TOBY WALSH

IT'S ALIVE

Wie Künstliche Intelligenz
unser Leben verändern wird

Edition Körber

noch einmal die Menge aller silberfarbenen Autos. Diese Menge enthält sich nicht selbst. Nennen wir sie eine normale Menge. Betrachten wir nun die Komplementärmenge, die Menge, die alles enthält, was kein silbernes Auto ist. Diese Menge enthält sich selbst. Nennen wir das eine nichtnormale Menge. Nun betrachten wir die Menge aller normalen Mengen. Ist die Menge aller normalen Mengen selbst normal? Wenn sie normal ist, dann ist sie in der Menge aller normalen Mengen enthalten. Das heißt, sie wäre in sich selbst enthalten. Aber dann wäre es eine nichtnormale Menge. Betrachten wir nun die Alternative. Nehmen wir an, die Menge aller normalen Mengen ist selbst eine nichtnormale Menge. Als nichtnormale Menge wäre sie in sich selbst enthalten, der Menge aller normalen Mengen. Aber das macht sie zu einer normalen Menge. Daraus ergibt sich Russells berühmte Antinomie: Eine Menge kann nicht zugleich normal und nichtnormal sein. Wir werden an späterer Stelle unserer Geschichte auf ähnliche Widersprüche stoßen. Cantors Mengentheorie war so mit Widersprüchen durchsetzt, dass Kritiker ihn einen »wissenschaftlichen Scharlatan« und »Verderber der Jugend« nannten. Doch wie die nächste Figur in unserer Geschichte beweist, ist all das nicht Cantors Schuld. Es liegt am Wesen der Mathematik selbst. Und dies ist eine grundsätzliche Herausforderung für den Bau denkender Maschinen – zumindest denkender Maschinen, die mit Logik operieren.

Als Antwort auf diese sogenannte Grundlagenkrise der Mathematik formulierte Hilbert ein Arbeitsprogramm, um die Mathematik auf eine präzise, logische Grundlage zu stellen. Das sogenannte Hilbertprogramm zielte darauf ab, eine überschaubare Menge grundlegender Sätze oder Bausteine zu finden, aus denen sich das gesamte Gebäude der Mathematik errichten ließ. Das Hilbertprogramm setzte es sich auch zum Ziel, den Nachweis zu erbringen, dass diese Formalisierung der Mathematik nicht die Widersprüche von Cantors Mengenlehre wiederholt. Sobald sich nämlich Widersprüche einschleichen, lässt sich praktisch alles beweisen. Wenn wir denkende Maschinen bauen wollen – Maschinen, die unter anderem auch Mathematik betreiben –, dann brauchen wir eine solide Grundlage.

Das Ende der Mathematik

Im Jahr 1931 erhielt das Hilbertprogramm einen schweren Schlag durch Kurt Gödel, einen der wichtigsten Logiker der Geschichte.[39] Er bewies die Undurchführbahrkeit des Hilbertprogramms durch seine beiden berühmten Unvollständigkeitssätze. In ihnen erbrachte er den Beweis, dass jede Formalisierung der Mathematik, die hinreichend auch etwas so Einfaches wie die natürlichen Zahlen beschreibt, unausweichlich unvollständig ist und Widersprüche enthält. Daraus folgt, dass jedes widerspruchsfreie mathematische System auch mathematische Wahrheiten enthält, die sich nicht beweisen lassen, mit anderen Worten, *unvollständig* ist.

Gödels Befund versetzte dem Hilbertprogramm den Todesstoß und ließ die Mathematik für alle Zeiten auf wackligem Fundament zurück. Die Aufgabe, die Mathematik insgesamt mit mathematischer Genauigkeit zu beschreiben, erwies sich als undurchführbar. Und dies stellt eine große philosophische Herausforderung an den Traum vom Bau denkender Maschinen dar. Wenn wir in der Lage wären, denkende Maschinen zu bauen und wenn diese Maschinen mathematisch denken sollen, wie Leibniz und Hobbes es uns beschrieben haben, dann müssen wir ihnen eine präzise, logische Formalisierung der Mathematik geben, mit der sie denken können. Doch Gödels Unvollständigkeitssätze zeigen, dass wir für die Mathematik keine präzisen Regeln formulieren können, jedenfalls keine Regeln, die man einem Computer einprogrammieren könnte, damit er die *gesamte* Mathematik beherrscht.

Der mathematische Physiker Sir Roger Penrose[40] ist ein entschiedener Kritiker des Gedankens, dass die Künstliche Intelligenz eines Tages die menschliche Intelligenz übertreffen könne.[41] Doch nicht alle lassen seine Argumente gelten. Computer müssen ebenso wenig wie Menschen alle mathematischen Sätze beweisen können, um mit Mathematik umzugehen, und sie können genauso gut wie Menschen mit einem System umgehen, das Widersprüche enthält. Gödels Sätze beziehen sich auf unendliche Systeme, aber Menschen und Computer sind stets endlich. Außerdem werden Computer in nicht allzu ferner Zukunft andere Techniken verwenden; Quantencomputer gehen neue Wege, die über Gödels Resultate hinausreichen. Unter Fachleuten herrscht nach wie vor Konsens, dass Gödels

Unvollständigkeitssätze dem Traum vom Bau denkender Maschinen theoretisch nicht im Weg stehen. Und in der Praxis machen wir gute Fortschritte in Richtung KI, wie ich nun kurz darlegen will.

Was Computer nicht können

Das bringt uns fast wieder zum Anfang des Buches zurück, zu Alan Turing. Er leistete in Theorie und Praxis einen entscheidenden Beitrag zum Bau von Computern. Von ihm stammt das grundlegende abstrakte Modell, die Turing-Maschine, die bis zum heutigen Tag Computer mathematisch beschreibt. Doch schon 1936, bevor die erste dieser Maschinen überhaupt gebaut worden war, hatte Turing eine andere bemerkenswerte Erkenntnis. Er kam dahinter, dass es gewisse Probleme gibt, die sich durch Berechnungen mit solchen Maschinen *nie* lösen lassen. Dabei hatte zu diesem Zeitpunkt noch kein Mensch einen Computer programmiert – es gab schließlich noch keine. Trotzdem erkannte Turing glasklar, dass es bestimmte Probleme gibt, die auch der klügste Programmierer keinem Rechner einspeisen kann.

Eines davon ist das »Halteproblem«. Lässt sich ein Computerprogramm schreiben, das die Entscheidung darüber trifft, wann ein anderes Programm anhalten soll? Ein solches Programm wäre sehr nützlich. Natürlich will man nicht, dass die Steuerungssoftware eines Flugzeugs irgendwann ihre Arbeit einstellt, aber das Programm, das nach neuen Fernsehkanälen für unseren Receiver sucht, sollte vielleicht irgendwann damit aufhören. Und hier zeigte Turing, dass dies für einen Computer nicht immer entscheidbar ist.[42] Man muss bedenken, dass ein Programm an sich auch nur aus Daten besteht. Damit kann es als Input für andere Programme dienen. Wir könnten ein Programm schreiben, das ein anderes Programm als Input nimmt und entscheidet, ob dieses Programm jemals zum Halt kommt. Turing kam zu dem verblüffenden Ergebnis, dass kein noch so cleverer Programmierer einen solchen Algorithmus schreiben könnte.

Turing benutzte dazu einen Zirkelschluss, der an Russells Paradox über Mengen, die sich nicht selbst enthalten, erinnert. Nehmen wir an, wir verfügten über ein Programm, das das Halteproblem lösen könnte. Nennen wir es das Turing-Programm. Wir setzen es als Teilprogramm

eines größeren Programms ein, das wir das Super-Turing-Programm nennen wollen. Dieses Programm kann mit sämtlichen Programmen gefüttert werden und mithilfe des Turing-Programms als Unterprogramm entscheiden, ob das eingefütterte Programm irgendwann zum Ende kommt. Wenn das eingefütterte Programm anhält, geht das Super-Turing-Programm in eine unendliche Schleife, die niemals anhält. Wenn andererseits das Input-Programm nicht anhält, dann hält das Super-Turing-Programm an. Hier kommt der Zirkelschluss von Turing ins Spiel. Was macht das Super-Turing-Programm, wenn wir es mit sich selbst füttern? Das Super-Turing-Programm kann dann entweder anhalten oder nicht anhalten.

Betrachten wir die beiden Möglichkeiten. Angenommen, das Super-Turing-Programm hält bei diesem Input nicht an. Dann hält das Super-Turing-Programm nicht bei einem Input an, der anhält. Das bedeutet, dass das Super-Turing-Programm anhält. Wenn das Super-Turing-Programm also nicht anhält, dann heißt das, dass es anhält. Angenommen, das Super-Turing-Programm hält bei diesem Input an. Nun hält das Super-Turing-Programm an, wenn es als Input ein Programm erhält, das nicht anhält. Dies heißt, dass das Super-Turing-Programm nicht anhalten würde. In beiden Fällen kommen wir zu einem Widerspruch. Das Super-Turing-Programm kann nicht zugleich anhalten und nicht anhalten. Dabei bin ich von einer entscheidenden Annahme ausgegangen, nämlich dass das Turing-Programm existiert. Wir können daraus schlussfolgern, dass Turings Programm nicht existieren kann. Es kann also kein Programm geben, das entscheiden kann, ob jedes beliebige Programm, mit dem man es als Input füttert, anhält. Mit diesem Gedankenexperiment konnte Turing nachweisen, dass es Probleme gibt, die Computer nicht lösen können.

Dies kann als ein weiteres grundsätzliches Hindernis auf dem Weg zu denkenden Maschinen betrachtet werden. Wir haben den eindeutigen Beweis, dass es Dinge gibt, die Computer nicht können.[43] Die Frage ist nun, ob sich diese Unentscheidbarkeit nur auf solch abgelegene Probleme bezieht wie das, ob ein Programm anhalten kann oder nicht. Die Antwort ist, es gibt viele weitere, sehr viel praxisnähere Probleme, die nicht durch Computer gelöst werden können, beispielsweise die Entscheidung, ob eine mathematische Aussage richtig oder falsch ist.[44]

Aus mehreren Gründen bedeutet die Existenz von Problemen, die kein Computer zu lösen vermag, nicht das Aus für den Traum von denkenden Maschinen. Erstens kann es immer noch Computerprogramme geben, die solche Probleme lösen, wenn auch vielleicht nicht alle. Schon jetzt kann man Software wie Mathematica oder Maple kaufen, die sehr oft entscheiden kann, ob mathematische Aussagen wahr oder falsch sind, auch wenn diese Programme manchmal als Ergebnis nur »Ich weiß es nicht« parat haben. Zweitens, gewissermaßen hilfsweise, wäre immer noch ein Computerprogramm möglich, das solche Probleme zwar im Prinzip löst, dessen Resultate allerdings eine Spur Unsicherheit enthalten. Die Software kann sich also gelegentlich irren, wenn sie zu dem Ergebnis kommt, dass das eingespeiste Programm anhält, es aber in Wahrheit nicht tut, und umgekehrt. Drittens sind Heuristiken, also Faustregeln, die oft ausreichend genau sind, aber manchmal eben zu Fehlern führen oder auch in Endlosschleifen münden, ein großes Arbeitsgebiet der Künstlichen Intelligenz. Viertens ergeben menschliche Denkprozesse normalerweise ohnehin keine hundertprozentige Richtigkeit. Wenn man eins mit absoluter Sicherheit sagen kann, dann, dass Menschen nicht immer zu 100 Prozent richtigliegen. Wir müssen auch keine Maschine bauen, die jederzeit Turings Halteproblem lösen kann. Folglich muss der Traum vom Bau denkender Maschinen nicht an der Tatsache scheitern, dass wir Probleme kennen, die Computer nicht präzise lösen können.

Der Beginn des Computerzeitalters

Als Turing sich damit beschäftigte, was Computer nicht berechnen können, gab es noch gar keine Computer, die überhaupt irgendetwas berechnen konnten. Der Traum vom Bau denkender Maschinen war zu diesem Zeitpunkt noch reine Theorie. Das änderte sich erst mit dem Zweiten Weltkrieg. Die Notwendigkeit, den feindlichen Nachrichtenverkehr zu entschlüsseln, und die umfangreichen Berechnungen, die zum Bau der Atombombe notwendig waren, gaben der praktischen Seite der Computertechnik einen gewaltigen Schub und führten zum Bau der ersten nutzbaren Computer.