Seminarkurs - Alan Turing

Clara Maier, Nico Fröhlich

January 27, 2022

Hiermit versichern wir, die vorliegende Arbeit selbstständig und nur unter Zuhilfenahme der angegeben Quellen und Hilfsmittel angefertigt zu haben. Aus den angegebenen Materialien entnommene Inhalte und Zitate sind als solche kenntlich gemacht. Wir erklären, dass wir zu diesem Thema nicht schon einmal eine solche Arbeit angefertigt habe.

1 Vorwort

Der Vater des Computer; die Person, die den zweiten Weltkrieg wahrscheinlich um zwei Jahre verkürzte. Was gibt es da noch zu sagen? Alan Turing ist einer der bekanntesten und einflussreichsten Mathematiker des letzten Jahrhunderts. Er beeinflusst nicht nur die Mathematik und heutige Informationstechnik durch die Turingmaschine, sondern auch die Philosophie, durch zum Beispiel den Turing Test. In seiner meist zitierten Arbeit ging es sogar um Biologie. Am bekanntesten, nicht zuletzt durch den Film 'The Imitation Hame', ist jedoch seine Beteiligung am zweiten Weltkrieg, durch die Entschlüsselung der Enigma der Nazis. Seine Arbeit im Bereich Computertechnik, sowie in Philosophie, sein Papier über Maschinen und Intelligenz, sind mehr als nur interessant. Turing hat mit diesen Werken unsere heutiges Leben maßgeblich beeinflusst. In Kapitel 2 wird die Biografie rund um Alan Turing beleuchtet. Hierzu wird auch etwas auf den geschichtlichen Kontext eingegangen. Des Weiteren werden ein paar seiner Werke hier zeitlich eingeordnet. Darauf in Kapitel 3 erfährt man alles rund um die Verschlüsselungsmaschine Enigma, wie sie funktioniert, aus welchen Teilen diese besteht und warum sie geknackt wurde. Des Weiteren wird auf die Geschichte der Enigma, so wie Turings Arbeiten an der Entschlüsselung mit Hilfe der Bombe eingegangen. In Kapitel 4 wird auf das theoretische Datenverarbeitungssystem von Alan Turing eingegangen. Abschließend in Kapitel 5 geht es um die Frage, wie man Menschen von Maschinen unterscheidet und ob Maschinen denken können. Wie wichtig diese Fragen geworden sind und noch weiter werden, sowie die Einflüsse Turings Arbeiten in der Philosophie, werden hier auch beleuchtet.

Inhalt

2 Historie

Alan Mathison Turing wurde 1912, am 23. Juni in London geboren.
Er machte seinen Abschluss an der Sherborne School in Dorset und besuchte anschließend ab Oktober 1931 das King's College in Cambridge, an dem er Mathematik studierte. Er beendete sein Studium 1934 und März 1935 arbeitete er im Alter von 22 Jahren bereits als Dozent am King's College.
1936 veröffentlichte er eine seiner wichtigsten Werke, *"On Computable Numbers, with an Application to the Entscheidungsproblem"* , welche das Prinzip der Turing Maschine beinhaltet. Durch die Unterstützung von John von Neumann und Max Newman arbeiteten 1945 bereits mehrere Gruppen an der Umsetzung der Turing Maschine, also an der Umsetzung des ersten Computers. Turing verließ 1936 Cambridge um seine Forschungen an der Princeton University weiter zu führen und veröffentlichte 1938 *"Systems of Logic Based on Ordinals"* .
Im Sommer 1938 kehrte Turing als Dozent an die King's zurück, bis er beim Ausbruch des Krieges im September 1939 in den Bletchley Park zog, ein militärisches Lager zur Entschlüsselung deutscher Nachrichten. Turings exzellente Arbeit hatte kriegsentscheidende Konsequenzen. Turing war der Hauptentwickler der Turing Bombe, einer extrem schnellen Entschlüsselungsmaschine für die Enigma-verschlüsselte Nachrichten der Deutschen. Im Nachhinein wird vermutet, dass die Arbeiten von Turing und seinen Kollegen den Krieg in Europa um mindestens 2 Jahre verkürzten.

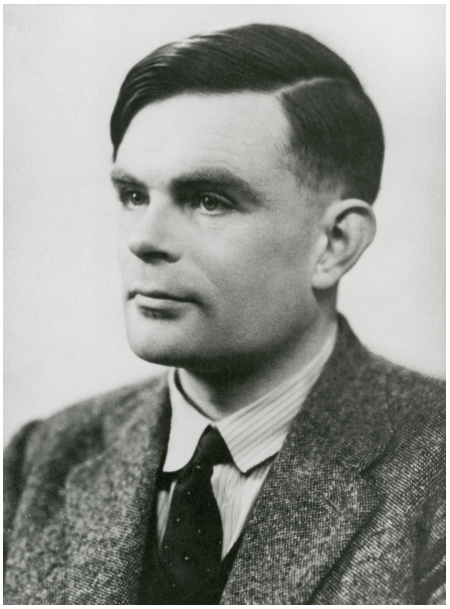

Abbildung 1: Alan Turing

Als der Krieg 1945 endete, ging Turing nach London ins National Physical Laboratory, um seine Theorie der Turing Maschine in funktionierende Hardware umzusetzen. Im Februar 1947 veröffentlichte er *"Lecture on the Automatic Computing Engine"* , in der er als Erster über Computerintelligenz schrieb und in seinem technischen Bericht 1948 "Intelligence Machinery" definierte er den Begriff 'Artificial Intelligence', kurz auch 'AI'. Zwei Jahre später erschien ein weiterer Artikel, *"Computing Machinery and Intelligence"*, in welchen er den heute bekannten Turing Test behandelte, welcher testet, ob ein System und die Intelligenz eines Menschen zu vergleichen ist. Der erste Computer wurde Juni 1948, auf Grundlage der Turing Maschine, am Computing Machine Laboratory, unter der Leitung von Max Newman, an der University of Manchester entwickelt. Noch im selben Jahr kam Turing der Einladung von Newman nach und wurde Direktor am Computing Machine Laboratory an der University of Manchester.

Turing wurde März 1951, aufgrund seines großen Einflusses im Krieg, als Mitglied der Königlichen Gesellschaft erwählt. Genau ein Jahr später, März 1952, wird Turing in Manchester für seine Homosexualität, die in Britannien damals eine Straftat darstellte, mit einer zwölfmonatigen Hormontherapie zur Rechenschaft gezogen, die ihn kastrierte. Turing erkrankte anschließend an einer Depression.

In den folgenden Jahren beschäftigte sich Turing mit künstlichem Leben und veröffentlichte 1952 den Artikel *"The Chemical Basis of Morphgenesis"*, 1953 dann einen Artikel über Computerschach und schließlich in 1954 den Artikel *"Solvable and Unsolvable Numbers"*, der sich auf den früheren Artikel *"On Computable Numbers"* bezieht.

Während weiterer Forschungen verstarb Turing schließlich unerwartet am 7. Juni 1954 im Alter von 41 Jahren in seinem Wohnort in Wilmslow in Cheshire. Es wird davon ausgegangen, dass Turing aufgrund seiner Depressionen Selbstmord begangen hat und sich mit Zyanid selbst vergiftete.

Man kann aber nicht ganz ausschließen, dass Turing entweder durch Zyanid Dämpfe seiner Experimente oder auch durchaus durch einen Attentat ums Leben kam.

3 Enigma

In diesem Abschnitt wird die Enigma und ihre Entschlüsselung thematisiert. Des Weiteren wird auf die historischen Hintergründe eingegangen, auf ihre Funktionsweise sowie ihre Fehler.

3.1 Enigma

Die Enigma ist eine Verschlüsselungsmaschine, die im zweiten Weltkrieg von den Deutschen genutzt wurde um militärische Fernkommunikation zwischen ihren Truppen zu etablieren. Diese sollte ihnen mit unter einen entschiedenen Vorteil liefern. Dennoch hatte diese Verschlüsselung ein paar sehr große Probleme, die dazu führten, dass die Alliierten immer wieder sehr viele Informationen entschlüsseln konnten.

Hauptbestandteile der Enigma waren:

1. Eingabetastatur
2. Lichtanzeige
3. Steckbrett
4. Reflektor
5. drei oder vier Walzen

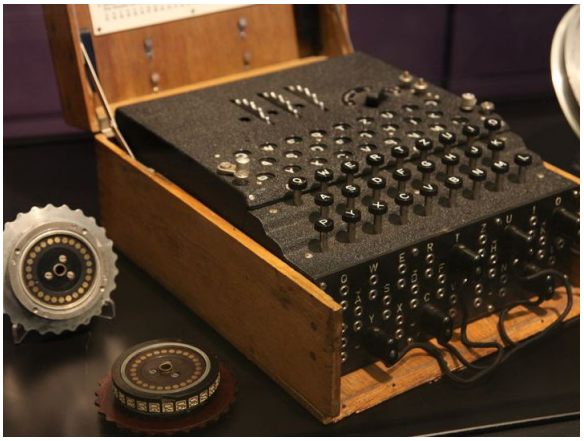

Abbildung 2: Enigma

3.1.1 Eingabetastatur

Eine zur Schreibmaschinen ähnliche Tastatur mit allen 26 Buchstaben des Alphabets, in die man die Nachricht eintippen konnte. Diese Tastatur war mit dem Steckbrett verknüpft.

3.1.2 Lichtanzeige

Eine Anzeige mit allen 26 Buchstaben des Alphabets, die einzeln beleuchtet werden und die verschlüsselte Nachricht anzeigen, z.B ein H wird eingetippt und ein R wird beleuchtet.

3.1.3 Steckbrett

Das Steckbrett, welches hinten an der Enigma angebracht war, diente zur weiteren Verschlüsselung, da man hier bis zu 13 von 26 Buchstaben manuell miteinander verlinken konnte. Somit wurde z.B. ein eingegebenes A zu einem H und vice versa. Wenn kein Kabel eingesteckt wurde, dann wurde der Buchstabe einfach normal durch geschaltet. Meist wurden nur 10 umgesteckt, da sonst die Anzahl der Möglichkeiten wieder geringer wurden.

3.1.4 Walzen

Bestandteile einer Walze:

1. Ring
2. Markierung des Buchstabens *A*
3. Buchstabenring
4. Kontaktplatten
5. Verbindungsdrähte (rewired)
6. Kontaktstifte (gefedert)
7. Sperrung (gefederte)
8. Nabe
9. Handrändel
10. Zahnrad

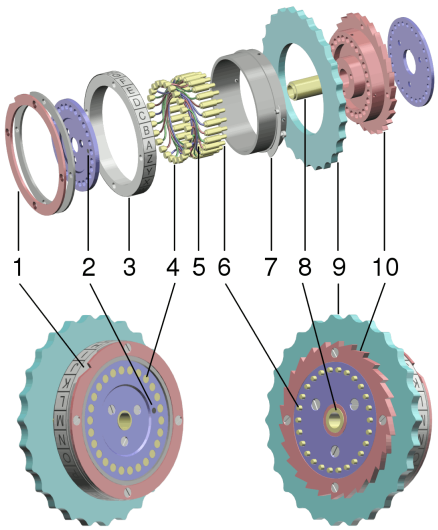

Abbildung 3: Walzen Innen

Die Walzen dienten zum weiteren Verschlüsseln der Nachrichten und jede Walze hatte 26 Stufen für jeden Buchstaben des Alphabets, wobei das Signal nun innerhalb der Walze durch neue Verkabelung auf einen anderen Buchstaben geleitet wurde (sog. rewiring). Das Signal wurde an die nächste Walze weitergeleitet, dort passiert das selbe nochmal. Es gab 8 verschiedene Arten von Walzen, diese besaßen pro Art immer eine andere Verkabelung innerhalb der Walzen. Walzen der gleichen Art hatten aber immer die selbe Verkabelung. Die erste Walze, welches die Eingabe passierte drehte sich bei jedem Tastendruck um einen Buchstaben weiter. Die nachfolgenden Walzen drehten sich immer dann um eine Stufe, wenn sich die vorherige an dessen Überrollpunkt befindet. Dieser ist meist, wenn sich die Walze einmal um sich selber gedreht hat, das kann etwa bei M oder aber auch Z sein. Der Überrollpunkt konnte über eine Offset-Markierung an den Walzen eingestellt werden (sog. ring setting), sodass sich die Neuverkabelung um so viele Stellen wie eingestellt in eine Richtung weiter drehten. Diese wurden am Anfang jeder Nachricht einmal eingestellt. Am Anfang wurden diese Starteinstellungen über die Nachricht mitgeteilt, in dem man ein deutsches Wort mit 3 Buchstaben zweimal hintereinander gesendet hat, was sich jedoch schnell als ziemlich schlechte Idee herausstellte und somit 1938 verboten wurde.

3.1.5 Reflektor

Der Reflektor schickte die Signale, welche von den Rollen kamen, durch eine andere Leitung an den Rollen wieder zurück. Dieser konnte nie zum gleichen Buchstaben zurück senden, sondern wurde wieder neu verkabelt.

3.1.6 Beispiel

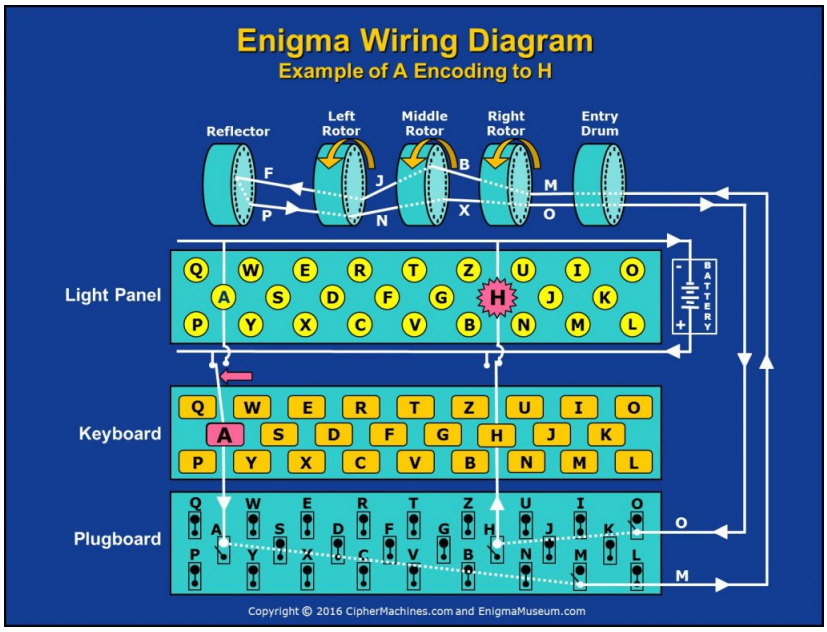

Abbildung 4: Beispiel Verkabelung

Der eingegebene Buchstabe A geht erst einmal an das Steckbrett und ist hier mit dem Buchstaben M verbunden. Dieser wird dann an die Walzen weitergegeben, die den Buchstaben je nach Stellung verändern, hier von M zu B zu J und schließlich zu F. Anschließend wird der Buchstabe wieder zurück durch die Walzen geleitet und kommt in diesem Beispiel als O heraus und wird wieder an das Steckbrett weitergeleitet. Der Buchstabe O ist hier mit H verbunden und so wird am Ende H in der Anzeige beleuchtet, und somit wurde A zu H verschlüsselt.

3.2 Historie

Turing beschäftigte sich bereits vor Beginn des zweiten Weltkriegs mit den Enigma Maschinen. Diese waren jedoch von den Italienern und wesentlich simpler als die Enigma Maschinen, die später von den Deutschen eingesetzt wurden. Die erste Version der Enigma war frei verkäuflich und wurde eher

privat genutzt, da sie nicht viel Sicherheit bot. In dieser Version gab es lediglich 3 der in Sektion 3.1.4 beschriebenen Räder, die in unterschiedlicher Reihenfolge eingesetzt werden konnten. Damit gab es lediglich $3! = 6$ Anordnungsmöglichkeiten. Zu dieser Zeit gab es außerdem kein wie in Sektion 3.1.3 aufgeführtes Steckbrett. Ein polnisches Verschlüsselungsteam unter der Leitung von Marian Rejewski legte den Grundstein und knackte die Verschlüsselungen der Enigma sehr schnell, dort wurden sie zum ersten Mal mit Steckbrett Versionen konfrontiert. Diese knackten sie von Hand mit Hilfe von Lochpapieren. Deutschland wollte sich diese Verschlüsselungsmaschinen nun für militärische Zwecke zu nutzen machen, so entwickelten sie die nächste Version, die militärische Enigma. In dieser Version wurden zwei Walzen zur Auswahl hinzugefügt. Nun wurden also 3 aus 5 Walzen genommen und in unterschiedlichen Anordnungen in der Enigma kombiniert. Diese wurden nachher großflächig für die deutsche Luftwaffe und Armee eingesetzt. Doch der Marine war das nicht genug. Unter Admiral Karl Dönitz wurde eine Enigma speziell für die Marine entwickelt, bei der zuerst 3 aus 8 Walzen und später dann 4 aus 8 Walzen für die Verschlüsselung verwendet wurden. Zu allem Überfluss drehten sich die Räder 6 bis 8 doppelt an ihrem Überrollpunkt. Damit explodierten die möglichen Kombinationen fast exponentiell. Bei Ausbruch des Krieges wurde Turing zusammen mit 9.000 anderen Spezialisten in den Bletchley Park geholt. Dieser Park wurde zu einer wahren Entschlüsselungsfabrik, in dem später mit Hilfe der *Bombe* 39.000 Enigma Nachrichten im Monat entschlüsselt wurden. Die sogenannte *Bombe* war eine automatisierte Maschine zur Entschlüsselung der Enigma Nachrichten. Diese wurde ursprünglich von den Polen bis zur Invasion entwickelt. Durch die Invasion jedoch übergaben sie die von den Polen *Bomba* genannte Maschine an die Briten. Dort verbesserten sie die Maschine weiter und bauten im Zuge des Krieges zwischen 60 und 100 verschiedene *Bombes* in Großbritannien und noch einmal mindestens so viel

in der USA. Dort halfen sie den Briten beim dechiffrieren per Unterseekabel. In Bletchley Park wurde der Platz sehr eng, deswegen wurden um das eigentliche Gebäude mehrere Hütten gebaut, die sogenannten *Huts*. Schnell bekamen die *Huts* spezielle Aufgaben. So war Hut 8, der Arbeitsplatz von Turing, mit der Entschlüsselung der sehr schweren Marine Enigma betraut. Die normalen Enigma Texte der Armee und der Luftwaffe wiederum wurden in Hut 6 unter der Leitung von Gordon Welchman, ein weiterer berühmter Statistiker, bearbeitet. [1] [2]

3.3 Anzahl der Möglichkeiten

Da die Turing Bombe versucht hat den Code größtenteils durch Ausprobieren heraus zu finden, war das aufgrund der Anzahl an Möglichkeiten schier unmöglich. Dennoch konnte man die Anzahl an Möglichkeiten reduzieren (siehe nächstes Kapitel). Hier möchte ich einmal auf die Anzahl der Möglichkeiten bei einer normalen Enigma der Armee eingehen. Dadurch, dass man 3 aus 5 Rotoren auswählen musste, ergibt sich $5 \cdot 4 \cdot 3 = 60$ Möglichkeiten (hier gilt Ziehen ohne zurücklegen, da nie zwei Walzen der gleichen Art in der Enigma waren). Dazu kommen noch die Ringeinstellungen. Hier gab es für jede Walze eine Ringeinstellung mit je 26 Möglichkeiten, bei 3 Walzen macht das $26^3 = 17576$ Möglichkeiten oben drauf. Das Schlimmste war jedoch das Steckbrett. Es gibt 26! Möglichkeiten Buchstaben miteinander zu kombinieren. Hier wurden jedoch immer nur 20 Buchstaben neu verbunden, somit bleiben 6 übrig also 6!. Die Ordnung der Verbindungen ist egal, da AB als äquivalent zu BA gilt. Somit gibt es für das Steckbrett $\frac{26!}{2^{10} \cdot 6! \cdot 10!}$, das sind ca. $1.5 \cdot 10^{13}$ Möglichkeiten. Wenn man alle Möglichkeiten zusammenrechnet kommt man auf ca. $1.6 \cdot 10^{20}$ Möglichkeiten. [3]

3.4 Schwachstellen

Die Enigma hatte einige Schwachstellen, die im ersten Moment vielleicht nicht als solche erscheinen. Dazu gehörte einmal, dass der Reflektor die Signale nie auf den gleichen Buchstaben zurückleiten konnte. Dadurch konnte man versuchen bekannte Wörter wie zum Beispiel "Wetterbericht" unter den verschlüsselten Text zu halten. Wenn einer der Buchstaben in dem Wort mit dem verschlüsselten Text an der Stelle übereinstimmte, dann konnte es sich nicht um das Wort handeln. Ziel der Entschlüsselung war es heraus zu finden, mit welchen der Räder und mit welchen Walzeneinstellungen sie die *Bombe* bestücken mussten. Diese sollte dann automatisch die Nachrichten entschlüsseln. Des Weiteren war bei den Rollen 6 bis 8 der Marine das zweite Überrollen ein Indikator dafür, dass eine solche Walze eingesetzt wurde. Dies konnte die Möglichkeiten sehr einschränken. Das oben genannte Problem mit dem doppelten Senden eines deutschen Wortes mit 3 Buchstaben hat natürlich dafür gesorgt, dass es hier über die Wahrscheinlichkeiten der Buchstaben in der Sprache sehr einfach war die Ringeinstellung zu erraten. Mithilfe des oben genannten Reflektor Problems minimierte es natürlich weiter die Möglichkeiten. Dieses Problem bemerkten die Deutschen und verbaten diese Übersendung von Ringeinstellungen. Des Weiteren halfen gestohlene Codebücher und Insiderinformationen beim Entschlüsseln. Dazu kam, dass die Deutschen jeden Morgen um Punkt 6 Uhr ihren Wetterbericht sendeten, denn dieser begann mit dem Wort *Wetterbricht* und endete mit *Heil Hitler*. So konnte man erheblich die Einstellungsmöglichkeiten reduzieren, da man Verbindungen ausschließen konnte. [4] [1]

3.4.1 Beispiel

Man nehme an wir haben den Ausgangstext "wetterberichttesttesttestheilhitler" und geben den in eine Enigma mit den folgenden Einstellungen ein.

Dann bekommt man folgendes Resultat: ”ejzsx opnes prwib kujgg izyac fjpdz hyzsd”. Wenn man nun versucht immer das Wort Wetterbericht darunter zu legen, kann man ganz schnell mögliche Positionen ausschließen, bei welchen das Wort nicht stehen kann.

ejzsxopnesprwibkujggizyacfjpdzhyzsd wetterbericht
ejzsxopnesprwibkujggizyacfjpdzhyzsd wetterbericht
ejzsxopnesprwibkujggizyacfjpdzhyzsd wetterbericht

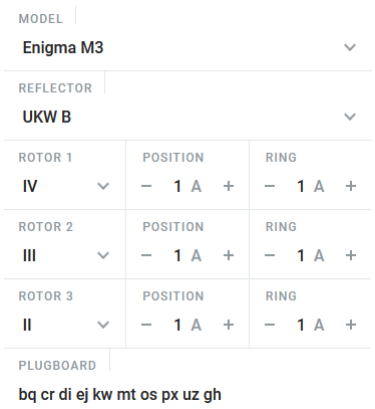

Abbildung 5: Beispiel Enigma Code [5]

Wie in der Tabelle ersichtlich ist, ist es möglich, dass das Wort Wetterbericht als erstes Wort im verschlüsselten Text ist, aber nicht an zweiter Stelle, da das dritte E mit dem Code übereinstimmt. Es kann aber wieder an der dritten Stelle stehen, und so weiter. Nun vermuten wir, dass es sich bei dem Text um den Wetterbericht handelt. Ziel ist es nun herauszufinden, wie das Steckbrett und die Walzen eingestellt worden sind. Denn diese Einstellungen veränderten sich täglich. Es kam auch sehr selten vor, dass man unbemerkt eines der deutschen Codebücher stehlen konnte, dies machte den Prozess natürlich überflüssig, war aber sehr sehr selten.

4 Turing Maschine

Die Turingmaschine wurde 1936 von Alan Turing entwickelt und ist rein theoretisch, denn eine Maschine exakt wie es Turing beschreibt wurde niemals gebaut, da dies auch nicht sinnvoll wäre. Turing entwickelte diese Theorie aufgrund des Entscheidungsproblem von David Hilber und Wilhelm Ackermann [2]. Die Turing Maschine ist die heutige Grundlage vieler Programmiersprachen, wie Java, C++ oder Python. Diese Programmiersprachen werden auch als *"Turing complete"* bezeichnet, was so viel heißt wie, dass die Turing Maschine diese theoretischerweise emulieren kann.

4.1 Deterministische Turingmaschine

Die Theorie der deterministischen Turingmaschine beschreibt die Aktionen der Turingmaschine als etwas, dass von Anfang an nur einen einzigen klaren Weg hat, sodass man theoretischerweise vor Programmstart sagen kann, welchen Weg die Turingmaschine nimmt. Diese Theorie ist auf alle Programme anwendbar, die Turing complete sind.

4.2 Nichtdeterministische Turingmaschine

Die Theorie der nichtdeterministischen Turingmaschine beschreibt keinen klaren von Anfang an festgelegten Weg, sondern vielmehr ein Programm, dass viele verschiedene mögliche Wege hat um ans Ziel zu kommen. Dieses Modell ist allerdings rein theoretisch und nach dem heutigen Wissensstand nicht realisierbar.

4.3 Funktion

Die Turing Maschine beinhaltet lediglich zwei Bauteile.

1. Ein unendlich langes Band, dass in eine Kette von horizontalen Kästchen eingeteilt ist. Jedes dieser Kästchen kann entweder eine 0, eine 1 oder

auch nichts beinhalten. Zudem kann das Band nach rechts und nach links verschoben werden.

2. Ein Kasten, auch "Heap" oder auch "Scanner" genannt, welcher die Zahlen auf dem Band einlesen, löschen und ändern kann.

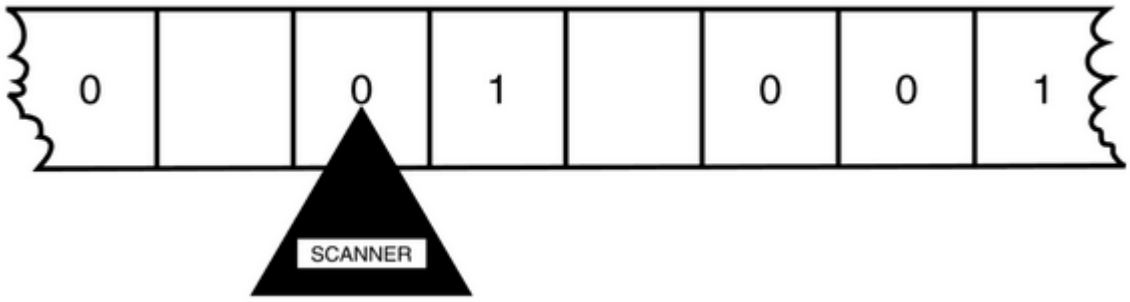

Abbildung 6: Beispielhafte Darstellung einer Turing Maschine [2]

4.4 Beispiel

Zur besseren Vorstellung, wird das Ganze an einem Beispiel erklärt. Nehmen wir mal an, wir wollen, dass die Turing Maschine für uns unendlich hoch zählt. So kann man dies mit nur zwei einfachen Befehlen ausführen.

- Befehl 1: Bei einer 1, ändere diese in eine 0 und gehe nach links
- Befehl 2: Bei einer 0 oder einer Lücke, ändere diese in eine 1 und gehe zum letzten Bit der Zahl

Die Befehle für die Turing Maschine kann man sich also heute wie die Programme auf unseren Computern vorstellen.

4.5 Codebeispiel

Dieser Code aus Java macht praktisch genau dasselbe, wie das kleine Beispielprogramm der Turing Maschine. Das Einzige was auffällt ist, dass dieses

Programm nur bis 1000 zählt, da unsere Turing Maschine einen unendlichen Speicher besitzt, anders als jeder PC auf der Welt.

```
for (int i = 0; i < 1000; i++) {}
```

In der for-Schleife wird zuerst die Variable i mit dem Datentyp Integer versehen und auf null gesetzt. Hinter dem ersten Semikolon steht dann die Bedingung, solange die Variable i kleiner wie 1000 ist. Hinter dem zweiten Semikolon steht dann die Anweisung dafür, was passiert solange die Bedingung erfüllt ist, das i++ bedeutet, dass die Variable i um eins hochgezählt wird.

5 Turingtest

In diesem Abschnitt geht es um Turings Ausflug in die Philosophie und sein Werk *"Computing Machinery and Intelligence"*. Dabei wird die Frage betrachtet, ob Maschinen denken können und wie man eine Maschine von einem Menschen unterscheiden kann, sowie heutige Anwendungen und Auswirkungen auf Kunst und Gesellschaft.

5.1 The Imitation Game

Turing stellt eine bessere Frage in den Raum, als die Frage "Können Maschinen denken?". Um die Frage allerdings stellen zu können, muss das "Imitation Game" eingeführt werden. Das Spiel funktioniert wie folgt. Lass A ein Mann sein, B eine Frau und C einen Detektiv. Der Detektiv (C) will also heraus finden, wer welches Geschlecht hat, ohne dabei die Stimme zu hören oder die Person zu sehen. Um diesen Zustand zu gewährleisten, gehen wir davon aus, dass der Detektiv sich in einem anderen Raum als A und B befindet. Die einzige Kommunikation welche besteht ist also textbasiert. Also versucht C Fragen zu verwenden, um mit Hilfe der Antworten beider Personen zu bestimmen, welche Person weiblich und welche männlich ist. Dies wendet Turing nun auf Maschinen an, dabei beschränkt er sich auf Digitalcomputer. Es wird versucht auf äußerliche Fragen zu verzichten, da dies zu keinem eindeutigen Ergebnis führen würde. Die Maschine könnte ganz einfach ein eigenes fingiertes Aussehen annehmen und die Person kennt ihres, daher ist es schwierig anhand der Antworten eindeutig zu unterscheiden. Deshalb versucht C logische Fragen zu verwenden um heraus zu finden, ob es sich um eine reale Person handelt. Beispiel: Subtrahiere 56 von 18934. [6] Dieses Spiel wird nun nicht nur dazu genutzt eine Maschine von einem Menschen zu unterscheiden, sondern es wird implizit die Frage gestellt, ob eine Maschine denken kann. Wenn man ihre Antworten nämlich nicht mehr voneinander unterscheiden kann, muss man sich Gedanken darüber machen, ob wir dann

nicht intelligentes Leben geschaffen haben könnten und wie wir dieses dann behandeln.

5.2 Chinese Room Problem

Eine der berühmtesten Probleme des Turing Tests ist das "Chinese Room Problem". Dieses Problem stellt in Frage, ob man mit dem Turingtest wirklich die Intelligenz eines Lebewesens feststellen kann. Dabei wird eine weitere Situation angenommen. Nehmen wir an, zwei Personen sitzen gegenüber voneinander in einem Raum. Die eine Person ist chinesisch, die andere Person deutsch. Auf dem Tisch liegt nun ein Deutsch-Chinesisch Wörterbuch. Die beiden Personen versuchen miteinander zu kommunizieren. Dazu nutzt der Deutsche das Wörterbuch. Nun stellt sich die Frage, ob der Deutsche, wenn er seinen Satz auf Chinesisch übersetzt, wirklich weiß was er sagt. Übertragen auf eine Maschine stellt sich nun ebenfalls die Frage, ob die Maschine wirklich wissen kann was sie tut und diese Aktionen auch selber hinterfragen kann. Darüber hinaus stellt sich die Frage, wie intelligent Maschinen wirklich werden können.

5.3 Anwendungen in der Moderne

Hier folgt eine kurze Beleuchtung des heutigen Einsatzfeldes des Turingtests. Dieser wird in der Internetsicherheit sehr oft verwendet um automatisierte Anfragen zu filtern. Als Beispiel wird hier exemplarisch ReCaptcha von Google beleuchtet. Es gibt aber genug andere sogenannte Captcha Methoden. Diese geben eine Art von Frage an, hierbei ist die Kommunikation allerdings nicht nur wie im Turing Test, auf textbasierte Kommunikation beschränkt. Im Gegenteil, hier werden sogar bewusst Bilder eingesetzt, damit der zu testende Computer, beziehungsweise die Person die den Computer bedient, eine Frage zu diesen Bildern beantworten muss.

Dabei sind die Antworten auf die eigentliche Frage nicht so wichtig wie es einem im ersten Moment erscheint. Die Ironie, und eine weitere Abweichung, ist dabei, dass man hier gegen ein neuronales Netzwerk "spielt". Dieses muss anhand der vorliegenden Informationen bestimmen, ob die Anfrage von einem Mensch stammt oder ob es sich dabei nur um eine automatisch generierte Anfrage handelt. Wie genau dieses Netzwerk entscheidet, weiß wahrscheinlich nicht mal Google. Wichtig ist aber, welche Daten dem Netzwerk dabei zur Verfügung gestellt werden, um seine Entscheidung zu treffen. Für weitere Informationen verweise ich hier auf einen Blog-Eintrag [8]. Zu den auszuwertenden Daten gehören unter anderem die Mausbewegungen, sowie die gesendeten Daten des Browsers (z.B. User-Agent, IP, Cookies).

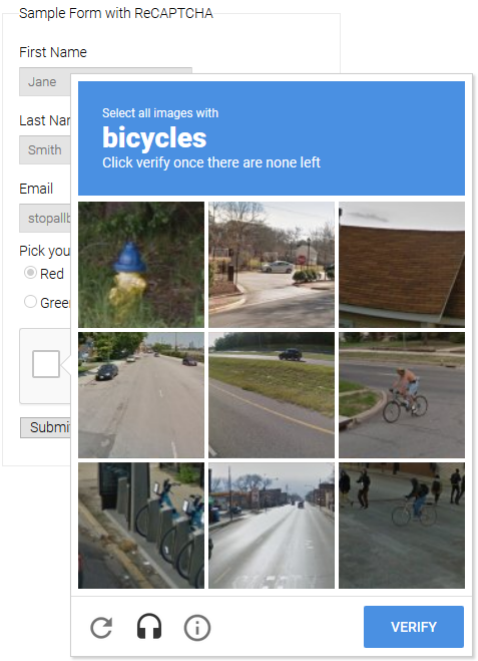

Abbildung 7: ReCaptcha Test von Google [7]

5.4 Auswirkungen auf Kunst und Gesellschaft

Diese grundsätzlichen Gedanken und die Frage, ob Maschinen überhaupt denken können, hat weitreichende Auswirkung auf die moderne Kunst, darunter vor allem Computerspiele. Sehr gute Beispiele sind das gleichnamige Spiel *"The Turing Test"* oder auch *"The Talos Principle"*, in denen der Turing Test sehr häufig vorkommt. In *"The Turing Test"* geht es weiter folgend um die Fragen: Was ist wenn Mensch und Maschine im selben Körper sind? Hier wird ein klassischer textbasierter Turingtest eingesetzt, um Person und KI zu testen. Der Plot an dem ganzen Spiel ist, dass die Protagonistin nicht

weiß, dass die KI teil von ihr ist und sie beeinflusst. Dieses Szenario erscheint erst einmal sehr kurios, wenn man aber einmal bedenkt, wie fortschrittlich unsere Implantattechnik bereits ist und wie diese weiter ausgebaut wird. In *"The Talos Principle"* geht es um die Frage, ob wir Menschen nicht auch Maschinen sind und inwieweit Glauben, Neugier, und Unterhaltung uns von Maschinen abhebt. Deshalb ist die Frage "Was macht das mit uns Menschen und mit der KI" berechtigt. Es ist wichtig hier darauf aufmerksam zu machen, dass das Imitation Game heute noch für sehr viel Kopfzerbrechen unter den Wissenschaftlern, Philosophen, sowie in der gesamten Gesellschaft auslöst. Wir werden bald an den Zeitpunkt kommen wo es wichtig sein wird, sich mit dieser Frage auseinander zu setzen. Weiterführend dazu siehe "The future of the mind: Exploring machine consciousness" von Dr. Susan Schneider. [9]

6 P versus NP

P versus NP ist ein ungelöstes Problem der Mathematik wie auch der theoretischen Informatik, es behandelt die Frage, ob Probleme die man einfach überprüfen kann, auch einfach zu lösen sind oder auch ob P das Gleiche ist wie NP. Hierbei steht das P für polynomiell und NP für nicht-deterministisch polynomiell. Dieses Problem gehört zu den Millennium-Problemen und eine Lösung dieses Problems wird mit einer Millionen US-Dollar vom Clay Mathematics Institute in Cambridge belohnt.

6.1 Beispiel

Ein NP Problem für eine deterministische Turingmaschine ist das Erraten eines mehrstelligen Passworts. Hierbei muss die Turingmaschine jede einzelne Kombination durchgehen und somit wird die benötigte Zeit um das Passwort zu erraten mit der Länge des Passworts auch exponentiell länger. Nehmen wir mal an, dass ein Passwort ohne Berücksichtigung von Groß- und Kleinbuchstaben nur die 26 Buchstaben des Alphabets zur Verfügung hat. Somit hätten wir bei einem Passwort mit der Länge eines Buchstaben 26 Möglichkeiten, bei einer Länge von zwei Buchstaben bereits 676 Möglichkeiten und bei einer Länge von zehn Buchstaben mehr als 141 Billionen Möglichkeiten.
Dagegen wird das Ganze zu einem P Problem, wenn das Passwort bekannt ist und nur noch die Richtigkeit überprüft werden muss. Diese Dauer der Überprüfung ist linear und somit für eine Turingmaschine in einer relativen kurzen Zeit zu erledigen.

6.2 Was wäre wenn P = NP

Würde jemand beweisen, dass P dasselbe ist wie NP, wäre die asymmetrische Kryptografie oder auch Public-Key-Kryptografie hinfällig, da diese auf dem

Konzept beruht, das P nicht dasselbe ist wie NP. Somit wären die meisten elektronischen Sicherheitssysteme einfach zu knacken, da man sie dann in einer polynomischen Zeitspanne lösen könnte. Andererseits würde es auch dafür sorgen, dass die Fahrwegberechnungen zwischen verschiedenen Punkten ebenfalls polynomiell werden und somit in kürzerer Zeit effizientere Fahrwege berechnet werden können.

7 Lambda calculus

Der Lambda calculus ist ein Datenverarbeitungskonzept erfunden von Alonzo Church, dem Professor von Alan Turing. Turing studierte unter Church einige Zeit an der Princeton Universität bevor er dann wieder zurück nach England ging. Im folgenden Abschnitt wird kurz auf Church eingegangen, danach wird Churches Werk, der Lambda calculus und dessen Gesetze, Notation und Beispiele behandelt. Hier wird ein Einblick in die Welt der funktionellen Programmiersprachen und ihrer Geschichte gegeben.

7.1 Alonzo Church

Alonzo Church wurde am 14 Juni 1903 in Washington DC geboren. [10] Nach 1920 ging er nach Princeton. In den 1930gern ging es in Princeton rund. Es waren viele Spezialisten der Logik dort, darunter auch John von Neumann und Kurt Gödel. Gegen 1936 kam auch Alan Turing als Gaststudent an die Universität. Dort kreuzten sich ihre Wege und so entstand die Church-Turing Thesis. Diese wird im nächsten Kapitel beschrieben.

7.2 Gesetze des Lambda calculus

Der Lambda calculus besteht aus 3 Grundgesetzen.

- Funktionen können definiert werden.
- diese können/müssen Daten ein und ausgeben.
- Funktionen können angewendet werden.

Ausführlicher: Es muss ein Weg vorhanden sein eine Funktion zu definieren, dafür gibt es eine mathematische Definition. Die Erzeugung einer Funktion ist aber nicht auf die mathematische Notation beschränkt, ganz im Gegenteil, die Notationen im modernen Lambda weichen, nicht zuletzt aus technischen Gründen, stark von der mathematischen Notation ab. Dabei

müssen im originalen Lambda Ein- und Ausgaben definiert werden. Dies weicht im modernen Lambda ebenfalls von der originalen Definition ab. Hier können Ein- und Ausgaben definiert werden, müssen es aber nicht. Dies führt zu mehr Flexibilität, so können die Lambda Funktionen auch an Stellen eingesetzt werden, wo nicht direkt Daten verarbeitet werden, sondern zum Beispiel Signale. Des weiteren kennt man beim Ausführen der Funktion den Inhalt der Funktion nicht. Um nun damit Daten verarbeiten zu können, kann man nun Eingaben auf Funktionen anwenden. [11]

7.3 Mathematische Notation

Für den ursprünglichen Lambda calculus definiert von Alonzo Church gibt es auch eine mathematische Notation die wie folgt durch das Zeichen Lambda eingeleitet wird.

$$\lambda x.x + 1 \tag{1}$$

Diese Funktion erhöht die Eingabe um eins. Dabei ist es wichtig anzumerken, dass wir nicht wissen was die Funktion wirklich tut. Wir können nur sagen was in die Funktion herein gegeben wird und was beim anwenden der Funktion heraus kommt. Wobei die Inputs mit dem λ (lambda) Symbol gekennzeichnet sind und mit einem Punkt enden. Am Schluss folgt die Ausgabe. Die Funktion könnte beim ausführen $x+2-1$ rechnen oder $x+4-3$ das kann man in diesem Model nicht beschreiben, klar ist nur es kommt immer $x+1$ heraus. Die ursprüngliche Notation ist weit aus mathematischer als die in derzeitigen Programmiersprachen. Dennoch lassen sich damit beliebig Daten durch das sogenannte Kodieren, wie im folgenden Beispiel, verarbeiten. Gegeben sind zwei Funktionen.

$$TRUE = \lambda x.\lambda y.x \tag{2}$$

In der originalen Notation wird der Punkt und das zweite Lambda bei den Inputs weggelassen. Wir bleiben aber bei der heutig genutzten Version der

Übersichtlichkeit halber. [12]

$$TRUE = \lambda xy.x \tag{3}$$

$$FALSE = \lambda x.\lambda y.y \tag{4}$$

Diese zwei Funktionen sollen nun zu einem logischen Und-Gatter zusammen gefügt werden. Dabei entsteht folgende Funktion.

$$AND = \lambda x.\lambda y.xyx \tag{5}$$

Nun wird auf die Und-Funktion als Beispiel die Funktionen TRUE und FALSE angewendet. Dazu werden einfach die variablen mit den als Eingabe verwendeten Funktionen ausgetauscht. In diesem konkreten Beispiel wird TRUE und TRUE auf die Funktion AND angewendet

$$AND\ TRUE\ TRUE = (\lambda x.\lambda y.xyx)TRUE\ TRUE \tag{6}$$

Hier wird detailliert aufgeschrieben wie man die Funktionen auflöst. Man nimmt immer die Funktion die am weitesten links steht und schreibt auf wie sie aufgebaut ist. Die nachfolgenden Methoden beziehungsweise Daten werden nun als Input verwendet, in diesem Fall natürlich die kodierten Funktionen. In diesem Fall ist die Funktion AND definiert als xyx und gegeben sind die Inputs $x = TRUE$ und $y = TRUE$. Dies wird nun ebenfalls hingeschrieben und es wird weiter aufgelöst.

$$TRUE\ TRUE\ TRUE = (\lambda x.\lambda y.x)TRUE\ TRUE = TRUE \tag{7}$$

Nun wird auf die ganz links stehende Funktion TRUE die Inputs $x = TRUE$ und $y = TRUE$ angewendet. Da sich TRUE immer zu x auflöst ist das Ergebnis also AND TRUE TRUE = TRUE. Dies bestätigt die Annahme. Um ein Gegenbeispiel zu bringen wird im nächsten Beispiel FALSE und TRUE übergeben.

$$AND\ FALSE\ TRUE = FALSE\ TRUE\ FALSE = FALSE \tag{8}$$

Und auch hier funktioniert die Und-Funktion wie gedacht. So kann man alle möglichen Arten von Datentransformation in Lambda Funktionen darstellen. [11]

7.4 Lambda in Programmiersprachen

Schauen wir uns nun zur weiteren Erläuterung ein Beispiel aus Java an. Diese Beispiele sind weit aus moderner und weichen in Teilen von der originalen Definition ab, beziehungsweise fügen neue Regeln hinzu. Es gibt eine Funktionsdeklaration die der Implementation vorgibt, welche Ein und Ausgänge eine gewisse Lambda Funktion hat, sowie ihren Namen.

```
@FunctionalInterface
public interface TestFunctionalInterface {
        public int test(double d);
}
```

Die Funktion wird nun als Interface übergeben und von einer ausführenden Funktion genutzt. Was hier auffällt ist das die ausführende Funktion nicht weiß, was die Funktion eigentlich tut.

```
public void runTest(TestFunctionalInterface interf) {
        int testres = interf.test(89.1);
        // Mache etwas mit dem resultat
}
```

Normalerweise müsste man dieses Interface überschreiben und dort dann implementieren was die Funktion tun würde. Hier gibt es allerdings eine separate Schreibweise um eine Funktion zu definieren. Die wie folgt durch die Zeichenkombination $->$ eingeleitet wird. Rechts von dem sog. Lambda-Operator stehen die Inputs, rechts die Operationen bzw. Outputs. Im folgenden Beispiel wird die gerade erzeugte Funktion auf *runTest* angewendet.

```
public void main() {
        runTest(input -> (int)(input + 1));
}
```

Und nicht nur Java hat diese Art Paradigma übernommen, auch C++ zum Beispiel, hat zusätzlich zu den bereits vorhandenen Funktionspointern, Lambda Funktionen eingeführt. In C++ haben die Lambda Funktionen eine leicht veränderte Anwendungsstruktur.

```
playercontroller = [](Input* input) {
topdown.positiony += input->y1;
topdown.positionx -= input->x1;
setTopDownCamera(&topdown);
};
```

Hier wird eine Player-Controller-Funktion in meiner Engine definiert. Der Player-Controller bekommt die letzten Inputs des Players in die Funktion und verarbeitet diese dann weiter.

8 Church-Turing Thesis

In diesem Kapitel geht es um die Church-Turing Thesis, was sie besagt und was für Auswirkungen sie hat. Außerdem wird im Zuge dessen die Turing Maschine und der Lambda calculus gegenübergestellt.

8.1 Die These

Die These besagt, dass alle möglichen Rechnungen beziehungsweise Datenverarbeitungen mit dem Modell der Turing Maschine dargestellt werden können. Des weiteren ist der Lambda calculus mit der Turing Maschine gleich zu stellen, das heißt, dass ebenfalls jeder Datensatz mit Hilfe dieser Paradigmen verarbeitet werden können. Beide Systeme sind also nur unterschiedliche Darstellungsweisen für Rechnungen beziehungsweise Datenverarbeitung. [13]

8.2 Gegenüberstellung

Wie bereits erwähnt hat der Lambda calculus keinen internen "State", dies steht im Kontrast zur Turing Maschine, diese hat einen internen "Speicher". Churchs Theorie basierte eher auf der Verarbeitung von ganzen Zahlen bei der Speicherbetrachtung nicht im Vordergrund war. Damit war er allerdings ein paar Jahre zu früh, denn die Turing Maschine entwickelte sich schnell zum Standardmodell. Dies ist primär dem geschuldet, dass die Betrachtung von Speichern zu dieser Zeit und bis heute, ein wichtiger Bestandteil der Informatik beziehungsweise der logischen Mathematik sind. Tatsächlich erkannte Gödel den Lambda calculus erst an, als er das Papier zur Turing Maschine sah und die Äquivalenz beider Systeme erkannte. [13] In der heutigen Zeit wiederum wird durch zunehmende Abstraktion und Komplexität der Lambda calculus wieder wichtiger. Er taucht vermehrt in den gängigen Programmiersprachen auf. So bekamen Java, C# und C++ in der letzten

Zeit vermehrt Lambda Unterstützung. Dies ist den immer komplexer werdenden Problemen geschuldet, da man sich bei der Nutzung von Lamda keine Gedanken über den "Heap" beziehungsweise die Speicher an sich machen muss. Dies kann die Arbeit und Komplexität sowie Fehleranfälligkeit reduzieren.

8.3 Beispiele

Widmen wir uns also dem oben bereits gezeigten Beispiel des Hochzählens. Mit Funktionen würden wir dies in JavaScript wie folgt schreiben.

```
incrementor = (x) => incrementor(x + 1)
incrementor(0)
```

Wie man hier schön sehen kann muss man sich nicht, wie bei der Turing Maschine, um Speicher kümmern, was diese Darstellung unabhängiger vom eigentlichen Speichersystem macht. Wie dem auch sei, da Rekursion, also das selbst aufrufen einer Funktion, komplizierter ist als es hier aussieht, wird diese Funktion zwangsläufig irgendwann einen Fehler werfen. Bei der klassischen Methode jedoch ist es nicht so, hier ist der Hochzählvorgang lediglich durch die eigentlichen Speicher begrenzt.

```
int i = 0;
while(true) {
        i++;
}
```

Wenn wir uns nun ein Gegenbeispiel anschauen. In diesem Fall versucht man eine dynamische Liste zu sortieren. Folgend wird hier die Turing Methode genutzt, dabei wird ein Standard Bubbl-Sort Algorithmus eingesetzt.

```
int temp;
for (int i = 1; i < arrayList.size(); i++) {
        for (int j = 0; j < arrayList.size() - i; j++) {
                if (arrayList.get(j) > arrayList.get(j + 1)) {
                        temp = arrayList.get(j);
                        arrayList.set(j, arrayList.get(j + 1))
                        arrayList.set(j + 1, temp);
                }
        }
}
```

Nun schauen wir uns dazu die Lambda Variante an.

```
ArrayList<Integer> arrayList = new ArrayList<Integer>();
arrayList.sort((x, y) -> x - y);
```

Wie man sieht ist dies viel simpler, da es hier keinerlei Darstellung von Speichern und dem eigentlichen Algorithmus bedarf. Man kann zwar den Algorithmus auch mit Lambda-Funktionen genauer darstellen. Dies ist allerdings nicht nötig, denn der Lambda calculus, hat hier den entscheidenden Vorteil. Durch die Turing Maschine lassen sich keine Abstraktionen modellieren, dieses Problem hat der Lambda calculus nicht.

Bildquellen

Abbildung 1: https://blog.sciencemuseum.org.uk/wp-content/uploads/2013/12/Alan-Turing-29-March-1951-picture-credit-NPL-Archive-Science-Museum1.jpg

Abbildung 2: https://d1e4pidl3fu268.cloudfront.net/10fd7805-c5e4- 42ac-8f67-bc7d0a42e6d0/enigma_machine.crop_720x540_68%2C0.preview.jpg

Abbildung 3: https://de.wikipedia.org/wiki/Enigma-Walzen# /media/Datei:Enigma_rotor_exploded_view.png

Abbildung 4: http://enigmamuseum.com/wp-content/uploads/2016/12/WiringDiagram-1024x776.jpg

Abbildung7: Selbst aufgenommen

References

[1] P. D. Brailsford, "Turing's enigma problem (part 1) - computerphile." `https://www.youtube.com/watch?v=d2NWPG2gB_A`, 2014. Aufgerufen 18.9.2019.

[2] B. Copeland, *The Essential Turing*. Clarendon Press, 2004.

[3] D. J. Grime, "158,962,555,217,826,360,000 (enigma machine) - numberphile." `https://www.youtube.com/watch?v=G2_Q9FoD-oQ`, 2013. Aufgerufen 28.02.2020.

[4] D. J. Grime, "Flaw in the enigma code - numberphile." `https://www.youtube.com/watch?v=V4V2bpZlqx8`, 2013.

[5] F. Friederes, "The enigma machine: Encrypt and decrypt." `https://cryptii.com/pipes/enigma-machine`, 2013. Aufgerufen 28.02.2020.

[6] A.M.Turing, "Computing machinery and intelligence." https://www.csee.umbc.edu/courses/471/papers/turing.pdf, 1950. Aufgerufen 19.9.2019.

[7] Google, "Recaptcha demo." https://www.google.com/recaptcha/api2/demo. Aufgerufen 09.05.2020.

[8] R. D. Peter Schmitz, "Wie funktioniert ein captcha?." https://www.security-insider.de/wie-funktioniert-ein-captcha-a-683209/, 2018. Aufgerufen 26.11.2019.

[9] D. S. Schneider, "The future of the mind: Exploring machine consciousness." https://www.youtube.com/watch?v=VypuNfR2GE0, 2019. Aufgerufen 23.11.2019.

[10] H.B.E, "Alonzo church: Life and work." https://web.archive.org/web/20120901152639/https://www.math.ucla.edu/~hbe/church.pdf, 2012. Aufgerufen 13.10.2019.

[11] P. G. Hutton, "Lambda calculus - computerphile." https://www.youtube.com/watch?v=eis11j_iGMs, 2017. Aufgerufen 24.9.2019.

[12] A. Church, *The Calculi of Lambda Conversion. (AM-6).* Annals of Mathematics Studies, Princeton University Press, 1951.

[13] B. J. Copeland, "The church-turing thesis," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), Metaphysics Research Lab, Stanford University, spring 2019 ed., 2019.