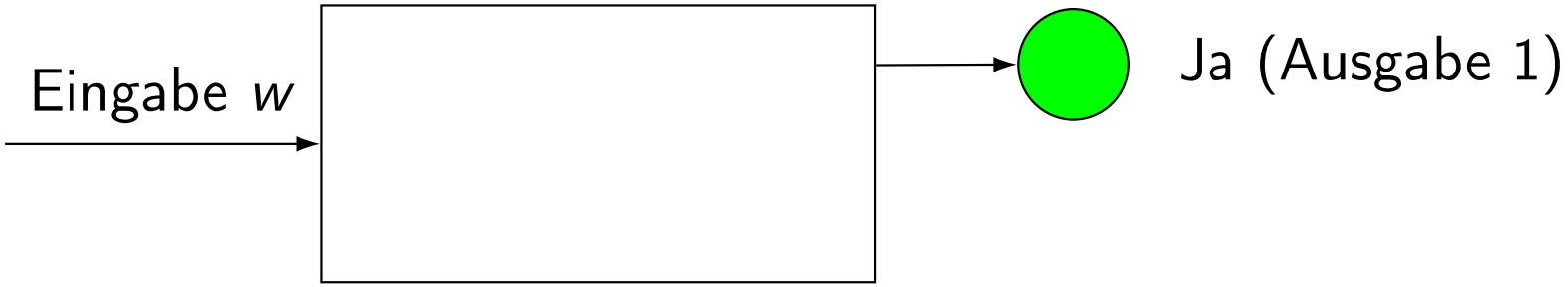
Eine Sprache $L \subseteq \Sigma^*$ heißt semi-entscheidbar, falls die „halbe“ charakteristische Funktion von L, d.h. die partielle Funktion $\chi'_L : \Sigma^* \dashrightarrow \{1\}$ mit

$$\chi'_L(w) = \begin{cases} 1 & \text{falls } w \in L \\ \text{undefiniert} & \text{falls } w \notin L \end{cases}$$

berechenbar ist.

Bei Semi-Entscheidbarkeit erlaubt man also, daß die berechnete Funktion χ'_L im negativen Fall undefiniert ist, d.h. keine Antwort zurückkommt.

Darstellung der Semi-Entscheidbarkeit an einem Maschinenmodell:



Bei jeder Eingabe rechnet die Maschine und gibt im Fall $w \in L$ nach endlicher Zeit „Ja“ aus. Falls $w \notin L$ gilt, so terminiert die Maschine nicht.

Das heißt, man kann sich nie sicher sein, ob nicht doch irgendwann „Ja“ ausgegeben wird, da die Antwortzeit der Maschine nicht beschränkt ist.

Beispiel

Sei $G = (V, \Sigma, S, P)$ eine Grammatik. Dann ist $L(G)$ semi-entscheidbar.

Beweis: Um $\chi'_{L(G)}(w)$ zu berechnen, geht eine Turing-Maschine wie folgt vor:
Sei $v_0, v_1, \ldots$ die längenlexikographische Aufzählung von $\big(V \cup \Sigma \cup \{\#\}\big)^*$.
Dann teste nacheinander, für jedes n, ob $v_n = w_0 \# w_1 \# \ldots \# w_k$ mit $w_0 = S$, $w_i \Rightarrow_G w_{i+1}$ für alle $1 \leq i < k$ und $w_k = w$. Ist dies der Fall, so terminiere mit Ausgabe 1, sonst betrachte v_{n+1}. □

Beispiel

Sei DiophGl die Menge aller multivariaten Polynome p über $\mathbb{Z}$, die eine ganzzahlige Nullstelle haben.
Die Menge DiophGl ist semi-entscheidbar.

Beweis: Um χ'_{DiophGl} zu berechnen, geht eine Turing-Maschine wie folgt vor:

1. Ist Eingabe $w \in \Sigma^*$ kein multivariates Polynom, so gehe in Endlosschleife. Sonst sei $w = p(x_1, \ldots, x_k)$.
2. Teste nacheinander für alle $m \in \mathbb{N}$, ob es Zahlen $n_1, \ldots, n_k \in \mathbb{Z}$ gibt mit $|n_i| \leq m$ und $p(n_1, n_2, \ldots, n_k) = 0$. Sobald ein solches Tupel gefunden ist, terminiere mit Ausgabe 1. □

Satz

Ein Problem $L \subseteq \Sigma^*$ ist genau dann entscheidbar, wenn sowohl L als auch $\overline{L} = \Sigma^* \setminus L$ semi-entscheidbar sind.

Beweis:

Sei zunächst L entscheidbar.

Dann ist also die charakterische Funktion χ_L berechenbar mittels einer Turing-Maschine M.

Die folgende Turing-Maschine M_L bzw. $M_{\overline{L}}$ berechnet die halbe charakteristisch Funktion χ'_L: bzw. $\chi'_{\overline{L}}$:

- M_L bzw. $M_{\overline{L}}$ simuliert zunächst die Maschine M.
- Falls M mit der Ausgabe 0 bzw. 1 terminiert, geht jedoch die TM M_L bzw. $M_{\overline{L}}$ in eine Endlosschleife über, andernfalls terminiert sie mit Ausgabe 1.

Also ist sind L und $\overline{L}$ semi-entscheidbar.

Seien umgekehrt sowohl L als auch $\overline{L}$ semi-entscheidbar.

Dann existieren Turing-Maschinen M_L und $M_{\overline{L}}$, die χ'_L bzw. $\chi'_{\overline{L}}$ berechnen.

Behauptung: Der folgende Algorithmus berechnet χ_L (woraus unmittelbar folgt, daß L entscheidbar ist).

```
input w
t := 1;
while true do
  if M_L terminiert bei Eingabe w nach t Schritten then
    return(1)
  elseif M_L̄ terminiert bei Eingabe w nach t Schritten then
    return(0)
  end
  t := t + 1
end
```

Beweis der Behauptung: Sei $w \in \Sigma^*$. Wir unterscheiden 2 Fälle:

1. $w \in L$. Dann existiert $t \in \mathbb{N}$, so daß M_L nach t Schritten terminiert. Wegen $w \notin \overline{L}$ terminiert $M_{\overline{L}}$ niemals (und insbesondere nicht vorher). Damit terminiert der Algorithmus mit der korrekten Ausgabe $1 = \chi_L(w)$.
2. $w \notin L$. Dann existiert $t \in \mathbb{N}$, so daß $M_{\overline{L}}$ nach t Schritten terminiert. Wegen $w \notin L$ terminiert M_L niemals (und insbesondere nicht vorher). Damit terminiert der Algorithmus mit der korrekten Ausgabe $0 = \chi_L(w)$.

Damit ist die Behauptung beweisen. □

Dieses letzte Argument heißt mitunter „Schwalbenschwanz-Argument“.

Die semi-entscheidbaren Sprachen können auf viele Arten charakterisiert werden, sie bilden also ein natürliches Konzept:

Satz

Sei $L \subseteq \Sigma^*$ eine nichtleere Sprache. Dann sind äquivalent:

1. L ist semi-entscheidbar
2. L wird von einer Turing-Maschine akzeptiert
3. L ist vom Typ 0 (d.h. von einer Grammatik erzeugt)
4. L ist Bild einer berechenbaren partiellen Funktion $\Sigma^* \dashrightarrow \Sigma^*$
5. L ist Bild einer berechenbaren totalen Funktion $\Sigma^* \to \Sigma^*$
6. L ist rekursiv aufzählbar
7. L ist Definitionsbereich einer berechenbaren partiellen Funktion $\Sigma^* \dashrightarrow \Sigma^*$

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine Turing-Maschine. Die von M akzeptierte Sprache ist

$$L(M) = \{w \in \Sigma^* \mid \text{es gibt akzept. Haltekonf. } k \text{ mit } z_0 w \square \vdash_M^* k\}.$$

Lemma ((1) $\Rightarrow$ (2))

Ist $L \subseteq \Sigma^*$ semi-entscheidbar, so existiert eine Turing-Maschine, die L akzeptiert.

Beweis: L semi-entscheidbar
$\Longrightarrow$ es gibt eine TM M, die χ'_L berechnet
$\Longrightarrow$ diese TM akzeptiert L. $\square$

Lemma ((2) ⇒ (3))

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ TM. Dann existiert eine Grammatik G mit $L(G) = L(M)$.

Beweisidee: Wir simulieren M rückwärts mittels der Grammatik $G = (V, \Sigma, P, S)$ mit $V = Z \cup \{S, B, \lhd, \rhd\} \cup \Gamma \setminus \Sigma$ und den folgenden Regeln für $z \in Z$, $a, b, c \in \Gamma$:

$$
\begin{aligned}
S &\to B \lhd \\
B &\to Bb \mid \rhd za \quad (\text{falls } z \in E, (z, a, N) = \delta(z, a)) \\
z'b &\to za \quad (\text{falls } (z', b, N) = \delta(z, a)) \\
bz' &\to za \quad (\text{falls } (z', b, R) = \delta(z, a)) \\
z'cb &\to cza \quad (\text{falls } (z', b, L) = \delta(z, a)) \\
\rhd &\to \rhd\square \text{ und } \rhd\square \to \rhd \\
\lhd &\to \square\lhd \text{ und } \square\lhd \to \lhd \\
\lhd &\to \varepsilon \text{ und } \rhd z_0 \to \varepsilon
\end{aligned}
$$

□

Beispiel

Nach Folie 8.12 ist DiophGl semi-entscheidbar. Da wir bereits (1) $\Rightarrow$ (3) gezeigt haben, gibt es also eine Grammatik G, die die Menge DiophGl erzeugt - ich habe keine Ahnung, wie diese aussieht (außer natürlich die aus dem Beweis gewonnene).

Lemma ((3) ⇒ (4))

Ist $G = (V, \Sigma, S, P)$ eine Grammatik, so gibt es eine berechenbare partielle Funktion $f : \Sigma^* \dashrightarrow \Sigma^*$, deren Bild $L(G)$ ist.

Beweis:

Betrachte die folgende Funktion f:

- $f(x) = w_n$ falls $x = w_0 \# w_1 \# \cdots \# w_n$ mit $w_i \in (V \cup \Sigma)^*$, $w_0 = S$, $w_i \Rightarrow_G w_{i+1}$ f.a. $0 \leq i < n$ und $w_n \in \Sigma^*$.
- Sonst ist $f(x)$ undefiniert.

Dann ist f berechenbar mit Bild $L(G)$. □

Beispiel

Betrachte die folgenden Funktion f:

$$f(x) = \begin{cases} p & \text{falls } x = p\#\text{bin}(m_1)\#\text{bin}(n_1)\#\ldots\#\text{bin}(m_k)\#\text{bin}(n_k) \\ & \text{mit } p \in \mathbb{Z}[x_1, \ldots, x_k] \\ & \text{und } p(m_1 - n_1, \ldots, m_k - n_k) = 0 \\ \text{undef.} & \text{sonst} \end{cases}$$

Dann ist f berechenbar mit Bild DiophGl.

Lemma ((4) ⇒ (5))

Sei $f: \Sigma^* \dashrightarrow \Sigma^*$ eine nichtleere berechenbare partielle Funktion. Dann existiert eine berechenbare totale Funktion $g: \Sigma^* \to \Sigma^*$, die dasselbe Bild hat wie f.

Beispiel

Sei g die folgende Funktion:

$$g(x) = \begin{cases} p & \text{falls } x = p\#\text{bin}(m_1)\#\text{bin}(n_1)\#\ldots\#\text{bin}(m_k)\#\text{bin}(n_k) \\ & \quad \text{mit } p \in \mathbb{Z}[x_1, \ldots, x_k] \ \&\ p(m_1 - n_1, \ldots, m_k - n_k) = 0 \\ x_1 & \text{sonst} \end{cases}$$

Dann ist g total und berechenbar mit Bild DiophGl.

Beweis: Seien M TM, die f berechnet und w_0 aus dem Bild von f beliebig. Definiere die Funktion g wie folgt:

- $g(x) = w$ falls $x = k_0\#k_1\#\ldots\#k_n$, wobei
  1. k_0 Startkonfiguration von M
  2. $k_i \vdash_M k_{i+1}$ für alle $1 \leq i < n$ und
  3. k_n akzeptierende Haltekonfiguration mit Ausgabe w ist.
- w_0 sonst.

Dann ist g total und berechenbar.

- Sei $f(v) = w$. Dann existiert eine erfolgreiche Berechnung $k_0 \vdash_M k_1 \vdash_M k_2 \cdots \vdash_M k_n$ von M mit $k_0 = z_0 v\Box$ und Ausgabe w. Es gilt $g(k_0\#k_1\#\ldots\#k_n) = w$.
- Sei $g(x) = w$ mit $w \neq w_0$. Dann existiert x wie oben mit $g(x) = w$. Sei $k_0 = z_0 v\Box$. Dann folgt $f(v) = w$.

Also haben f und g dasselbe Bild. □

Definition

Eine Sprache $L \subseteq \Sigma^*$ heißt rekursiv aufzählbar, falls $L = \varnothing$ oder es eine totale und berechenbare Funktion $f : \mathbb{N} \to \Sigma^*$ gibt mit

$$L = \left\{ f(n) \;\middle|\; n \in \mathbb{N} \right\} = \left\{ f(0), f(1), f(2), \ldots \right\}.$$

Bemerkungen:

- f muß nicht unbedingt injektiv sein!
- Sprechweise: die Funktion f zählt die Sprache L auf.
- man kann eine TM angeben, die nacheinander $f(0)$, $f(1)$, ... berechnet und ausgibt - die also L aufzählt.
- rek. aufz. = „effektiv aufzählbar“ = „computably enumerable“

Beispiel

Sei M TM und L die Menge der Konfigurationen, die M bei Eingabe des leeren Wortes erreicht.

Die Funktion $f: \mathbb{N} \to \Gamma^* Z \Gamma^+$ mit $f(n)$ ist die Konfiguration, die M bei Eingabe des leeren Wortes nach n Schritten erreicht, ist berechenbar und total und sie erfüllt

$$L = \left\{ f(n) \mid n \in \mathbb{N} \right\}.$$

Also ist L rekursiv aufzählbar.

Lemma ((5) ⇒ (6))

Sei $g\colon \Sigma^* \to \Sigma^*$ eine berechenbare totale Funktion. Dann ist das Bild von g rekursiv aufzählbar.

Beweis: Es gibt eine berechenbare surjektive Abbildung $c\colon \mathbb{N} \to \Sigma^*$. Dann ist $g \circ c\colon \mathbb{N} \to \Sigma^* : n \mapsto g\big(c(n)\big)$ berechenbar und total. Da c surjektiv ist, ist das Bild von $g \circ c$ gleich dem von g. Also ist das Bild von g rekursiv aufzählbar. □

Lemma ((6) ⇒ (7))

Sei $L \subseteq \Sigma^*$ rekursiv aufzählbar. Dann existiert eine berechenbare partielle Funktion $f : \Sigma^* \dashrightarrow \Sigma^*$, deren Definitionsbereich L ist.

Beweis: Ist $L = \emptyset$, so ist L der Definitionsbereich der berechenbaren Funktion Ω.

Sei nun $g : \mathbb{N} \to \Sigma^*$ berechenbar und total mit Bild L. Der folgende Algorithmus definiert eine partielle Funktion $f : \Sigma^* \dashrightarrow \Sigma^*$ mit Definitionsbereich L:

```
input w
n := 0
while g(n) ≠ w do n := n + 1 end
return 1
```

□

Lemma ((7) ⇒ (1))

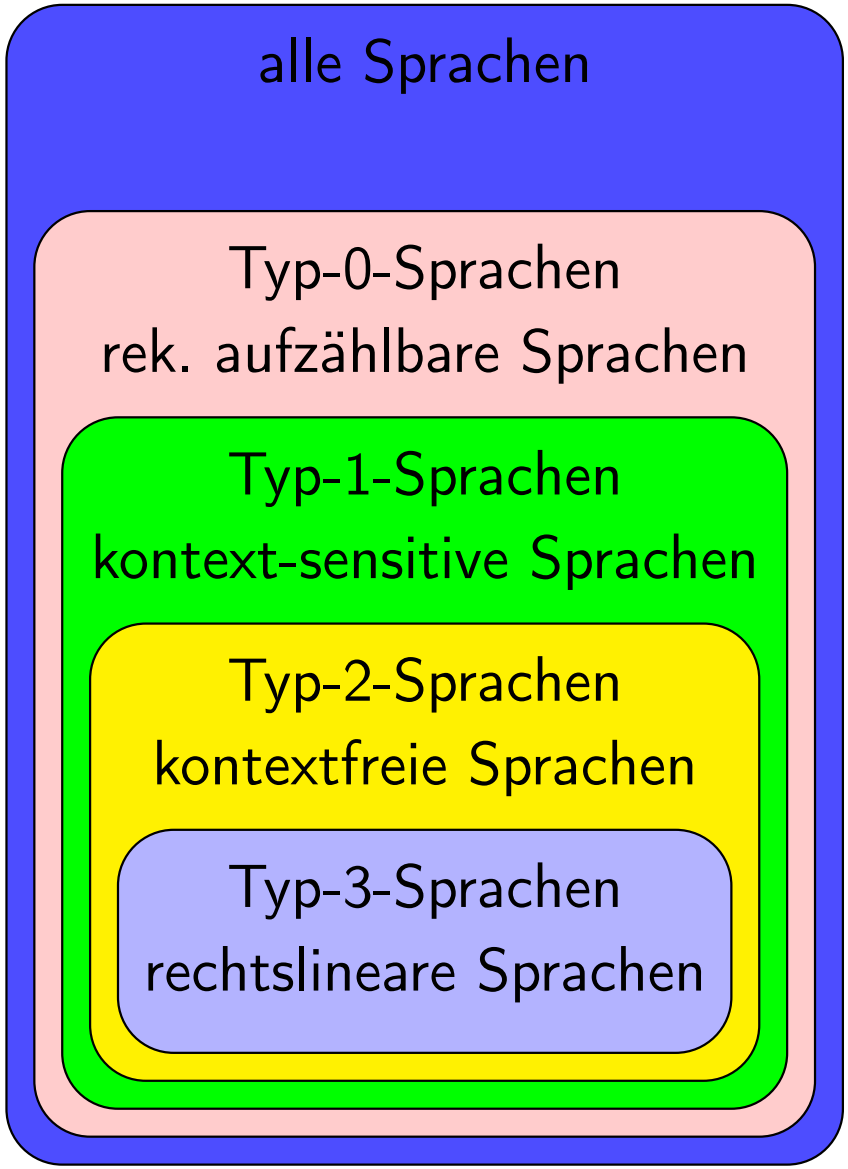
Sei $g\colon \Sigma^* \dashrightarrow \Sigma^*$ eine berechenbare partielle Funktion. Ihr Definitionsbereich L ist semi-entscheidbar.

Beweis: Sei M TM, die g berechnet. Ändere diese so, daß sie beim Anhalten den Inhalt des Bandes durch 1 ersetzt. Diese neue TM berechnet die halbe charakteristische Funktion von L, d.h. die Funktion χ'_L. □

Damit ist der Beweis des Satzes auf Folie 8.16 abgeschlossen.

Die Sprachen vom Typ 3 / 2/ 1 heißen rechtslinear / kontext-frei / kontext-sensitiv, weil sie von rechtslinearen / kontext-freien / kontext-sensitiven Grammatiken erzeugt werden.

Die Sprachen vom Typ 0 heißen rekursiv aufzählbar, weil es genau die rekursiv aufzählbaren sind.



Zusammenfassung 8. Vorlesung

in dieser Vorlesung neu

- Satz von Rice: nur triviale Eigenschaft des Verhaltens von M_w sind entscheidbar
- aber viele können „zur Hälfte“ entschieden werden
- viele, sehr verschiedene Charakterisierungen der semi-entscheidbaren Mengen

kommende Vorlesung

- eine für alle: universelle TM
- ein unentscheidbares Puzzle