2. Entscheidbarkeit, Halteproblem

Wir wollen nun zeigen, dass es keinen Algorithmus geben kann, der als Eingabe ein (beliebiges) Programm P und Daten x für P erhält und (für jedes solches Paar (P, x)!) entscheidet, ob P hält, wenn es mit Eingabe x gestartet wird.

Wir erinnern uns:

1. Eine Sprache A ist rekursiv gdw die charakteristische Funktion χ_A berechenbar ist.
2. Eine Sprache A ist rekursiv aufzählbar (r.a.) gdw die semi-charakteristische Funktion χ'_A berechenbar ist.

Definition 152

Eine Sprache $A \subseteq \Sigma^*$ heißt rekursiv auflistbar, falls es eine berechenbare Funktion $f : \mathbb{N}_0 \to \Sigma^*$ gibt, so dass

$$A = \{f(0), f(1), f(2), \ldots\}.$$

Bemerkung: Es ist nicht verlangt, dass die Auflistung in einer gewissen Reihenfolge (z.B. lexikalisch) erfolgt!

Beispiel 153

Σ^* (mit $\Sigma = \{0,1\}$) ist rekursiv auflistbar. Wir betrachten dazu etwa folgende Funktion:

$$
\begin{array}{rl}
\text{alle nullstelligen Wörter} & f(0) = \epsilon \\
\text{alle einstelligen Wörter} & \begin{cases} f(1) = 0 \\ f(2) = 1 \end{cases} \\
\text{alle zweistelligen Wörter} & \begin{cases} f(3) = 00 \\ f(4) = 01 \\ f(5) = 10 \\ f(6) = 11 \end{cases} \\
\text{alle dreistelligen Wörter} & \begin{cases} f(7) = 000 \\ \quad\vdots \end{cases}
\end{array}
$$

Beispiel 153

Eine weitere Möglichkeit, eine Funktion f anzugeben, die alle Wörter $\in \{0,1\}^*$ auflistet, ist:

$$f(n) = \text{Binärkodierung von } n+1 \text{ ohne die führende } 1$$

Also:

$$\begin{aligned} f(0) &= 1\epsilon \\ f(1) &= 10 \\ f(2) &= 11 \\ f(3) &= 100 \\ \vdots \quad & \quad \vdots \end{aligned}$$

Beispiel 153

$L_{TM} = \{w \in \{0,1\}^*;\ w$ ist Codierung einer TM$\}$ ist rekursiv auflistbar:

Wir listen $\{0,1\}^*$ rekursiv auf, prüfen jedes erzeugte Wort, ob es eine syntaktisch korrekte Codierung einer Turing-Maschine ist, und verwerfen es, falls nicht.

Wir wählen stattdessen die kanonische Auflistung von $\{0,1\}^*$ und ersetzen jedes dabei erzeugte Wort, das keine korrekte Codierung darstellt, durch den Code einer Standard-TM, die $\emptyset$ akzeptiert.

Satz 154

Eine Sprache A ist genau dann rekursiv auflistbar, wenn sie rekursiv aufzählbar (semi-entscheidbar) ist.

Beweis:

Wir zeigen zunächst „$\Rightarrow$".

Sei $f : \mathbb{N}_0 \to \Sigma^*$ eine berechenbare Funktion, die A auflistet.

Betrachte folgenden Algorithmus:

lies die Eingabe $w \in \Sigma^*$
$x := 0$
while true do
  if $w = f(x)$ **then return** („ja"); **halt fi**
  $x := x + 1$
od

Beweis:

Wir zeigen nun „$\Leftarrow$“.
Sei P eine WHILE-Programm, das die semi-charakteristische Funktion χ'_A berechnet, und sei f eine berechenbare Funktion, die Σ^* auflistet.

Betrachte folgenden Algorithmus:

lies die Eingabe $n \in \mathbb{N}_0$
$count := -1$; $k := -1$
repeat
  $k := k + 1$
  $w := f(c_1(k))$; $m := c_2(k)$
  if P hält bei Eingabe w in genau m Schritten **then**
    $count := count + 1$
until $count = n$
return w

Hier sind c_1 und c_2 die Umkehrfunktionen einer Paarfunktion. $\square$

2.2 Halteproblem

Definition 155

Unter dem speziellen Halteproblem H_s versteht man die folgende Sprache:

$$H_s = \{w \in \{0,1\}^*;\ M_w \text{ angesetzt auf } w \text{ hält}\}$$

Hierbei ist $(M_\epsilon, M_0, M_1, \ldots)$ eine berechenbare Auflistung der Turing-Maschinen.

Wir definieren weiter

Definition 156

$$L_d = \{w \in \Sigma^*;\ M_w \text{ akzeptiert } w \text{ nicht}\}$$

Satz 157

L_d ist nicht rekursiv aufzählbar.

Beweis:

Wäre L_d r.a., dann gäbe es ein w, so dass $L_d = L(M_w)$.
Dann gilt:

$$\begin{aligned} M_w \text{ akzeptiert } w \text{ nicht} \quad &\Leftrightarrow \quad w \in L_d \\ &\Leftrightarrow \quad w \in L(M_w) \\ &\Leftrightarrow \quad M_w \text{ akzeptiert } w \end{aligned}$$

$\Longrightarrow$ Widerspruch!

Korollar 158

L_d ist nicht entscheidbar.

Satz 159

H_s ist nicht entscheidbar.

Beweis:

Angenommen, es gäbe eine Turing-Maschine M, die H_s entscheidet. Indem man i.W. die Antworten von M umdreht, erhält man eine TM, die L_d entscheidet. Widerspruch!

2.3 Unentscheidbarkeit

Definition 160

Unter dem (allgemeinen) Halteproblem H versteht man die Sprache

$$H = \{\langle x, w\rangle \in \{0,1\}^*;\ M_x \text{ angesetzt auf } w \text{ hält}\}$$

Satz 161

Das Halteproblem H ist nicht entscheidbar.

Beweis:

Eine TM, die H entscheidet, könnten wir benutzen, um eine TM zu konstruieren, die H_s entscheidet.

Bemerkung: H und H_s sind beide rekursiv aufzählbar!

Definition 162

Seien $A, B \subseteq \Sigma^*$. Dann heißt A (effektiv) reduzierbar auf B gdw $\exists f : \Sigma^* \to \Sigma^*$, f total und berechenbar mit

$$(\forall w \in \Sigma^*)[w \in A \Leftrightarrow f(w) \in B]\,.$$

Wir schreiben auch

$$A \hookrightarrow_f B \text{ bzw. } A \hookrightarrow B\,.$$

bzw. manchmal

$$A \leq B \text{ oder auch } A \preceq_f B\,.$$

Ist A mittels f auf B reduzierbar, so gilt insbesondere

$$f(A) \subseteq B \text{ und } f(\bar{A}) \subseteq \bar{B}\,.$$

Satz 163

Sei $A \hookrightarrow_f B$.

(i) B *rekursiv* $\Rightarrow$ A *rekursiv.*

(ii) B *rekursiv aufzählbar* $\Rightarrow$ A *rekursiv aufzählbar.*

Beweis:

(i) $\chi_A = \chi_B \circ f$.

(ii) $\chi'_A = \chi'_B \circ f$.

Definition 164

Das Halteproblem auf leerem Band H_0 ist

$$H_0 = \{w \in \{0,1\}^*;\ M_w \text{ hält auf leerem Band}\}\,.$$

Satz 165

H_0 ist unentscheidbar (nicht rekursiv).

Beweis:

Betrachte die Abbildung f, die definiert ist durch:

$$\{0,1\}^* \ni w \mapsto f(w),$$

$f(w)$ ist die Gödelnummer einer TM, die, auf leerem Band angesetzt, zunächst $c_2(w)$ auf das Band schreibt und sich dann wie $M_{c_1(w)}$ (angesetzt auf $c_2(w)$) verhält. Falls das Band nicht leer ist, ist es unerheblich, wie sich $M_{f(w)}$ verhält.

f ist total und berechenbar.

Es gilt:
$$\begin{aligned} w \in H \quad &\Leftrightarrow M_{c_1(w)} \text{ angesetzt auf } c_2(w) \text{ hält} \\ &\Leftrightarrow M_{f(w)} \text{ hält auf leerem Band} \\ &\Leftrightarrow f(w) \in H_0 \end{aligned}$$

also $H \hookrightarrow_f H_0$ und damit H_0 unentscheidbar.

Bemerkung

Es gibt also keine allgemeine algorithmische Methode, um zu entscheiden, ob ein Programm anhält.