TELEPOLIS

Zuse und Turing: Der Draht des Mephistopheles

Raul Rojas *21.12.2011*

Hundert Jahre Alan Turing

Konrad Zuse wurde bereits 2010 gefeiert - 2012 wird es nun das Alan-Turing-Jahr werden. Beide Pioniere des Computers waren in der gleich Zeitepoche aktiv, lernten aber ihre jeweiligen Arbeiten erst nach dem Zweiten Weltkrieg kennen. Die Anzahl der Tagungen und Events im Alan-Turing-Jahr verspricht alle bisherigen Rekorde zu brechen. Überall auf der Welt wird das Leben und Werk des englischen Wissenschaftlers zelebriert. Aufgrund der Zahl der Ankündigungen platzt eine Webseite der Universität Leeds bereits aus allen Nähten.[1]

Alan Turing hat in seinem verhältnismäßig kurzen Leben viel geleistet, sowohl auf verschiedensten Gebieten der Mathematik als auch in der Informatik. Die Maschine, die seinem Namen trägt, ist ein rein theoretisches Konstrukt, eine "virtuelle Maschine", die jedoch trotzdem so intuitiv einfach ist, dass bis heute kaum ein anschaulicheres Beispiel für die Bedeutung von Universalität beim Rechnen vorgeschlagen worden konnte. Universalität bedeutet, dass mit dieser Maschine im Prinzip jede mechanisierbare Rechenvorschrift implementiert werden kann: Kann es mit einem Computer gemacht werden, so ist es auch mit einer Turing-Maschine möglich.

Eine solche Maschine ist konzeptuell übersichtlich: Sie besteht aus einem unendlichen linearen Speicher, wo in jeder Speicherzelle ein Symbol untergebracht werden kann (man kann diese Symbole binär codieren). Ein Schreib-Lese-Kopf kann jeweils ein Symbol lesen bzw. überschreiben. Ein "Prozessor" wechselt seinen Zustand, je nachdem welches Symbol zuletzt gelesen wurde, und darf mit einem Ausgabe-Symbol den Speicher überschreiben. Der Schreib-Lese-Kopf bewegt sich danach um eine Stelle nach links oder nach rechts (auf die nächste Speicherzelle) - je nachdem, was wir berechnen wollen. Der Prozess wiederholt sich so lange bis die Maschine eventuell stoppt.

Es gibt viele verschiedene Ausprägungen der Turing-Maschine, aber dieser Zyklus ist ihnen gemeinsam: Symbol lesen, je nach Prozessorzustand zu einem anderen Zustand schalten, Symbol ausgeben, Schreib-Lesekopf bewegen, Zyklus wiederholen (bis eine Haltebedingung erreicht wird).

Vergleicht man diese Abfolge mit dem eines Computers, so ist alles sehr ähnlich. Ein moderner Rechner besitzt ebenfalls einen Prozessor mit internen Daten (der Prozessorzustand). Der nächste Befehl liest interne oder externe Daten (Register bzw. Speicher), führt eine Operation aus und ändert damit den Prozessorzustand. Das Resultat kann erneut intern oder extern gespeichert werden und danach geht es mit dem nächsten Befehl wieder los. Der wesentliche Unterschied zur Turing-Maschine ist nur, dass moderne Computer adressierbare Speicher haben. So laden wir zum Beispiel Daten aus Adresse 100 oder Adresse 101. Eine Turing-Maschine dagegen lädt Daten nur lokal, aus der Position an welche der Schreib-Lesekopf sich befindet. Man braucht deswegen keine Speicheradresse anzugeben, diese ist implizit durch die Position des Lesekopfes festgelegt. Jeder, der eine Turing-Maschine programmiert hat, weiß, wie umständlich dies für die Programmierung sein kann, aber wie kompakt die

Beschreibung der Turing-Maschine damit wird.

Turing hat ebenfalls gezeigt, dass eine spezielle Turing-Maschine die Beschreibung einer beliebigen anderen Turing-Maschine vom Leseband ablesen und ausführen kann. Es verhält sich wie Hardware und Software: die Universelle Turing Maschine (UTM) ist diese Sondermaschine, die die Beschreibung (das Programm) einer durchführbaren Berechnung lesen und ausführen kann. Man kann universelle Turing-Maschinen mit unterschiedlicher Anzahl von Symbolen und Zuständen definieren und es ist eine wahre kleine Wissenschaft rund um die Frage entstanden, wie klein und kompakt die "ultimative" UTM sein kann.[2] Die UTM wurde anno 1986 fünfzig Jahre alt und bereits damals fand aus diesem Anlass eine Reihe von Veranstaltungen statt.[3]

Die Logistische Maschine von Konrad Zuse

Dies alles ist natürlich hinreichend bekannt und eingefleischte Telepolis-Leser (solche mit Piratentuch) lächeln bis hierher sicherlich nur müde. Was aber nicht so richtig bekannt ist, ist der Fakt, dass Konrad Zuse gegen Ende seiner wichtigsten Schaffensperiode zu der Definition von etwas Turing-Maschine-Ähnlichem gekommen ist, ohne allerdings die letzte Hürde zu überwinden, d.h. die Einführung der bedingten Sprünge. Ich rede von der "logistischen Maschine", ein theoretisches Konstrukt, welches Konrad Zuse um 1943 entwarf und Jahre danach zum Patent anmeldete. Wie wir gleich sehen werden, kann die logistische Maschine in einer Form verwendet werden, die sie trotzdem universell macht. So gesehen bekommt 2010 eine ungeahnte Verbindung zu 2012. Im Folgenden beschreibe ich die einfachste Ausprägung der LM, um danach die von Zuse spezielle verwendete Form zu erwähnen.

Konrad Zuses logistische Maschine ist noch überschaubarer zu beschreiben als eine Turing Maschine. Bild 2 zeigt die wesentlichen Komponenten. Es gibt einen addressierbaren Speicher, wo jede Zelle jedoch nur ein Bit enthält. Man kann den Speicher ab Adresse Null durchnummerieren, wobei jede Speicherzelle eine um Eins erhöhte Adresse erhält. Das Programm ist extern gespeichert (z.B. in einem Lochstreifen). Der Prozessor hat nur zwei 1-Bit-Register: Register A und Register B. Die einzigen Befehle lauten: "Load n", "Store m" und "NAND". Hier stehen die Zahlen n und m für Adressen im Speicher. Zusätzlich gibt es ein paar Vereinbarungen: Zuerst wird Register A geladen (er ist dann "besetzt"), danach Register B. Resultate landen in Register A (der "besetzt" bleibt, Register B wird gelöscht und kann danach erneut geladen werden). Der Store-Befehl bezieht sich also auf Register A, der danach als "unbesetzt" markiert wird. Und das ist schon alles!

Durch diesen Befehlssatz ist es möglich, Bits aus beliebiger Position im Speicher zu laden und miteinander zu kombinieren. In der Informatik ist bekannt, dass NAND von zwei Bits A und B (Negation der Konjunktion, d.h. die Operation "NICHT(A UND B)") eine logische Basis bildet. Das bedeutet, es ist möglich UND- bzw. ODER-Operationen, sowie Negationen und jede andere logische Bit-Operation mit NAND als einzigem Baustein zu berechnen. Anders gesagt: Ein vollständiger Computer kann aus hinreichend vielen NAND-Gattern gebaut werden.

Nur die Programmierung der logistischen Maschine ist umständlich. Will man zwei 32-Bit-Zahlen addieren, muss man Bit für Bit die Addition, die Übertrag-Bits und noch das Resultat berechnen. Es ist zwar möglich, aber das Programm wird am Ende hunderte Zeilen lang sein. Es ist umständlich, aber machbar.

Jede zyklenfreie logische Schaltung kann mit dieser Maschine emuliert werden. Man muss nur die Schaltung auf NAND-Gatter reduzieren. Da die Schaltung endlich ist, ist auch die Berechnung in einer endlichen Anzahl von Schritten möglich. Übliche Digital-Schaltungen sind getaktet, alles erfolgt in wohl definierten Zeitintervallen. Eine solche Digitalschaltung kann dann Gatter für Gatter mit der logistischen Maschine emuliert werden - die notwendige Reihenfolge der Operationen ist durch das Programm geregelt. Muss man ein Zwischenresultat der Schaltung für einen Moment festhalten, so kann man dieses Resultat vorläufig speichern.

Jetzt kann man sich folgendes vorstellen: Ein Prozessor ist nichts anderes als eine logische Schaltung. Der Prozessor führt immer denselben Zyklus durch (Daten rein, Berechnung, Daten raus). Dieser elementare Prozessorzyklus ist, was das Programm der logistischen Maschine implementieren kann. Wir benötigen dafür lediglich einen Lochstreifen mit geklebten Anfang und Ende, so dass dasselbe Programm immer wieder ausgeführt werden kann. Register eines Prozessors können mit dem externen Speicher emuliert werden und die Schaltung des Prozessors selbst kann Gatter für Gatter emuliert werden. Jedes Mal wenn der Lochstreifen zum Ende kommt, ist ein Prozessordurchgang fertig und dann geht es wieder von vorne los, da wir eine Programmschleife haben.

Es bleibt nur noch eine letzte Schwierigkeit: Im Computer muss man immer wieder mit Tabellen von Zahlen arbeiten und "indexiert" adressieren. D.h. eine Variable speichert die Adresse, aus der die nächste Zahl zu holen ist. In einem Lochstreifen kann man aber nur feste Adressen unterbringen. Die Lösung dieses Problems würde den Rahmen dieses Textes sprengen, man kann aber problemlos zeigen, dass eine Rechenmaschine, die die arithmetischen Operationen beherrscht und eine feste Schleife von arithmetischen Operationen durchführen kann, eine Turing-Maschine (mit einem durch den vorhandenen Speicher begrenzten Leseband) vollständig emulieren kann. Der Beweis ist exakt derselbe, den ich für den Beweis der Universalität der Z3 verwendet habe.[4]

Und nur zur Klarheit: Zuses ursprüngliche Definition der Logistischen Maschine kann im Prozessor die Befehle AND bzw. OR ausführen, wobei die Register A bzw. B vorher negiert werden dürfen. Dies erhöht zwar die Anzahl der Befehle, macht aber den Prozessor nur etwas komplizierter. Die hier besprochene Ausprägung des Prozessors ist einfacher als das Original, gibt aber den Geist der Definition anschaulicher wieder, da es sich um das Design einer minimalen Maschine für Sequenzen von arithmetischen Operationen handelt.

Somit steht aus dem Gesagten fest: die logistische Maschine kann alles tun, was ein Computer mit endlichen Speicher tun kann (und alle echte Computer haben endlichen Speicher). Der Preis, den wir für die Einfachheit des Prozessors zahlen, ist dass die Länge der notwendigen Programme (d.h. der Umfang der Software) beträchtlich wächst.

Der Draht des Mephistopheles

Der aufmerksame Leser kann nun einwenden, dass diese Maschine nie hält: Da der Lochstreifen zur Schleife gebunden wurde, wird das Programm ununterbrochen ausgeführt. Man könnte dagegen halten, dass unsere Rechner daheim auch die ganze Zeit laufen. Der Prozessor ist immer aktiv, heizt die Wohnung nebenbei und führt so wichtige Sachen aus, wie z.B. ein Bildschirmschoner oder SETI@Home.

Es gibt aber noch ein unumstößliches Problem mit all den Maschinen von Konrad Zuse. Die Z1, Z2, Z3 und Z4 wurden allesamt mit einem Lochstreifen gesteuert und nur feste Sequenzen von Operationen konnten damit ausgeführt werden. Zwischen 1938 (Fertigstellung der Z1) und 1945 (Fertigstellung der Z4) versuchte Zuse nie, den bedingten Sprung im Befehlssatz einzubauen. Dies ist umso erstaunlicher, da der Gedanke des bedingten Sprungs ("lebendige Rechenpläne") in seinen Notizbüchern häufig zu finden ist und in seinem Entwurf für einen "Plankalkül", der als erste Programmiersprache der Welt gelten kann, die bedingten Schleifen bereits fester Bestandteil des Befehlssatzes waren.[5] Noch Jahre danach hat Zuse wiederholt gesagt, dass nur ein Draht für den bedingten Sprung notwendig gewesen wäre, er aber von der "faustischen" Verselbständigung der Maschine immer wieder zurückgeschreckt sei.[6]

Diese psychologische Erklärung ist wenig glaubhaft. Wahrscheinlicher ist es, dass es Zuse damals vor allem auf die Berechnung von festen wiederkehrenden Formeln ankam, wie es bei der Erstellung von mathematischen Tabellen notwendig ist. Die große Gelegenheit für ein Umdenken wäre der Bau der Z4 gewesen (1941-45), um ein für alle Mal den bedingten Sprung im Befehlssatz zu integrieren. Dass dies nicht geschah, ist bedauerlich. Es ist das größte Manko der Zuse-Rechner, die ansonsten in puncto Architektur allen anderen britischen und amerikanischen Rechenmaschinen seiner Zeit weit überlegen waren.

Es gibt aber einen Ausweg, um die Logistische Maschine anzuhalten. Dafür musste man eine Null in eine spezielle Adresse speichern. Diese Sonderadresse hätte dann den Ausschalter für den Lochstreifenleser repräsentiert. Das Programm würde in jeder Iteration in diese Adresse eine 1 speichern, womit die Maschine nicht anhält - und nur zum Ende eine Null, um die Maschine zu stoppen. Diese kleine zusätzliche Schaltung ist alles, was man in der Logistischen Maschine einbauen müsste, um die Programmschleife zu beenden. Es wäre wie das Spielzeug von Claude Shannon gewesen: eine robotische Hand, die beim Einschalten des Geräts diese wieder ausschaltet. Nur ein Draht - aber was für ein Draht!

Die Wirkungsgeschichte von Zuse und Turing

Das Ausmaß der Veranstaltungen um das Turing-Jahr zeigt bereits, dass die Wirkungsgeschichte der zwei Informatik-Pioniere sehr unterschiedlich war. Konrad Zuse blieb viele Jahre außerhalb Deutschlands unbekannt. Erst in den siebziger und achtziger Jahren gab es erste Berichte über seine Rechenmaschinen in zahlreichen Büchern und Aufsätzen. Es ist auch klar warum: Obwohl Konrad Zuse einen deutlichen Vorsprung im Jahr 1941 bei der Vorstellung der Z3 besaß, schmolz dieses Polster während des Krieges dahin.

Als nach dem Konflikt die Alliierten Hunderte von deutschen Wissenschaftlern interviewten und gegebenenfalls zu sich "luden", wurde auch Konrad Zuse zu seinen Maschinen sowohl in Deutschland als auch in Großbritannien befragt. Weder die Briten noch die Amerikaner erkannten die Tragweite der Ideen von Zuse, wodurch er später keine maßgebliche Rolle mehr bei der Entstehung der Computerindustrie in den USA oder in Großbritannien spielen konnte. Die spätere Zuse KG war nie ein "global player" und musste letztendlich den Betrieb einstellen, als die amerikanischen Maschinen den europäischen Markt überrollten.

Die Turing-Maschine war hingegen eine globale Erfolgsgeschichte. Sehr früh wurde bewiesen, dass die TM äquivalent zu anderen Formalisierungen von Berechenbarkeit

war (wie z.B. dem Lambda-Kalkül von Church) und sie ist immer noch das bevorzugte Arbeitspferd für viele Berechenbarkeitsbeweise. Turing hat außerdem Beiträge in der mathematischen Logik geleistet, vielgelesene Artikel über Künstliche Intelligenz verfasst, eine Rechenmaschine in bit-serieller Architektur entworfen, während des Krieges die kryptographische Forschung der Briten bereichert, und so weiter und so fort. Turings tragischer freier Tod hat das übrige getan, um den genialen Mathematiker in eine Art Schutzpatron der Informatik zu verwandeln. Nicht umsonst ist der höchste Preis auf dem Gebiet der Informatik der begehrte "Turing Award", der Nobelpreis dieser Disziplin.

So gelangten Zuse und Turing, aus sehr unterschiedlichen Ecken kommend, zu vergleichbaren Interessen. Beide haben z.B. die ersten Programme für ein Schachspiel geschrieben, rein auf dem Papier, da sie keine Maschinen zum Ausführen hatten. Sie wurden beide in der Maschinerie des Krieges auf der jeweils gegnerischen Seite eingespannt. Ab 1945 galt für beide die selbstgesteckte Aufgabe der Entwicklung von Rechenmaschinen und maschineller Intelligenz. Und wie das Beispiel der logistischen Maschine zeigt, die im Übrigen auch kein theoretischer "global player" wurde, kamen beide auf verwandte Ideen, obwohl sie sich gar nicht kannten.

Anhang

Fußnoten

1) http://www.mathcomp.leeds.ac.uk/turing2012/[1]

2) M. Minsky, *Computation: Finite and Infinite Machines*, Prentice Hall, 1967.

3) R. Herken (Hrsg.), *The Universal Turing Machine: A Half-Century Survey*, Springer-Verlag, Zweite Ausgabe, 1995.

4) R. Rojas, "How to make Zuse's Z3 a Universal Computer", *Annals of the History of Computing*, V. 20, No. 3, 1998.

5) Zuse, Konrad, *Der Plankalkül*, Berichte der Gesellschaft für Mathematik und Datenverarbeitung, Nr. 63, Sankt Augustin, 1972.

6) K. Zuse, *Der Computer mein Lebenswerk*, Springer-Verlag, Berlin, 3. Auflage, 1993.

Links

[1] http://www.mathcomp.leeds.ac.uk/turing2012/