2.6 Universelle Turingmaschinen

Die bislang beschriebenen DTMs sind *special purpose computers*, d. h. sie können nur ein Problem lösen. Das entspricht der historischen Entwicklung, denn erste Anwendungen von Computern waren special-purpose-Anwendungen: Ballistikprobleme in den USA, Dechiffrieren des deutschen ENIGMA-Codes in England, Flügelvermessung der V1- und V2-Rakete in Deutschland (Zuse).

Erst später wurden programmierbare (auch universell genannte) Rechner entwickelt. Bei den programmierbaren Rechnern besteht die Eingabe aus einem Programm und einer Eingabe für das Programm. Dabei ist das Programm eine Beschreibung einer (virtuellen) special purpose Maschine.

Um programmierbare oder universelle Rechner durch Turingmaschinen zu formalisieren, führen wir zunächst eine kanonische Beschreibung für Turingmaschinen ein.

2.6.1 Beschreibung einer Turingmaschine - Gödelnummern

Wir beschränken uns auf Turingmaschinen mit Eingabealphabet $\Sigma = \{0, 1\}$ und Bandalphabet $\Gamma = \{0, 1, \triangleright, \sqcup\}$. In einem Exkurs am Ende dieses Abschnitts zeigen wir, dass man damit alle Turingmaschinen erfasst, wenn andere Alphabete geeignet codiert werden. Weiter beschränken wir uns auf 1-Band Turingmaschinen. Mehrband Turingmaschinen können durch diese simuliert werden (siehe Satz 2.3.2).

Definition 2.6.1 *Sei M eine 1-Band-DTM mit*

$$Q = \{q_0, \ldots, q_n\}, q_{\text{accept}} = q_{n-1}, q_{\text{reject}} = q_n, \textit{Startzustand } s = q_0.$$

Sei $X_1 \hat{=} 0, X_2 \hat{=} 1, X_3 \hat{=} \sqcup, X_4 \hat{=} \triangleright, D_1 \hat{=} L, D_2 \hat{=} R$. Wir kodieren einen Eintrag

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

der Funktionstabelle der Übergangsfunktion δ durch

$$0^{i+1}10^j10^{k+1}10^l10^m.$$

*Wir nummerieren die Einträge der Tabelle für δ mit den Zahlen 1 bis $(n-1)|\Gamma| = 4(n-1)$ durch. Die Nummerierung erfolgt von links nach rechts und beginnt in der ersten Zeile. Es gibt zwar $n+1$ Zustände, aber für den akzeptierenden Zustand q_{accept} und den ablehnenden Zustand q_{reject} gibt es nichts zu codieren. $Code_t$ sei die Codierung des t-ten Eintrags, $1 \leq t \leq 4(n-1)$. Die **Gödelnummer** von M ist*

$$\langle M \rangle = 111Code_1 11Code_2 11Code_3 \ldots 11Code_g 111,$$

wobei $g = 4(n-1)$.

Die Bezeichnung *Gödelnummer* geht auf den deutschen Mathematiker Kurt Gödel zurück und ist motiviert durch die Tatsache, dass wir die obige 0-1-Folge auch als Binärdarstellung einer Zahl auffassen können. Sie sollten sich davon überzeugen, dass aus der Gödelnummer einer Turingmaschine ihre Übergangsfunktion rekonstruiert werden kann. Hierzu überlegen Sie sich insbesondere, was zwei aufeinanderfolgende Einsen bedeuten.

Beispiel: Wir betrachten eine Turingmaschine M, die die Sprache

$$L = \{w \in \{0,1\}^* \mid w \text{ enthält } \geq 2 \text{ Einsen}\}$$

entscheidet. Die DTM M hat Zustände $q_0, q_1, q_2 = q_{\text{accept}}, q_3 = q_{\text{reject}}$. Die Tabelle der Übergangsfunktion δ sieht folgendermaßen aus.

δ	0	1	$\sqcup$	$\triangleright$
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_3, \sqcup, L)$	$(q_0, \triangleright, R)$
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	$(q_3, \sqcup, L)$	$(q_3, \triangleright, R)$

Da $n = 3$ müssen also $4 \cdot 2 = 8$ Einträge codiert werden. Die folgende Tabelle enthält die Codierung dieser 8 Einträge.

Nummer des Eintrags	Eintrag	Codierung
1	$\delta(q_0, 0) = (q_0, 0, R)$	0101010100
2	$\delta(q_0, 1) = (q_1, 1, R)$	0100100100100
3	$\delta(q_0, \sqcup) = (q_3, \sqcup, L)$	0100010000100010
4	$\delta(q_0, \triangleright) = (q_0, \triangleright, R)$	0100001010000100
5	$\delta(q_1, 0) = (q_1, 0, R)$	001010010100
6	$\delta(q_1, 1) = (q_2, 1, R)$	001001000100100
7	$\delta(q_1, \sqcup) = (q_3, \sqcup, L)$	00100010000100010
8	$\delta(q_1, \triangleright) = (q_3, \triangleright, R)$	00100001000010000100

Die Gödelnummer dieser Maschine ist nun

$$\begin{aligned}\langle M \rangle \;=\; & 1110101010100110100100100100110100010000010001011 \\ & 0100001010000100110010100101001100100100010010011 \\ & 0010001000010001011001000010000100001000100111\end{aligned}$$

Als natürliche Zahl betrachtet ist die Gödelnummer dieser Maschine

$$6387798827615802518730094253999341154150 79.$$

Definition 2.6.2 *Eine Turingmaschine M_0 heißt **universell**, falls für jede 1-Band-DTM M und jedes $x \in \{0,1\}^*$ gilt:*

- *M_0 gestartet mit $\langle M \rangle x$ hält genau dann, wenn M gestartet mit x hält.*
- *Falls M gestartet mit x hält, berechnet M_0 gestartet mit $\langle M \rangle x$ die gleiche Ausgabe wie M gestartet mit x. Insbesondere akzeptiert M_0 die Eingabe $\langle M \rangle x$ genau dann, wenn M die Eingabe x akzeptiert.*

Satz 2.6.3 *Es gibt eine universelle 2-Band-DTM M_0.*

Beweis: Gegeben eine beliebige DTM M zeigen wir, wie M_0 Konfigurationen von M kodiert, und wie wir einzelne Schritte von M auf M_0 simulieren können. Zu Beginn der Simulation wird der Beginn von x bestimmt. Nach Definition der Gödelnummer zeigt das zweite Auftauchen von 111 das Ende von $\langle M \rangle$ an. Die Maschine kopiert dann zunächst x auf ihr zweites Band. Sie bewegt den Kopf des zweiten Bandes auf das erste Zeichen in x. Dann löscht sie x auf dem ersten Band und schreibt 0, die Kodierung des Startzustandes q_0 hinter $\langle M \rangle$. Dann bewegt M_0 den Kopf des ersten Bandes zurück zum linken Ende des Bandes. Von nun an folgt die Kodierung von Konfigurationen und die Simulation der Schritte den Regeln:

Kodierung einer Konfiguration Eine Konfiguration $\alpha q_i X_j \beta$ von M wird von M_0 wie folgt Kodiert:

- Auf Band 1 steht $\langle M \rangle$ und der Zustand q_i, kodiert durch 0^{i+1}.
- Auf Band 2 steht die Bandinschrift $\alpha X_j \beta$ von M.
- Der Kopf von Band 2 steht auf X_j.

Simulation eines Schritts

- Suche auf Band 1 in $\langle M \rangle$ die Zeichenreihe $110^{i+1}10^j1$. Diese bildet den Anfang der Kodierung des Übergangs $\delta(q_i, X_j)$. Dazu kann $j \in \{1, 2, 3, 4\}$ im Zustand gespeichert werden, 0^{i+1} wird durch Vergleich mit der Kodierung von q_i ($\hat{=}\ 0^{i+1}$, auf Band 1 hinter $\langle M \rangle$) gefunden. Beachten Sie, dass q_i nicht im Zustand gespeichert werden kann. (Warum nicht?)
- Lese die dahinterstehende Zeichenreihe der Form $0^{k+1}10^l10^m$, ersetze hinter $\langle M \rangle$ die Kodierung 0^{i+1} von q_i durch 0^{k+1}, die Kodierung des neuen Zustands q_k, und speichere l, m im Zustand. Damit ist im Zustand die Information über den auszuführenden nächsten Schritt gespeichert: „überschreibe die Zelle der Kopfposition mit X_l und bewege den Kopf gemäß D_m“. Man beachte, dass der Zustandswechsel von q_i nach q_k bereits hinter $\langle M \rangle$ auf Band 1 gespeichert ist. Der Zustandswechsel muss nicht (und kann auch nicht) im Zustand gespeichert werden.
- Verändere Band 2 entsprechend dem im Zustand gespeicherten Befehl.

Ende der Simulation und Berechnung Halte, falls der Zustand $q_{n-1} = q_{\text{accept}}$ oder der Zustand $q_n = q_{\text{reject}}$ von M auf Band 1 hinter $\langle M \rangle$ gespeichert ist. Akzeptiere, falls der akzeptierende Endzustand q_{n-1} von M auf Band 1 hinter $\langle M \rangle$ steht. □

2.6.2 Eigenschaften von Turingmaschinen als entscheidbare bzw. rekursiv aufzählbare Sprachen

Da wir DTMs mit Hilfe der Gödelnummern mit Elementen aus $\{0, 1\}^*$ identifizieren, können wir nun auch Eigenschaften von Turingmaschinen mit Hilfe von Sprachen definieren. Weiter können wir uns dann fragen, ob diese Sprachen entscheidbar oder rekursiv aufzählbar sind.

Beispiel 1: Die einfachste Frage in Bezug auf Turingmaschinen ist sicherlich, ob etwas eine korrekt definierte DTM ist. Als Sprache über dem Eingabealphabet $\{0, 1\}$ erhalten wir dann

$$\text{Gödel} := \{w \in \{0,1\}^* \mid w \text{ ist die Gödelnummer einer DTM } M.\}$$

Lemma 2.6.4 *Die Sprache Gödel ist entscheidbar.*

Beweis: Zunächst überprüfen wir, ob die Eingabefolge w das richtige Format hat, d.h., die Folge beginnt und endet mit drei Einsen, zwischen zwei Teilfolgen bestehend aus jeweils zwei Einsen tauchen 4 Einsen auf, zwischen denen wiederum jeweils eine gewisse Anzahl von Nullen steht. Ist sichergestellt, dass die Folge w das richtige Format hat, müssen wir prüfen, dass alle

Forderungen, die an eine DTM gestellt werden, erfüllt sind. Z.B. müssen wir überprüfen, dass beim Lesen von $\triangleright$ die Übergangsfunktion festlegt, dass der Lesekopf nach rechts wandert. Für die Folge w bedeutet dieses, dass bei jeder Teilfolge $0^{i+1}10^j10^{k+1}0^l10^m$ mit $j = 4$ (das gelesene Zeichen ist $\triangleright$) gelten muss, dass $m = 2$ (der Kopf bewegt sich nach rechts). Diese Eigenschaft einer Folge kann sicherlich leicht durch eine Turingmaschine überprüft werden. Man überzeugt sich, dass auch alle anderen Forderungen an eine DTM leicht anhand der Gödelnummer überprüft werden können. □

Beispiel 2: Wir wollen wissen, ob eine DTM eine gewisse Anzahl von Zuständen besitzt. Als Sprache formuliert erhalten wir

$$\text{States} := \{(\langle M\rangle, d) \mid \text{Die DTM } M \text{ besitzt mindestens } d \text{ Zustände, } d \in \mathbb{N}.\}$$

Lemma 2.6.5 *Die Sprache States ist entscheidbar.*

Beweis: Zunächst einmal ist klar, dass wir eine DTM konstruieren können, die alle Folgen ablehnt, die nicht Paar einer Gödelnummer und einer natürlichen Zahl sind. Ist hingegen die Eingabe ein Paar bestehend aus der Gödelnummer $\langle M\rangle$ und der natürlichen Zahl d, so können wir anhand der Gödelnummer leicht feststellen, ob M mindestens d Zustände besitzt. Hierzu müssen wir nur überprüfen, ob die Gödelnummer eine Übergangsfunktion kodiert, die mindestens aus $d-2$ Zeilen besteht. Es müssen nur mindestens $d-2$ Zeilen sein, da für die Zustände $q_{\text{accept}}, q_{\text{reject}}$ die Übergangsfunktion einer DTM nicht definiert ist. □

Beispiel 3 - Das Halteproblem: Nun kommen wir zu einem Beispiel, das nicht nur für diese Vorlesung sondern auch für die Praxis von großer Bedeutung ist. Es handelt sich hierbei um das Halteproblem, das wir als Sprache immer mit H abkürzen werden:

$$H := \{\langle M\rangle x \mid M \text{ ist DTM, die gestartet mit Eingabe } x \text{ hält.}\}.$$

In die etwas modernere Terminologie übersetzt, fragen wir uns bei gegebenem Computerprogramm und gegebener Eingabe für das Programm, ob das Programm bei dieser Eingabe hält oder in eine Endschlosschleife gerät. Ein Problem, das offensichtlich sehr praxisrelevant ist. Wir wollen uns nun überlegen, dass das Halteproblem rekursiv aufzählbar ist. Später werden wir sehen, dass das Halteproblem nicht entscheidbar ist.

Satz 2.6.6 *Das Halteproblem ist rekursiv aufzählbar.*

Beweis: Um zu zeigen, dass das Halteproblem rekursiv aufzählbar ist, müssen wir eine DTM $\bar{M}$ konstruieren, die die Sprache H akzeptiert. Die DTM ist ein Abbildung 2.13 beschrieben. Die Beschreibung ist in einer Form angeben, die wir von nun an immer wieder zur Beschreibung von DTMs verwenden werden.

Um den ersten Schritt umzusetzen, erinnern wir uns, dass wir das Ende der Gödelnummer an der zweiten Teilfolge von drei aufeinanderfolgend Einsen erkennen können. Existiert eine solche Teilfolge nicht, ist die Eingabe nicht von der geforderten Form, und wir können ablehnen. Sonst können wir wie in Beispiel 1 überprüfen, ob die Teilfolge beginnend mit den ersten drei Einsen und endend mit den zweiten drei Einsen eine Gödelnummer ist. Den 2. Schritt bis 4. Schritt können wir mit der universellen Turingmaschine M_0 umsetzen.

Wir müssen noch zeigen, dass die DTM $\bar{M}$ die Sprache H akzeptiert. Hierzu ist für $w = \langle M\rangle x$ zweierlei zu zeigen.

$\bar{M}$ bei Eingabe $w \in \{0,1\}^*$

1. Falls w nicht von der Form $\langle M \rangle x$ ist, lehne ab.
2. Simuliere M mit Eingabe x.
3. Wird in 2. festgestellt, dass M die Eingabe x akzeptiert, akzeptiere $\langle M \rangle x$.
4. Wird in 2. festgestellt, dass M die Eingabe x ablehnt, akzeptiere die Eingabe $\langle M \rangle x$.

Abbildung 2.13: DTM $\bar{M}$, die H akzeptiert.

i) Ist $w \in H$, so akzeptiert $\bar{M}$ die Eingabe w.

ii) Ist $w \notin H$, so akzeptiert $\bar{M}$ die Eingabe w nicht.

zu i) Ist $w \in H$, so hält M bei Eingabe x. Dann wird auch $\bar{M}$ bei Eingabe $w = \langle M \rangle x$ halten. Aus 3. und 4. der Beschreibung von $\bar{M}$ folgt, dass $\bar{M}$ dann w akzeptiert.

zu ii) Ist $w \notin H$, so hält M bei Eingabe x nicht. Damit kann auch $\bar{M}$ bei Eingabe w nicht halten. Dieses bedeutet, dass $\bar{M}$ die Eingabe w nicht akzeptiert.

□

Man beachte, dass die DTM $\bar{M}$ die Sprache H wirklich nur akzeptiert und nicht entscheidet. Denn ist die Eingabe $w = \langle M \rangle x$ für $\bar{M}$ derart, dass M bei Eingabe x nicht hält, so wird $\bar{M}$ bei Eingabe $w = \langle M \rangle x$ ebenfalls nicht halten. Wie oben schon gesagt, werden wir auch sehen, dass H nicht entscheidbar ist.

Beispiel 4: Wir betrachten die folgende Sprache

$$\text{Useful} := \left\{ (\langle M \rangle, q) \;\middle|\; \begin{array}{l} M \text{ ist DTM mit Zustand } q, \text{ und es gibt eine Eingabe} \\ w, \text{ so dass } M \text{ gestartet mit } w \text{ den Zustand } q \text{ erreicht.} \end{array} \right\}$$

Auch diese Sprache ist praktisch von Bedeutung. Dieses wird klarer, wenn wir das Problem etwas umformulieren. Gegeben ist ein Computerprogramm bestehend aus mehreren Unterprogrammen. Wir wählen eines der Unterprogramme aus, und fragen uns, ob dieses Unterprogramm jemals von irgendeiner Eingabe aufgerufen wird. Wir zeigen nun

Lemma 2.6.7 *Useful ist rekursiv aufzählbar.*

Beweis: Um eine DTM zu konstruieren, die Useful akzeptiert, werden wir eine nützliche Technik kennenlernen. Die Technik beruht darauf, dass wir die Elemente in $\{0,1\}^*$ aufzählen oder nummerieren können. Und zwar geben wir einer Folge $w \in \{0,1\}^*$ die Nummer $(1w)_2$. In Worten, wir stellen der Bitfolge w zunächst eine 1 voran und erhalten dann eine neue Folge $1w$. Diese Folge interpretieren wir als die Binärdarstellung einer natürlichen Zahl. Diese Zahl wiederum ist die Nummer, die wir der Folge w zuordnen. Die Folge, der wir auf diese Weise die Nummer i zuordnen, bezeichnen wir mit w_i.

Betrachten wir einige Beispiele. Der leeren Folge ϵ ordnen wir die Nummer zu, deren Binärsdarstellung $1\epsilon = 1$ ist. Dieses ist die Zahl 1. Der Folge 00 ordnen wir die Nummer mit der Binärdarstellung 100 zu. Dieses ist die Zahl 4.

Wir ordnen auf die oben beschriebene Weise nicht nur jeder Folge eine Nummer zu. Umgekehrt gibt es auch zu jeder Zahl ≥ 1 eine Folge mit dieser Nummer. Um die Folge mit Nummer i zu erhalten, betrachten wir die Binärdarstellung von i, streichen die führende 1 und erhalten so die Folge mit Nummer i. Betrachten wir $i = 13$. Die Binärdarstellung von 13 ist 1101. Damit ist $w_{13} = 101$.

Nun können wir eine DTM E beschreiben, die die Sprache Useful akzeptiert.

E bei Eingabe $w \in \{0,1\}^*$

1. Lehne ab, falls w nicht von der Form $(\langle M \rangle, q)$ ist,
   wobei q ein Zustand von M ist.
2. Wiederhole für $i = 1, 2, 3, \ldots$ Schritte bis die DTM hält:
3. Simuliere für i Schritte die DTM M mit Eingabe $w_1, \ldots, w_i$.
4. Wird in 3. festgestellt, dass M bei einer der Eingaben nach
   höchstens i Schritten den Zustand q erreicht, akzeptiere.

Der 1. Schritt ist wieder leicht durchzuführen. Um die Simulationen im 3. Schritt durchzuführen, können wir wieder vorgehen wie bei der universellen Turingmaschinen. Diese statten wir aber jetzt noch zusätzlich mit einem Zähler aus, der jeweils sagt, wie viele Schritte bereits simuliert wurden. Im i-ten Durchlauf der Schleife in 2. wird dann jede Simulation abgebrochen, sobald der Zähler den Wert i erreicht hat.

Wir müssen nun zeigen, dass E die Sprache Useful akzeptiert. Eingaben, die nicht von der korrekten Form sind, werden im 1. Schritt abgelehnt. Ist die Eingabe für E von der Form $(\langle M \rangle, q)$, wobei q ein Zustand von M ist, M aber bei keiner Eingabe w in den Zustand q geht, wird E nicht halten. Vielmehr wird die Schleife in 2. für $i = 1, 2, 3, 4, \ldots$ durchlaufen.

Ist hingegen $(\langle M \rangle, q) \in$ Useful, so gibt es mindestens eine Folge $w_j \in \{0,1\}^*$, so dass M gestartet mit Eingabe w_j in den Zustand q geht. Erreicht nun M gestartet mit Eingabe w_j den Zustand nach k Schritten, so betrachten wir den Schleifendurchlauf im 2. Schritt von E für den Wert $i = \max\{j, k\}$. Hat E vorher noch nicht im akzeptierenden Zustand gehalten, wird nun M mit Eingabe w_j für mindestens k Schritte simuliert. Dann aber wird festgestellt, dass M bei Eingabe w_j den Zustand q erreicht. E hält dann im akzeptierenden Zustand.

Wir haben also gezeigt, dass E Eingaben, die nicht in Useful liegen, nicht akzeptiert, während alle Eingaben, die in Useful liegen auch von E akzeptiert werden. E akzeptiert somit die Sprache Useful. □

Wir wollen nun noch kurz die Frage klären, warum wir die Laufzeit für die Simulationen im 3. Schritt von E sukzessive erhöhen. Warum können wir nicht nacheinander M mit Eingabe $w_1, w_2, w_3, \ldots$ simulieren, um zu sehen, ob M bei einer Eingabe jemals den Zustand q erreicht? Betrachten wir hierzu den Fall, dass M bei Eingabe w_2 den Zustand q erreicht, M aber bei Eingabe w_1 nicht hält. Simulieren wir nun M nacheinander mit allen Eingaben, ohne die Zeit der Simulation zu beschränken, werden wir nie zur Simulation von M mit Eingabe w_2 gelangen, denn bei Eingabe w_1 gerät M, und damit auch die Simulation von M, in eine Endlosschleife. $(\langle M \rangle, q)$ liegt dann zwar in Useful, würde aber von E nicht akzeptiert werden.

Exkurs: Einschränkung auf die Standardalphabete Wir haben zu Beginn dieses Abschnitts vor der Definition von Gödelnummern behauptet, dass wir uns bei DTMs auf die so genannten Standardalphabete $\Gamma = \{0, 1, \triangleright, \sqcup\}$, $\Sigma = \{0, 1\}$ beschränken können. Dieses wollen wir nun begründen. Die Einschränkung auf die Standardalphabete wird sich auch im weiteren Verlauf der Vorlesung als nützlich erweisen.

Beliebige Alphabete Γ' können durch $\{0, 1, \sqcup\}$ simulieren werden, indem wir die Buchstaben $\neq \sqcup$ aus Γ' mit $1, \ldots, r$ durchnummerieren und den i-ten Buchstaben durch i Nullen gefolgt von einer 1 kodieren. Als Beispiel betrachten wir $\Gamma' = \{a, b, c, d, \triangleright, \sqcup\}$. Die Elemente aus Γ' werden dann durch $01, 001, 0001, 00001, \triangleright, \sqcup$ kodiert. Die letzte 1 fügen wir immer an, um das Ende der Kodierung einer Buchstabens erkennen zu können. Es gibt bessere Kodierungen, aber die gerade beschriebene reicht für unsere Zwecke. Von nun an werden wir daher immer annehmen $\Sigma = \{0, 1\}, \Gamma = \{0, 1, \triangleright, \sqcup\}$.