2.5 Halteproblem und Unentscheidbarkeit

Der Berechenbarkeitsbegriff ist auf Funktionen zugeschnitten. Wir wollen nun einen entsprechenden Begriff für Mengen einführen.

Definition 2.55 *Eine Menge $A \subseteq \Sigma^*$ heißt entscheidbar, falls die charakteristische Funktion von A, nämlich $\chi_A \colon \Sigma^* \to \{0,1\}$, berechenbar ist. Hierbei ist für alle $w \in \Sigma^*$:*

$$\chi_A(w) = \begin{cases} 1 & \textit{falls } w \in A, \\ 0 & \textit{falls } w \notin A, \end{cases}$$

Definition 2.56 *Eine Menge $A \subseteq \Sigma^*$ heißt semi-entscheidbar, falls die „halbe“ charakteristische Funktion von A, nämlich $\chi'_A \colon \Sigma^* \to \{0,1\}$, berechenbar ist. Hierbei ist für alle $w \in \Sigma^*$:*

$$\chi'_A(w) = \begin{cases} 1 & \textit{falls } w \in A, \\ \textit{nicht definiert} & \textit{falls } w \notin A, \end{cases}$$

Beide Definitionen lassen sich natürlich auch für Teilmengen $A \subseteq \mathbb{N}$ erweitern.

Bemerkung 2.57 Im Zusammenhang mit der Frage der Entscheidbarkeit werden Mengen oft auch als *Entscheidungs-Probleme* dargestellt (in der Form: gegeben – gefragt).

Der Menge

$$\{x \in \mathbb{N} \mid x \text{ gerade}\}$$

würde dann das Problem

Gegeben: $x \in \mathbb{N}$
Frage: Ist x gerade?

entsprechen.

Bildhaft gesprochen besagen die beiden Definitionen, dass im ersten Fall ein immer stoppender Algorithmus zur Verfügung steht, der das Entscheidungsproblem für A löst (siehe Abbildung 2.5).

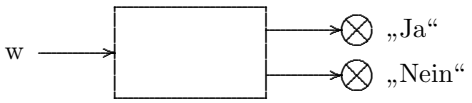

Abbildung 2.5: Veranschaulichung von entscheidbaren Mengen

Im Fall der Semi-Entscheidbarkeit sieht das Bild so aus, dass der Algorithmus *nur einen* definitiven Ausgang hat. Falls der Algorithmus also für lange Zeit nicht gestoppt hat, so ist es nicht klar, ob der „Nein“-Fall ($w \notin A$) vorliegt, oder ob er doch noch mit der Ausgabe „Ja“ stoppt (siehe Abbildung 2.6).

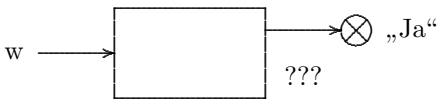

Abbildung 2.6: Veranschaulichung von semi-entscheidbaren Mengen

In diesem Fall ist die Situation also etwas unbefriedigend, aber bei vielen algorithmischen Problemen ist es das Beste, was erreichbar ist (z. B. bei Entscheidungsverfahren für die Prädikatenlogik, „Theorembeweisern“).

Der folgende Satz ist offensichtlich.

Satz 2.58 *Eine Menge A ist entscheidbar genau dann, wenn sowohl die Menge A als auch ihr Komplement $\overline{A}$ semi-entscheidbar sind.* □

Im folgenden vergleichen wir den Begriff der Semi-Entscheidbarkeit mit der *rekursiven Aufzählbarkeit*.

Definition 2.59 *Eine Menge $A \subseteq \Sigma^*$ heißt rekursiv aufzählbar, falls $A = \emptyset$ oder falls eine totale und berechenbare Funktion $f : \mathbb{N} \to \Sigma^*$ gibt, so dass*

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

gilt. Sprechweise: f zählt A auf. Man beachte, dass $f(i) = f(j)$ zulässig ist.

Wir bringen den folgenden Satz wiederum ohne Beweis, der geneigte Leser möge ihn in der Literatur nachlesen.

Satz 2.60 *Eine Menge ist rekursiv aufzählbar genau dann, wenn sie semi-entscheidbar ist.* □

Zusammenfassend können wir folgendes konstatieren.

Folgerung 2.61 *Sei $A \subseteq \Sigma^*$ eine Menge. Dann sind folgende Aussagen äquivalent.*

(i) *A ist rekursiv aufzählbar.*

(ii) *A ist semi-entscheidbar.*

(iii) *χ'_A ist berechenbar.*

(iv) *A ist Definitionsbereich einer berechenbaren Funktion.*

(v) *A ist Wertebereich einer berechenbaren Funktion.* □

Der Begriff der *Abzählbarkeit* von Mengen kann (siehe Definitionen 1.54 und 1.55) auch so definiert werden, dass er dem der rekursiven Aufzählbarkeit – bis auf einen kleinen, aber wichtigen Unterschied – ähnlich sieht:

Definition 2.62 *Eine Menge A heißt abzählbar, falls $A = \emptyset$ oder falls es eine Funktion f gibt, so dass*

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

gilt.

Der Unterschied ist der, dass hier nicht die *Berechenbarkeit* der Funktion f verlangt wird. Er wird bei folgendem Beispiel klar: Jede Teilmenge A' einer abzählbaren Menge

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

ist wieder abzählbar. Sei etwa $a \in A' \neq \emptyset$ ein festgehaltenes Element. Wenn wir

$$g(n) = \begin{cases} f(n) & \text{falls } f(n) \in A', \\ a & \text{sonst.} \end{cases}$$

setzen, so ist g sicher eine wohl-definierte (aber nicht notwendigerweise berechenbare) Funktion. Es gilt nun

$$A' = \{g(0), g(1), g(2), \dots\}$$

Nicht jede Teilmenge einer rekursiv aufzählbaren Menge muss dagegen wieder rekursiv aufzählbar sein.

Wir wollen nun ein paar nicht-entscheidbare Probleme kennenlernen. Bei den ersten dieser Probleme sollen Turingmaschinen selbst (in geeignet codierter Form) als Eingaben vorkommen. Wir müssen uns also kurz darum kümmern, wie man Turingmaschinen als Wort über $\{0,1\}$ schreiben kann.

Zunächst nehmen wir an, dass die Elemente von Γ und Z durchnummeriert sind, also

$$\Gamma = \{a_0, a_1, \ldots, a_k\} \qquad \text{und}$$
$$Z = \{z_0, z_1, \ldots, z_\ell\},$$

wobei festgelegt sein soll, welche Nummern die Symbole $\square, 0, 1, \#$ und die Start- und Endzustände erhalten. Jeder δ-Regel der Form

$$\delta(z_i, a_j) = (z_{i'}, a_{j'}, y)$$

ordnen wir das Wort

$$w_{i,j,i',j',y} = \#\#\text{bin}(i)\#\text{bin}(j)\#\text{bin}(i')\#\text{bin}(j')\#\text{bin}(m)$$

zu, wobei

$$m = \begin{cases} 0 & \text{falls } y = L, \\ 1 & \text{falls } y = R, \\ 2 & \text{falls } y = N. \end{cases}$$

Alle diese zu δ gehörenden Wörter schreiben wir nun in beliebiger Reihenfolge hintereinander und erhalten – als Zwischenschritt – einen Code der zugrundeliegenden Turingmaschine über dem Alphabet $\{0, 1, \#\}$.

Jedem solchen Wert können wir nun noch ein Wort über $\{0,1\}$ zuordnen, indem wir noch folgende Codierung vornehmen

$$0 \mapsto 00,$$
$$1 \mapsto 01,$$
$$\# \mapsto 11.$$

Es ist klar, dass auf diese Weise nicht jedes Wort in $\{0,1\}^*$ ein sinnvoller Code einer Turingmaschine ist. Sei aber M_0 irgendeine beliebige feste Turingmaschine, dann können wir *für jedes* $w \in \{0,1\}^*$ festlegen, dass M_w eine bestimmte Turingmaschine bezeichnet, nämlich

$$M_w = \begin{cases} M & \text{falls } w \text{ Codewort von } M \text{ ist}, \\ M_0 & \text{sonst}. \end{cases}$$

Definition 2.63 *Unter dem speziellen Halteproblem für Turingmaschinen verstehen wir die Menge*

$$K = \{w \in \{0,1\}^* \mid M_w \textit{ angesetzt auf } w \textit{ hält}\}.$$

Satz 2.64 *Das spezielle Halteproblem für Turingmaschinen ist nicht entscheidbar.*

Beweis. Angenommen, K ist entscheidbar. Dann ist χ_K berechenbar mittels einer Turingmaschine M. Diese Maschine M könnte nun leicht zu einer Turingmaschine M' umgebaut werden, die durch Abbildung 2.7 definiert ist.

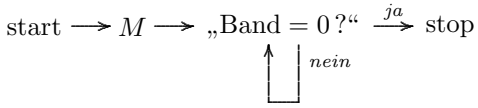

Abbildung 2.7: Die Turingmaschine M' aus dem Beweis zu Satz 2.64

Das heißt, M' stoppt genau dann, wenn M den Wert 0 ausgeben würde. Falls M den Wert 1 ausgibt, gerät M' in eine Endlosschleife. Sei w' ein Codewort der Maschine M'. Nun gilt

M' angesetzt auf w' hält
- $\Leftrightarrow$ M angesetzt auf w' gibt 0 aus (wegen Definition von M'),
- $\Leftrightarrow$ $\chi_K(w') = 0$ (da M die Menge K entscheidet),
- $\Leftrightarrow$ $w' \notin K$ (wegen Definition von χ_K),
- $\Leftrightarrow$ $M_{w'}$ angesetzt auf w' hält nicht (wegen Definition von K),
- $\Leftrightarrow$ M' angesetzt auf w' hält nicht (da w' Code von M' ist).

Dieser Widerspruch beweist, dass die Eingangsannahme falsch war, also ist K nicht entscheidbar. □

Dieses spezielle Halteproblem für Turingmaschinen kann zum (allgemeinen) *Halteproblem für Turingmaschinen* verallgemeinert werden in dem Sinne, dass wir als Eingabe eine beliebige Turingmaschine und ein beliebiges Wort zulassen und fragen, ob die Turingmaschine bei dieser Eingabe in einen Stopzustand kommt, also

Definition 2.65 *Das Halteproblem für Turingmaschinen ist die Menge*

$$H = \{w\#x \mid M_w \text{ angesetzt auf } x \text{ hält}\}.$$

Wir geben hier ohne Beweis an, dass auch dieses Problem nicht entscheidbar ist, wobei der Beweis so geführt werden kann, dass die Unentscheidbarkeit des speziellen Halteproblems für Turingmaschinen verwendet wird.

Satz 2.66 *Das Halteproblem für Turingmaschinen (H) ist nicht entscheidbar.* □

Dieses Halteproblem wiederum kann natürlich auch auf für andere Berechenbarkeitsmodelle formuliert werden, also z. B. für WHILE-Programme und GOTO-Programme. Wegen der auch dort geltenden Unentscheidbarkeit bedeutet das, verallgemeinert auf eine beliebige höhere Programmiersprache: Es ist nicht berechenbar, ob ein Computerprogramm bei einer gegebenen Eingabe hält oder nicht!

Ein weiteres Problem, welches unentscheidbar ist, ist das sogenannte X. Hilbertsche[6] Problem:

Definition 2.67 *Das X. Hilbertsche Problem ist definiert durch:*

Gegeben: $n \in \mathbb{N}$, *ein Polynom* $p(x_1, \ldots, x_n)$ *in* n *Unbekannten,*
Frage: *Besitzt* p *ganzzahlige Lösungen?*

Satz 2.68 *Das X. Hilbertsche Problem ist nicht entscheidbar.* □

Kommen wir nun zu einem Problem, das als *Postsches*[7] *Korrespondenzproblem* (PKP) bezeichnet wird.

[6] David Hilbert (1862–1943), deutscher Mathematiker, formulierte dieses Problem neben anderen auf dem Mathematikerkongress 1900 in Paris.
[7] Emil Leon Post (1897-1954, amerikanischer Mathematiker.

Definition 2.69 *Das Postsche Korrespondenzproblem ist definert durch:*

Gegeben: *Alphabet* A, $k \in \mathbb{N}$ *sowie die Folge von Wortpaaren* $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ *mit* $x_i, y_i \in A^+$ *für* $1 \leq i \leq k$.

Frage: *Gibt es eine Folge von Indizes* $i_1, i_2, \ldots, i_n$ *mit* $i_j \in \{1, 2, \ldots, k\}$ *für* $1 \leq j \leq n$, $n \in \mathbb{N}$, *so dass* $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$ *gilt?*

Beispiel 2.70 Das Korrespondenzproblem

$$K = ((1, 101), (10, 00), 011, 11)),$$

also

$$\begin{array}{lll} x_1 = 1 & x_2 = 10 & x_3 = 011 \\ y_1 = 101 & y_2 = 00 & y_3 = 11 \end{array}$$

besitzt die Lösung $(1, 3, 2, 3)$, denn es gilt

$$x_1 x_3 x_2 x_3 = 101110011 = y_1 y_3 y_2 y_3.$$

Das Postsche Korrespondenzproblem besitzt ein hohes Maß an „Komplexität“, wie das folgende harmlos aussehende Beispiel zeigt.

Beispiel 2.71 Gegeben ist folgende Belegung des PKP:

$$\begin{array}{llll} x_1 = 001 & x_2 = 01 & x_3 = 01 & y_4 = 10 \\ y_1 = 0 & y_2 = 011 & y_3 = 101 & y_4 = 001. \end{array}$$

Dieses Problem besitzt eine Lösung, aber die kürzeste Lösung besteht aus 66 Indizes, nämlich

$$2, 4, 3, 4, 4, 2, 1, 2, 4, 3, 4, 3, 4, 4, 3, 4, 4, 2, 1, 4, 4, 2, 1, 3, 4, 1, 1, 3, 4, 4, 4, 2, 1,$$
$$2, 1, 1, 1, 3, 4, 3, 4, 1, 1, 1, 4, 4, 2, 1, 4, 1, 1, 3, 4, 1, 1, 3, 1, 1, 3, 1, 2, 1, 4, 1, 1, 3$$

Der naive Algorithmus, der bei gegebener Eingabe (x_1, y_1), (x_2, y_2), $\ldots$, (x_k, y_k) systematisch alle immer länger werdende Indexfolgen $i_1, i_2, \ldots, i_n$ daraufhin untersucht, ob sie eine Lösung darstellen und im positiven Fall stoppt, demonstriert, dass das PKP semi-entscheidbar ist. Bei Eingaben, die keine Lösung besitzen stoppt das Verfahren allerdings nicht. Man kann beweisen, dass es kein Verfahren gibt, das das PKP entscheidet, also

Satz 2.72 *Das Postsche Korrespondenzproblem ist nicht entscheidbar.* □