
4. Turingmaschinen

Der Vorschlag, Algorithmen durch Angabe geeigneter Maschinen zu formalisieren, geht auf den Englischen Mathematiker A.M. Turing (1937) zurück. Dieser ließ sich bei seinen Überlegungen davon leiten, wie ein Mensch einen Algorithmus ausführt. Er beobachtete, dass die elementaren Schritte, wenn man sie nur fein genug zerlegt, bei allen Algorithmen gleich sind und aus einfachen Zeichenmanipulationen bestehen. Ausgeführt werden diese auf einem Blatt Papier, lassen sich aber auch linearisieren, sodass als Speichermedium ein in Felder aufgeteiltes Band ausreicht, wobei jedes Feld eines der benutzten Symbole aufnehmen kann. Formal gesehen, besteht die gespeicherte Zwischeninformation also aus einem Wort über einem festen Alphabet. Da wir uns bei Ein- und Ausgaben auf Wörter beschränken können, ergibt sich hiermit die folgende intuitive Beschreibung einer Turing-Basismaschine zur Berechnung einer n -stelligen Funktion $f: (\Sigma^*)^n \rightarrow T^*$:

Der *Speicher* der Maschine besteht aus einem nach beiden Seiten unbeschränkten (*Turing-*) *Band*, das in unendlich viele *Felder* eingeteilt ist. Jedes Feld kann einen Buchstaben aus dem *Bandalphabet* Γ der Maschine aufnehmen. Dabei sind zu jedem Zeitpunkt nur endlich viele Felder belegt. Der Zugriff auf das Speicherband erfolgt durch einen *Lese-Schreibkopf* von der Größe eines Feldes. Das Feld, auf das der Kopf zeigt, heißt das *Arbeitsfeld*. Dieses Feld kann eingelesen und neu beschriftet werden. Weiter kann der Kopf um ein Feld nach links oder rechts verlegt werden. Durch Wiederholung dieser elementaren Operationen kann also jede Stelle des Bandes besucht, die dort stehende Information gelesen und gegebenenfalls ersetzt oder ergänzt werden.

Die *Eingabe* erfolgt dadurch, dass die n Eingabewörter jeweils durch ein leeres Feld getrennt auf das ansonsten leere Band geschrieben werden. (Dies erfordert natürlich, dass $\Sigma \subseteq \Gamma$ gilt.) Der Lese-Schreibkopf wird auf das leere Feld vor der ersten Eingabe gesetzt. Die *Ausgabe* erfolgt entsprechend: Nach Beendigung der Rechnung wird das längste Wort, das nur aus Buchstaben aus T besteht und direkt rechts an das Arbeitsfeld anschließt, ausgegeben ($T \subseteq \Gamma$).

In der folgenden Definition formalisieren wir dieses Konzept als Mathematische Maschine, wobei Folgendes zu beachten ist: Wir erweitern das Bandalphabet um ein Symbol b (*Blank* oder *Leerzeichen*), das in leere Felder geschrieben wird. Die Felder des Arbeitsbandes nummerieren wir mit ganzen Zahlen, sodass eine endliche Bandschrift als Funktion $f: \mathbb{Z} \rightarrow \Gamma$ aufgefasst werden kann, wobei $f(z)$ die Inschrift des Feldes z ist. Die Endlichkeitsbedingung bedeutet dann, dass $f(z) = b$ für fast alle – d.h. alle bis auf endlich viele – $z \in \mathbb{Z}$. Das Lesen modellieren wir durch Tests. Eine elementare Operation besteht aus dem Neube- (d.h. Über-) drucken des Arbeitsfeldes oder einer Kopfbewegung.

4.1 DEFINITION. Seien Σ, T, Γ Alphabete mit $\Sigma \cup T \subseteq \Gamma$, wobei b der erste Buchstabe von Γ sei und dieser in Σ und T nicht vorkomme, und sei $n \geq 1$. Die *Turing-Basismaschine mit Bandalphabet Γ zur Berechnung n -stelliger Funktionen von Σ^* nach*

T^* ist die mathematische Basismaschine

$$TB(\Sigma, T, \Gamma, n) = (I, O, S, \text{in}, \text{out}, \text{OPER}, \text{TEST})$$

wobei

- $I = (\Sigma^*)^n$
- $O = T^*$
- $S = BI \times \mathbb{Z}$ wobei

$$BI = \{f : \mathbb{Z} \rightarrow \Gamma \mid f(z) = b \text{ für fast alle } z \in \mathbb{Z}\}$$

Für $s = (f, z) \in S$ ist f die *Bandinschrift* und z die *Position des Arbeitsfeldes* der Speicherbelegung s .

- $\text{in} : (\Sigma^*)^n \rightarrow BI \times \mathbb{Z}$ ist gegeben durch $\text{in}(w_1, \dots, w_n) = (f_{w_1, \dots, w_n}, 0)$, wobei $f_{w_1, \dots, w_n}(z) = w_m(p)$, falls $z = \sum_{i=1}^{m-1} (|w_i| + 1) + p + 1$ mit $1 \leq m \leq n$ und $0 \leq p < |w_m|$; und $f_{w_1, \dots, w_n}(z) = b$ sonst.
- $\text{out} : BI \times \mathbb{Z} \rightarrow T^*$ ist gegeben durch $\text{out}(f, z) = f(z+1) \dots f(z+m-1)$, wobei m die kleinste Zahl ≥ 1 ist, sodass $f(z+m) \notin T$.
- $\text{OPER} = \bar{\Gamma} \cup \{R, L, S\}$, wobei $\bar{\Gamma} = \{\bar{a} : a \in \Gamma\}$. Hierbei ist (für $a \in \Gamma$) $\bar{a}(f, z) = (f_{(a,z)}, z)$ wobei

$$f_{(a,z)}(x) = \begin{cases} a & \text{falls } x = z \\ f(x) & \text{sonst} \end{cases}$$

und (für $B \in \{R, L, S\}$) ist $B(f, z) = (f, z')$ wobei

$$z' = \begin{cases} z-1 & \text{falls } B = L \\ z & \text{falls } B = S \\ z+1 & \text{falls } B = R \end{cases}$$

Zur Vereinfachung der Notation schreiben wir in der Regel $a(f, z)$ statt $\bar{a}(f, z)$.

- $\text{TEST} = \tilde{\Gamma}$, wobei $\tilde{\Gamma} = \{t_a : a \in \Gamma\}$ und $t_a(f, z) = 1$ g.d.w. $f(z) = a$.

Die Turing-Basismaschinen unterscheiden sich durch die Bandalphabet und den Typ der zu berechnenden Funktionen (Stelligkeit sowie Ein- und Ausgabealphabet). Eine Turingmaschine ist solch eine Basismaschine mit zugehörigem (deterministischem) Programm.

4.2 DEFINITION. Eine *Turingmaschine* (TM) $M = (B, P)$ ist eine Turing-Basismaschine B zusammen mit einem B -Programm P .

Die im letzten Abschnitt für beliebige mathematische Maschinen M eingeführten Begriffe können wir nun auf die Turingmaschinen anwenden. Insbesondere erhalten wir:

4.3 DEFINITION. Eine (partielle) Funktion ϕ heißt (*partiell*) *Turing-berechenbar*, falls es eine Turingmaschine M gibt, die ϕ berechnet. Entsprechend ist eine Sprache L *Turing-entscheidbar* (*Turing-aufzählbar*), wenn es eine Turingmaschine M gibt, die c_L (χ_L) berechnet.

Man geht allgemein davon aus, dass alle Algorithmen – bei geeigneter Darstellung von Ein- und Ausgaben durch Wörter – durch Turingmaschinen simuliert werden können. Insbesondere liefern daher diese Maschinen eine Formalisierung des Berechenbarkeitsbegriffs über den natürlichen Zahlen (oder über Wörtern). Diese Annahme ist als *Churchsche These* oder *Church-Turing-These* bekannt:

Church-Turing-These: Eine (partielle) Funktion $f : \mathbb{N}^n \rightarrow \mathbb{N}$ ist genau dann im intuitiven Sinne (partiell) berechenbar, wenn sie (partiell) Turing-berechenbar ist.

Diese These ist kein mathematischer Satz und damit auch nicht (im mathematischen Sinn) beweisbar: Hier wird ein intuitives Konzept einem präzisen mathematischen Konzept gegenübergestellt. Man kann jedoch gute Argumente angeben, die es glaubhaft machen, dass der intuitive Berechenbarkeitsbegriff durch die Turing-Berechenbarkeit adäquat formalisiert wird. Hier ist zunächst die Erfahrungstatsache zu nennen, dass sich kein Algorithmus bislang einer kanonischen Übersetzung in eine Turingmaschine widersetzt. Weiter haben sich sehr unterschiedliche Ansätze, den Berechenbarkeitsbegriff zu formalisieren, zu dem Ansatz von Turing äquivalent erwiesen.

Wir werden hierzu in den folgenden Abschnitten einige Beispiele betrachten. In den nächsten beiden Abschnitten führen wir zunächst Varianten des Turingmaschinen-Konzepts ein, die zu dem Grundkonzept äquivalent sind und damit dessen Robustheit demonstrieren.