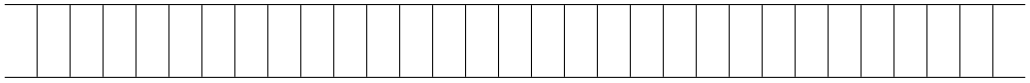
Rechnerische Fähigkeiten und Beschränkungen der Turingmaschine



Elia Doumerc

Jahresarbeit im Fach Mathematik

Deutsche Schule Las Palmas

Klasse 11b

Schuljahr 2020 / 2021

Betreuende Lehrkraft: Elisabeth Dauzenroth

Inhaltsverzeichnis

1 Einleitung

Im Jahre 1936 veröffentlichte der britische Mathematiker Alan Turing die zu jener Zeit äußerst ungewöhnliche Arbeit „On Computable Numbers, with an Application to the Entscheidungsproblem“. Darin wurde das Modell eines Rechners vorgestellt und auf seine mathematischen Eigenschaften hin gründlich untersucht. Durch deren weitreichende Folgen gilt Turing für viele als Begründer der theoretischen Informatik und einer der Väter des Computers. Das Ziel dieser Arbeit ist, die Turingmaschine und ihre Fähigkeiten und Beschränkungen [1] den LeserInnen näher zu bringen. Dabei stütze ich mich hauptsächlich auf das sehr empfehlenswerte Buch „The Annotated Turing: a Guided Tour through Alan Turing's Historic Paper on Computability“, geschrieben von dem amerikanischen Informatiker Charles Petzold (geb. 1953). Es erklärt Turings Schrift ausführlich und liefert die zum Verständnis des Textes benötigten Grundlagen und Hintergründe.

2 Hintergründe

2.1 Das Entscheidungsproblem

Im 19. und 20. Jahrhundert entwickelte sich die mathematische Welt rasend schnell. So wurde zum Beispiel die nichteuklidische Geometrie (die, in der das Parallelenaxiom[2] nicht gilt) geschaffen. Dazu wurde die Geometrie neu axiomatisiert, das heißt, ihre Grundlagen wurden völlig überarbeitet, da sie in den Augen der Mathematiker jener Zeit zu nah an der Wahrnehmung der Welt lagen und deswegen nicht solide genug waren (Petzold 36 - 38). [3]

Zu dieser Zeit entstand auch die *Principia Mathematica* (Russel und Whitehead), ein kolossales Werk, das als Hauptvertreter des Logizismus gilt und versuchte, alle mathematischen Sätze auf die Logik zurückzuführen (Es wurde zum Beispiel bei dieser Gelegenheit $1 + 1 = 2$ bewiesen.) (44). Dazu kamen noch viele weitere Erkenntnisse, deren Aufzählung hier den Rahmen sprengen würde.

Voraussetzung für die allgemeine Akzeptanz einer mathematischen Erkenntnis oder Formel war immer die Aufstellung eines wohlgeformten Axiomensystems, das diese

[1] Damit ist allgemein das von ihr Berechenbare gemeint.

[2] Ein Axiom ist ein Grundsatz, der keines Beweises bedarf, weil er als absolut richtig anerkannt ist.

[3] Euklid (3. Jh. v. Chr.) sprach zum Beispiel von Punkten und Linien, während andere versuchten, die Geometrie von solchen unabhängig (formaler) zu machen.

bewies. Ein Axiomensystem ist eine Sammlung von Axiomen, von denen Sätze bzw. Theoreme unter Beachtung bestimmter Regeln hergeleitet werden. Es muss zudem einige Eigenschaften vorweisen: Unabhängigkeit, was bedeutet, dass kein Axiom von anderen Axiomen hergeleitet werden kann, und Widerspruchsfreiheit. Wenn man von den Axiomen eine Aussage $a = b$ herleitet, darf man auch nicht zu einer Aussage $a \neq b$ gelangen.

Allerdings gab es Mathematiker wie David Hilbert (1862 - 1943), auch die Göttinger Mathematiker genannt, die noch einen Schritt weiter gingen und damit den Formalismus begründeten. Demnach wurden Axiomensysteme gründlich mit der mathematischen Logik untersucht. Diese sollten zudem zwei weitere Eigenschaften vorweisen: Vollständigkeit und Entscheidbarkeit. Mit Vollständigkeit war gemeint, dass alle aufgestellten Theoreme tatsächlich auf die Axiome zurückgeführt werden konnten (Man könnte ja zuerst eine wahr erscheinende Aussage formulieren und dann versuchen, sie zu beweisen.) (45 - 46).

Entscheidbarkeit ist für uns die wichtigste Eigenschaft. Ein System sollte nach Hilbert ein allgemeines Verfahren haben um zu entscheiden, ob ein Theorem beweisbar, bzw. von den Axiomen ableitbar ist. Mit Verfahren war eine schrittweise Anleitung gemeint. Wichtig dabei war, dass die Schritte nur Rechnungen waren, die durch ihre Einfachheit von Personen ausgeführt werden könnten, die absolut nichts von Mathematik verstehen, oder eben von Maschinen. Gesucht wurde also nach dem, was wir heute als Algorithmus kennen (47 - 48).

Um den Formalismus in Gang zu bringen, musste erst einmal das Axiomensystem der mathematischen Logik (genauer gesagt Prädikatenlogik) selbst untersucht werden. An der Widerspruchsfreiheit und Unabhängigkeit der Logik wurde nicht gezweifelt. Kurt Gödel (1906 - 1978) stellte zudem ihre Vollständigkeit fest. Die Logik sollte allerdings eine Grundlage für die Arithmetik bilden, weswegen sie von Gödel erweitert wurde, um jene darzustellen. Er bewies aber, dass das Axiomensystem dadurch unvollständig wird. Folglich kann man mit der Logik nicht beweisen, dass die Arithmetik widerspruchsfrei ist (48 - 50). [4]

Offen blieb aber noch deren Entscheidbarkeit. Für einige Sonderfälle gab es schon Verfahren, die von „menschlichen Maschinen“ durchgeführt werden konnten. Was Hilbert interessierte war, ob es ein allgemeines Verfahren für alle mit der Prädikatenlogik gebildete Formeln gab. Es sollte dessen Axiome und eine Formel als Eingabe nehmen

[4] Viele Mathematiker verloren dadurch das Interesse an der Logik.

und dann ausgeben, ob die Formel beweisbar war. Dieses Rätsel wurde auch als Entscheidungsproblem bekannt. Im Jahr 1936 bewiesen Alonzo Church und Alan Turing, unabhängig voneinander arbeitend, dass ein solches Verfahren unmöglich ist. Church schuf dazu das λ-Kalkül, Turing die äquivalente Turingmaschine (52). Die wichtigste Folge daraus ist, dass Mathematiker sich nicht mehr davor fürchten müssen, irgendwann durch Computer ersetzt zu werden.

2.2 Alan Turing

Alan Turing (siehe Abbildung 1) wurde 1912 in London geboren. Er studierte Mathematik am King's College in Cambridge und später, nach der Veröffentlichung seiner Arbeit über die Turingmaschine, in Princeton bei Alonzo Church, wo er den Doktortitel für eine Arbeit über Logik und Algebra erlangte. Während des zweiten Weltkriegs entschlüsselte er mit einer selbstgebauten Maschine den von den Nazis benutzten Enigma-Code und trug so in großem Maße zum Sieg der Alliierten bei. Nach dem Krieg beschäftigte er sich weiterhin mit Maschinen, aber auch neuronalen Netzen, die er teilweise selber berechnete, da dazu noch keine Maschine fähig war. 1950 formulierte er den Turing-Test. In ihm führt eine Person ein Gespräch mit zwei Partnern. Einer ist Mensch, der andere Maschine. Wenn die Person nicht entscheiden kann, welcher der beiden Partner die Maschine ist, besteht diese und gilt als intelligent. 1952 wurde Turing wegen seiner Homosexualität verhaftet und unterzog sich einer Hormontherapie, um dem Gefängnis zu entgehen. Durch diese wurde er depressiv, weshalb er sich 1954 schließlich das Leben nahm (Hodges). Für weitere Informationen über das Leben Turings ist der Film *The Imitation Game* sehr empfehlenswert.

Abbildung 1: Alan Turing ("Alan Turing in den 30er Jahren")

3 Die Turingmaschine

3.1 Struktur

Um das Entscheidungsproblem anzugehen, erfand Turing das Modell einer Maschine, die den gesuchten Algorithmus theoretisch ausführen sollte. Dabei richtete er sich nach dem mächtigsten Rechner, den er kannte: dem Menschen. Turings Modell sollte also analog zum Menschen beim schriftlichen Rechnen und dazu möglichst einfach, aber präzise sein. Seine Bestandteile waren:

- Eine Steuerungseinheit (das Hirn), bestehend aus einer endlichen Zahl an **Konfigurationen** (Maschinenzuständen). Diese beschreiben das Verhalten der Maschine und entsprechen den Gedankenzuständen des Menschen, da dieser zum Beispiel bei der schriftlichen Division in einem bestimmten Moment zwei Zahlen vergleicht, dann vielleicht dividiert, den Rest als Ziffer der gesuchten Zahl notiert und schließlich noch einmal von vorne beginnt. Man kann sie auch als das in der Schule verwendete „Regelheft“ sehen, nur das die Anleitungen deutlich kleinschrittiger sind.

- Ein **Band**, das man sich wie den heutigen Arbeitsspeicher vorstellen kann. Es entspricht dem vom Menschen benutzten mathematischen Papier, ist aber der Einfachheit halber eindimensional. Es wird in Kästchen unterteilt, die jeweils ein **Symbol** (beliebiges Zeichen) enthalten können.

- Ein **Lese-Schreibkopf** (Stift und Radiergummi). Er „scannt“ immer nur ein Kästchen und führt je nach Symbol und Konfiguration eine oder mehrere Aktionen aus. Diese können die Verschiebung um ein Kästchen in eine Richtung, das Wechseln der Konfiguration und / oder das Schreiben bzw. Löschen eines Symbols sein (Petzold 68 - 71).

Die Abbildung 2 verdeutlicht die beschriebenen Komponenten.

Die Maschine ist zudem **deterministisch**, das heißt, ihr Verhalten ist völlig durch die Konfigurationen vorbestimmt und danach nicht beeinflussbar (72).

An dieser Stelle sei erwähnt, dass Turing in keinem Moment eine Bauanleitung für seine Maschine lieferte oder erwartete, einen praktischen Nutzen aus ihr zu ziehen. Sie war ein reines Gedankenkonstrukt und nur dazu da, die theoretischen Fähigkeiten und Beschränkungen einer Maschine zu erforschen.

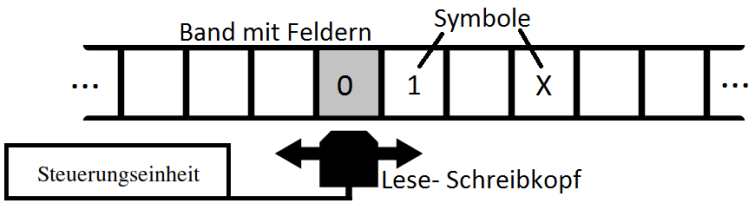


Abbildung 2: Skizze einer Turingmaschine
("1-Band Turingmaschine")

3.2 Die Maschine in Ausführung

Turing entschied sich, die Zahlen im Binärsystem darzustellen, da dies die Rechnungen einfacher machte (mit weniger Konfigurationen lösbar). Hier ist ein Beispiel für eine Binärzahl und die entsprechende Dezimalzahl:

$$0,1101 = 2^{-1} + 2^{-2} + 2^{-4} = 0,5 + 0,25 + 0,0625 = 0,8125$$

Er beschränkte seine Rechnungen zudem auf den Zahlenbereich zwischen 0 und 1, um Probleme mit der Kommadarstellung zu vermeiden. In der Maschine werden daher **Sequenzen** berechnet. Die Sequenz 01 entspricht beispielsweise der Binärzahl $0,01$. Man kann die Rechnungen der Maschine aber auch auf Zahlen außerhalb des betrachteten Bereiches übertragen (76 - 77).

3.2.1 Einfaches Beispiel

Im Folgenden werde ich anhand einiger Beispiele die Funktionsweise der Maschine verdeutlichen. Ich empfehle die Nutzung eines Simulators, um die Prozesse einfacher nachzuvollziehen. Zunächst etwas sehr Einfaches: die Sequenz $001001001...$ soll berechnet (geschrieben) werden. Die Steuerungseinheit (als Konfigurationstabelle dargestellt) dazu könnte wie Tabelle 1 aussehen.

Tabelle 1: Einfaches Beispiel

Konfiguration	Symbol	Aktionen	Endkonfiguration
𝔟		P0, R , P0, R	𝔠
𝔠		P1, R	𝔟

Gehen wir das die Funktionsweise der zugehörigen Maschine einmal durch: Sie befindet sich in der Startkonfiguration 𝔟 auf einem leeren Kästchen (grau markiert).

Nun scannt sie das Feld und führt die zum Symbol gehörende Aktion aus. In der Tabelle ist keine Aktion einem bestimmten Symbol zugewiesen, es wird also immer dieselbe ausgeführt. Der Befehl P0 beschreibt, dass das Symbol 0 gedruckt wird, R signalisiert dagegen die Bewegung um einen Schritt nach rechts.

Die Maschine hat die in 𝔟 festgelegten Aktionen ausgeführt und ist dann zur Konfiguration 𝔠 übergegangen (wurde in der Spalte „Endkonfiguration“ bestimmt). Nun erledigt sie die in 𝔠 bestimmten Aufgaben und wechselt wieder zu 𝔟, wo der Prozess noch einmal von vorne beginnt.

Turing benutzte für die Darstellung der „Maschinenschnappschüsse“ eine wesentlich kompaktere Notation: 𝔟_ : 00𝔠_ : 001𝔟_ : ...

Zu sehen sind die wichtigen Abschnitte des Bandes, mit einem Symbol pro Kästchen und einem Unterstrich für ein leeres Kästchen. Die Doppelpunkte sind dazu da, die verschiedenen Schnappschüsse zu trennen. Die Namen der Konfigurationen geben den Zustand und die Position des Kopfes an, da sie links vom aktuellen Feld (in unserem Beispiel leer) gedruckt sind (90 - 92).

3.2.2 Addition

Allerdings bevorzugte Turing es, das Band in F- und E-Kästchen einzuteilen. Sie sollten abwechselnd auftreten. In die F-Kästchen kamen die Ziffern der berechneten Sequenz, in die E-Kästchen andere Symbole, die zum Markieren von Ziffern verwendet wurden. So konnte er komplexere Operationen durchführen (93 - 94), wie das nächste Beispiel zeigen wird. Es sollen die binären Zahlen 0.101 und 0.0011 addiert werden. Sie wurden schon auf die F-Kästchen geschrieben und durch das Symbol #, worauf sich der Kopf der Maschine befindet, voneinander getrennt. [5]

Eine Konfigurationentabelle dazu könnte wie die Tabelle 2 aussehen.

Mit dieser Tabelle verhält sich die Maschine so, als würde sie die schriftliche Addition durchführen. In der Konfiguration *beg* wird die erste Ziffer des zweiten Summanden mit

[5] Eigentlich hätte ich es auf ein E-Kästchen schreiben müssen, das ist hier aber nicht so wichtig. Turing brauchte diese Unterscheidung vor allem für seine universelle Turingmaschine.

Tabelle 2: Additionstabelle

Konfiguration	Symbol	Aktionen	Endkonfiguration
beg	#	R, R, R, Px, L	$bewL$
$inkr$	0, leer	P1, R	$bewR0$
	1	P0, L, L	$inkr$
	#	Stopp	
$bewL$	leer	R, R	unt
	sonst	L, L	$bewL$
unt	0	E, R	$bewR0$
	1	E, R	$bewR1$
	#	Stopp	
$bewR1$	leer	R, R	$bewR1$
	x	L	$inkr$
$bewR0$	leer	R, R	$bewR0$
	x	E, R, R, Px, L	$bewL$

einem x markiert (rechts von ihr). Die Maschine wechselt dann zu $bewL$, wonach sie die erste Ziffer des ersten Summanden sucht. Sobald diese gefunden ist, wird sie mit unt untersucht. Hier gibt es drei mögliche Fälle:

- Ziffer 0 - Sie hat keinen Einfluss auf das Resultat, wird nicht weiter beachtet und gelöscht (Aktion E in der Tabelle). Die Maschine wechselt zu $bewR0$, um x eine Ziffer nach rechts zu verschieben, und bewegt sich dann mit $bewL$ zur nächsten Ziffer des ersten Summanden, um sie zu scannen.

- Ziffer 1 - Sie wird ebenfalls gelöscht, hat aber Einfluss auf das Ergebnis, weswegen mit $bewR1$ die Marke x gesucht wird, um die zugehörige Zahl dann zu inkrementieren, wofür $inkr$ sorgt. Ist die Ziffer bereits 1, wird sie übertragen (aus 0,01 wird zum Beispiel 0,10). Schließlich wird die Marke verschoben und die Maschine geht zur nächsten Ziffer des ersten Summanden.

- Symbol # - Dies bedeutet, dass wir schon alle Ziffern des ersten Summanden betrachtet haben und der Rechenvorgang beendet ist.

Spätestens an dieser Stelle sollten die Vorteile der Verwendung des Binärsystems klar geworden sein. Hätten wir zehn verschiedene Ziffern im System, müssten wir auch alle zehn Fälle einzeln behandeln.

Mit einer etwas ausgeklügelteren Tabelle, auf die ich jetzt nicht näher eingehen werde, lassen sich ebenfalls Zahlen multiplizieren (100 - 108). Die Turingmaschine ist

also zu den Grundoperationen, und somit auch allen abgeleiteten Operationen, wie die Subtraktion, Division, Wurzelrechnung usw., fähig. Das bedeutet, dass sie im Prinzip alle beliebigen Funktionen berechnen kann, auch wenn ihre Steuerungseinheit sehr schnell äußerst kompliziert wird und sie nicht effizient ist.

3.3 Die Universelle Turingmaschine

Strukturiert man eine Konfigurationstabelle (nehmen wir mal die von 3.2.1) um, so dass die Konfigurationen nun $q_1, q_2, ...$ und die Symbole $S_0, S_1, ...$ [6] heißen, und es in einer Konfiguration pro abgelesenes Symbol nur eine Schreibaktion und optional eine Bewegung gibt, dann wird die Tabelle zwar insgesamt länger und unübersichtlicher (vergleiche mit der Tabelle 3), die Maschine erleidet aber keinen Funktionsverlust (131 - 136).

Tabelle 3: Modifizierte Tabelle von 3.2.1

Konfiguration	Symbol	Aktionen	Endkonfiguration
q_1	S_0	S_1, R	q_2
q_2	S_0	S_1, R	q_3
q_3	S_0	S_2, R	q_1

Desweiteren lässt sich q_i durch D und $i \times A$, und S_j durch D und $j \times C$ ersetzen, wodurch sich für die Konfigurationen Buchstabenfolgen bilden, die Turing, durch Semikolons getrennt, in eine Zeile schreibt:

DADDCRDAA;DAADDCRDAAA;DAAADDCCRDA

Kodiert man diese Zeichenkette, indem man alle Buchstaben durch Zahlen ersetzt, erhält man eine ganze Zahl, die **Beschreibungsnummer** (137).

31332531173113325311173111331322531

Die von ihr definierte Maschine kann exakt eine Zahl berechnen. [7] Allerdings lässt sich dieselbe Zahl von mehreren Maschinen berechnen, da man zum Beispiel eine Konfiguration hinzufügen könnte, die nie aufgerufen wird und so zwar die Beschreibungsnummer und die Maschine, aber nicht ihre Funktionsweise verändert hätte (138).

[6] S_0 entspricht einem leeren Feld, S_1 entspricht 0, S_2 entspricht 1 usw.

[7] Hier wird davon ausgegangen, dass alle Maschinen ihre Aktivität mit einem leeren Band beginnen.

Diese Beschreibungsnummern sind in vieler Hinsicht interessant. Man kann sie zum Beispiel auf das Band von anderen, besonderen Turingmaschinen schreiben.

Eine davon ist die **Universelle** Turingmaschine, deren Konfigurationen Turing vollständig in seine Publikation einbezog. Diese ermöglichten der Maschine, den Ablauf von anderen Maschinen Schritt für Schritt nachzuvollziehen, indem sie, anhand der Beschreibungsnummer, die einzelnen Schnappschüsse wie in 3.2.2 durch relativ einfaches Kopieren von Symbolen druckte. Damit zeigte Turing, dass seine Maschine **programmierbar** war und die Fähigkeit hatte, alles zu berechnen, was andere Maschinen, die seine Konventionen befolgten, auch berechnen konnten (143 - 161).

4 Berechenbarkeit

4.1 Zahlen

Offen bleibt noch, welche genau die Fähigkeiten und Beschränkungen der Turingmaschine sind oder, mit anderen Worten, was von ihr berechenbar ist. Dem ging Turing nach, indem er die berechenbaren Zahlen definierte. Sie sind die reellen Zahlen, deren Dezimalstellen von einer Maschine geschrieben werden können. Das heißt, es muss immer eine Rechenanleitung geben (einen Algorithmus), um die nächste Ziffer der Zahl zu bestimmen (64).

Es ist deswegen wichtig, dass eine Maschine, die eine solche Zahl berechnet, immer fähig ist, die nächste Ziffer nach einer endlichen Zahl an Schritten zu bestimmen. Sie darf niemals aufhören, Ziffern auf das Band zu schreiben, unabhängig davon, zu welcher Art die berechnete Zahl gehört, und ob wir zum Beispiel aus einer unendlichen Folge an Nullen (wie bei der Berechnung von $\frac{1}{2} = 0,5000...$) einen Nutzen ziehen können. Erfüllt sie diese Anforderung, wird sie **kreisfrei** genannt. Ansonsten gilt sie als kaputt, und die Zahl ist von ihr nicht berechenbar (76).

Da die Turingmaschine, wie wir in 3.2.2 gesehen haben, die Grundrechenarten beherrscht, gehören zu den berechenbaren Zahlen der reelle Teil der algebraischen Zahlen (die imaginären Zahlen werden in Turings Arbeit nicht behandelt) und einige transzendente Zahlen wie π, e oder $01234567891011...$, die eine mehr oder weniger komplizierte Anleitung für ihre Bildung haben. Turing zeigte, dass nicht alle reellen Zahlen berechenbar sind, indem er die **Abzählbarkeit** der berechenbaren Zahlen mithilfe der Beschreibungsnummern (siehe 3.3) bewies. Da die reellen Zahlen nicht abzählbar

sind bedeutet das, dass so gut wie keine reelle Zahl berechenbar ist und der Großteil davon zufällige, unendliche Ziffernsequenzen sind (64 - 66).

Es gibt allerdings auch reelle Zahlen, deren Ziffern nicht zufällig sind (diese Zahlen sind also definiert), sich aber trotzdem nicht berechnen lassen. Eine davon werde ich in 5.1 vorstellen.

4.2 Funktionen

Im Grunde sind berechenbare Funktionen dasselbe wie berechenbare Zahlen. Turing hatte vor, die Beziehung zwischen beiden in einer weiteren Publikation zu verdeutlichen, kam aber nie dazu (64).

An Turings Konventionen kann man aber erkennen, dass er seine kreisfreie Maschine vor allem zur Berechnung von Zahlen gedacht hatte. Mein Beispiel von 3.2.2 hat kaum eine von seinen Regeln befolgt und war letztendlich zur einfachen Addition besser geeignet, da der Automat von alleine stoppte und von den Summanden unabhängig war. [8] Auch die modernen Rechner runden die reellen Zahlen immer bis zur benötigten Stelle. Um die Maschine in dieser Hinsicht tiefgründiger zu untersuchen, haben andere Mathematiker Jahre später ihre eigenen Varianten entworfen (mehr dazu in 6.1).

An dieser Stelle sei gesagt, dass selbst Funktionen wie

$$sin(x) = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

von kreisfreien Maschinen berechnet werden können. Das interessante hier ist, dass die Maschine unendlich viele Zahlen mit unendlichen Nachkommastellen gleichzeitig berechnen muss, da x als Radiant angegeben wird und deshalb mit $x \cdot \frac{\pi}{180}$ umgeformt werden muss (x ist also transzendent) und auf jeden der Summanden angewendet wird. Man kann die Maschine nämlich so konfigurieren, dass sie nur eine Stelle von π berechnet, damit eine Stelle von x und damit wiederum die erste Ziffer von $sin(x)$. Wiederholt man diese Schritte für jede weitere Ziffer, vermeidet man, dass die Maschine bei der Berechnung der ersten Zahl stecken bleibt. Wichtig ist immer die Fähigkeit, die Funktionalität von mehreren Maschinen zu schachteln (232 - 234).

[8] Um die Abzählbarkeit der berechenbaren Zahlen zu zeigen hatte Turing ja festgelegt, dass das Band am Anfang leer sein sollte

5 Anwendung auf das Entscheidungsproblem

5.1 Das Halteproblem

Wie in 4.1 angekündigt, stelle ich jetzt eine definierbare, aber nicht berechenbare Zahl vor, die zudem zeigen wird, dass sich das Diagonalargument von Georg Cantor (1845 - 1918) [9] nicht auf die berechenbaren Zahlen anwenden lässt.

Da die berechenbaren Zahlen abzählbar sind, können wir uns denken, die n-te Ziffer der n-ten Zahl zu nehmen, sie zu invertieren und aus den entnommenen Ziffern eine neue Zahl β zu konstruieren (siehe Abbildung 3). Obwohl β klar definiert ist, ist sie nicht berechenbar.

$$
\begin{array}{ll}
s_1 & = 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots \\
s_2 & = 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\ldots \\
s_3 & = 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\ldots \\
s_4 & = 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\ldots \\
s_5 & = 1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\ldots \\
s_6 & = 0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\ldots \\
s_7 & = 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\ldots \\
s_8 & = 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\ldots \\
s_9 & = 1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\ldots \\
s_{10} & = 1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\ldots \\
s_{11} & = 1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\ldots \\
\vdots & \quad \vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\ddots \\
\hline
s & = 1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\ldots
\end{array}
$$

Abbildung 3: Diagonalargument (Burghardt)

Um dies zu beweisen, nehmen wir an, es gibt eine Maschine zur Berechnung von β, genannt H. H besteht aus der universellen Turingmaschine U und einer Maschine D, die für eine Beschreibungsnummer allgemein bestimmt, ob sie zu einer kreisfreien Maschine gehört. Um β zu berechnen, generiert H alle natürlichen Zahlen und übergibt sie D. D entscheidet für jede Zahl, ob sie eine gültige Beschreibungsnummer ist. Wenn ja, gibt sie diese an U weiter, welche die codierte Maschine ausführt, bis die nächste Stelle von β erreicht ist. H invertiert die Ziffer dann und schreibt sie auf das Band. Dann fängt das ganze mit der nächsten Zahl erneut an, und es scheint so, als wäre H kreisfrei (Petzold 178 - 182).

Allerdings hat H ein Problem, sobald sie ihre eigene Beschreibungsnummer K generiert. Um die nächste Stelle von β zu bestimmen, muss sie alle bisher gefundenen Ziffern erneut berechnen, bis sie zu K kommt und das Gleiche noch einmal beginnt. H gerät in eine Endlosschleife, kann also nicht kreisfrei sein, und es entsteht ein Widerspruch. K lässt sich auch nicht einfach überspringen, da es unendlich viele Beschreibungsnummern wie K gibt, die zu Maschinen gehören, welche β berechnen oder Zahlen, die β ähnlich sind. Daraus folgt, dass D nicht existieren kann. Das heißt auch, dass es keine Maschine geben kann, die bestimmt, ob eine andere Maschine ein bestimmtes Symbol unendlich oft druckt (182 - 186).

[9] Er verwendete es um zu beweisen, dass die reellen Zahlen nicht abzählbar sind (Petzold 28 - 29).

5.2 Church-Turing-These

Diese Erkenntnis war für Turing von großer Bedeutung um zu zeigen, dass das Entscheidungsproblem unlösbar ist. Er stellte nämlich, eine Turingmaschine mit der Prädikatenlogik darstellend, eine Formel auf, die nur beweisbar ist, wenn die Formel bzw. Maschine irgendeinmal das Symbol 0 auf das Band schreibt. Da es dafür aber keine allgemeine Bestimmungsmethode gibt, ist das Entscheidungsproblem unlösbar (259). [10]

Es handelt sich hierbei um eine These, da der Begriff des menschlichen Rechnens zu intuitiv formuliert ist. Turing konnte nicht beweisen, dass seine Maschine rechnerisch genauso mächtig war wie ein Mensch. Bis jetzt wurde die These allerdings nicht widerlegt, was bedeutet, dass Menschen, moderne Computer und Turingmaschinen aus der Perspektive der Berechenbarkeitstheorie gesehen dasselbe sind. Die Rechenleistung bzw. Geschwindigkeit wird dabei nicht betrachtet, nur das Potential. Sie wird Church-Turing-These genannt, weil sich das λ-Kalkül von Church als Äquivalent zur Turingmaschine erwies (189).

6 Turingmaschinen in der Nachwelt

6.1 Varianten der Turingmaschine

Mit der Entwicklung der modernen Rechner ab dem zweiten Weltkrieg entstand auch der Bedarf nach realisierbaren Modellen, die ihnen eine Grundlage bieten konnten. Beim Bau von Rechnern wurde zwar hauptsächlich die Von-Neumann-Architektur [11] beachtet, die Turingmaschine wurde dennoch für theoretische Zwecke weiterentwickelt und erforscht. Es entstanden dabei zahlreiche Varianten, die jeweils mindestens eine eigene Jahresarbeit verdienen.

So untersuchten in den 50er Jahren die Mathematiker Stephen Kleene (1909 - 1994) und Martin Davis (*1928) zum Beispiel die Eigenschaften einer Maschine, die ausschließlich aus natürlichen Zahlen bestehende Funktionen berechnete. Sie stellte die Zahlen im Unärsystem dar und musste halten, um als brauchbar angesehen zu werden (234).

[10] Ich habe hier auf lange Formeln in der Sprache der Prädikatenlogik verzichtet und damit die Erklärung natürlich minimal gehalten.

[11] Benannt nach dem ungarisch-amerikanischen Mathematiker John von Neumann (1903 - 1957). Ihr zufolge sollte, kurz gefasst, ein Rechner elektronisch sein, mit Binärzahlen arbeiten und, unter anderem, aus einer Recheneinheit und einem zentralen Arbeitsspeicher bestehen. Sie übernahm also die wesentlichen Konzepte der Turingmaschine (166).

Interessant sind auch die nichtdeterministischen Maschinen, die nach jedem Schritt die nächste Konfiguration aus einer endlichen Liste **zufällig** [12] wählen. Diese Maschine halten ebenfalls, und man unterscheidet zwischen erfolgreichen und nicht-erfolgreichen Endzuständen.

Weitere Varianten sind die mehrbändige Maschine (kommt den modernen Rechnern am nächsten), die Turingmaschine mit n-dimensionalem Band und die Quanten-Turingmaschine, die für die Erforschung der Fähigkeiten von Quantencomputern verwendet wird (De Mol 2.5).

Obwohl all diese Modelle die originale Maschine beschränken oder erweitern, sind ihre rechnerischen Fähigkeiten gleich. Deswegen werden sie heute eher in der Komplexitätstheorie verwendet, die sich mit der Minimierung des Ressourcenverbrauchs von Algorithmen beschäftigt. Dieser wird mit dem Speicherplatzbedarf und der Rechenzeit gemessen (Walz).

6.2 Turing-Vollständigkeit

Die Turing-Vollständigkeit ist heutzutage ein wichtiger Maßstab geworden. Sie bezeichnet die Fähigkeit, alle Berechnungen durchzuführen, die eine universelle Turingmaschine ebenfalls durchführen kann. Diese und die betrachtete Maschine müssen sich gegenseitig simulieren können (Petzold 329). [13]

Die erste Maschine, die sie erreichte, war die 1944 von der Regierung der Vereinigten Staaten gebaute ENIAC (siehe Abbildung 4). Ihre Programmierung erfolgte durch das Ändern von Schaltkreisen und dauerte deswegen oft Tage. Sie wurde hauptsächlich zur Berechnung von Raketenflugbahnen verwendet (Swaine und Freiberger). So gut wie alle Rechner, die danach gebaut wurden, sind ebenfalls turingmächtig. Heutzutage wird auch bei der Erstellung einer neuen Programmiersprache darauf geachtet, dass ihre

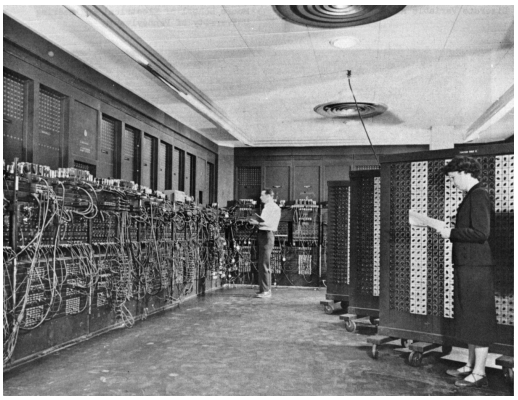


Abbildung 4: ENIAC ("ENIAC")

[12] Sie ist nicht realisierbar, da sich der Zufall per Definition nicht berechnen lässt. Moderne Rechner greifen für die Erstellung von scheinbar zufälligen Zahlen auf Messungen wie die zur Millisekunde genaue Uhrzeit oder die aktuelle Netzwerkgeschwindigkeit zurück (170 - 173).

[13] Man sieht dabei davon ab, dass Turingmaschinen eigentlich ein unendliches Speicherband haben.

Syntax das Simulieren einer universellen Turingmaschine ermöglicht (Petzold 329). Einige Programme wie Microsoft Excel (Jones) oder Minecraft (Zwinkau) sind teilweise turingmächtig.

7 Zusammenfassung

Im Wesentlichen ist die Turingmaschine ein theoretisches Rechnermodell, das lange vor der Entstehung der modernen Computer die Bestimmung derer Fähigkeiten und Beschränkungen ermöglichte. Diese lassen sich, wie in 4.1 gezeigt, auf das Bestimmen von berechenbaren Zahlen reduzieren, welche im Grunde die reellen Zahlen mit einem Algorithmus zur Bestimmung ihrer Dezimalstellen sind.

Im Abschnitt 3.3 habe ich zudem eine besondere Form der Turingmaschine vorgestellt: die universell programmierbare Turingmaschine. Sie kann jede andere Maschine simulieren und damit jede berechenbare Zahl berechnen. Erstaunlich daran ist, dass ihre Fähigkeiten und Beschränkungen denen von bis heute und in absehbarer Zukunft realisierbaren Rechnern und denen von Menschen entspricht. Bei solchen Überlegungen werden natürlich die in der Praxis sehr bedeutungsvollen Unterschiede in der Rechenzeit außer Acht gelassen.

Die erste ihrer zahlreichen Anwendungen war die Aufstellung einer These, nach der Hilberts Entscheidungsproblem, wie wir in 5.2 gesehen haben, unlösbar ist. Durch die zunehmende Bedeutung der digitalen Rechner wurde die Turingmaschine konstant erforscht und modifiziert (einige Varianten wurden in 6.1 erwähnt) und hat sowohl in der Berechenbarkeitstheorie, als auch in der Komplexitätstheorie eine große Bedeutung erlangt.

8 Literaturverzeichnis

Buch

Petzold, Charles. *The Annotated Turing: a Guided Tour through Alan Turing's Historic Paper on Computability*. Indianapolis: Wiley, 2008. **print**.

Lexika

De Mol, Liesbeth. "Turing Machines". *The Stanford Encyclopedia of Philosophy*. Winter 2019. Metaphysics Research Lab, Stanford University, 2019. **web**. 16. Jan. 2021.

Swaine, Michael R. und Paul A. Freiberger. "ENIAC". *Encyclopædia Britannica*. Encyclopædia Britannica, Inc., 7. Okt. 2008. **web**. 28. Jan. 2021.

Walz, Guido. "Komplexitätstheorie". *Lexikon Der Mathematik*. Spektrum der Wissenschaft Verlagsgesellschaft mbH, 2017. **web**. 28. Jan. 2021.

Webseiten

Hodges, Andrew. "Alan Turing — a short biography". *Alan Turing: The Enigma*. 1995. **web**. 9. Jan. 2021.

Jones, Brian. "Announcing LAMBDA: Turn Excel formulas into custom functions". *Microsoft Tech Community*. 12. März 2020. **web**. 2. Feb. 2021.

Zwinkau, Andreas. "Accidentally Turing-Complete". *Andreas Writing*. 20. Okt. 2013. **web**. 2. Feb. 2021.

Grafiken

"1-Band Turingmaschine". 20. Jan. 2009. Ich habe das Bild etwas verändert. **web**. 23. Feb. 2021.

"Alan Turing in den 30er Jahren". *Wikimedia Commons*. 7. Feb. 2017. **web**. 2. Feb. 2021.

Burghardt, Jochen. "Diagonal argument". *Wikimedia Commons*. 30. Dez. 2013. **web**. 2. Feb. 2021.

"ENIAC". *Wikimedia Commons*. 23. Sep. 2005. **web**. 2. Feb. 2021.

9 Erklärung

Ich, Elia Doumerc, erkläre, dass ich die Jahresarbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.

Unterschrieben in Las Palmas, am 2. März 2021