**Zeitschrift:** Mitteilungen / Vereinigung Schweizerischer Versicherungsmathematiker = Bulletin / Association des Actuaires Suisses = Bulletin / Association of Swiss Actuaries

**Herausgeber:** Vereinigung Schweizerischer Versicherungsmathematiker

**Band:** 61 (1961)

**Artikel:** Simulation von Intelligenz durch Maschinen

**Autor:** Leepin, P.

**DOI:** https://doi.org/10.5169/seals-966730

# Simulation von Intelligenz durch Maschinen

*Von P. Leepin, Arlesheim*

## Zusammenfassung

Nach einer kurzen Darstellung der prinzipiellen Arbeitsweise elektronischer programmgespeicherter Rechenmaschinen wird anhand einiger Beispiele gezeigt, wieweit sich heute Intelligenz durch Maschinen simulieren lässt.

Die erfolgreiche Verwendung elektronischer Geräte für Aufgaben auf den verschiedensten Gebieten führt dazu, dass immer wieder die Frage gestellt wird, ob diese Maschinen denken können. Sehr häufig wird eine negative Antwort erteilt, ohne sich mit dem Problem näher zu befassen. Es ist das Ziel dieser Arbeit zu zeigen, dass die Frage des Verhältnisses von Mensch und Maschine inbezug auf Denkprozesse etwas tiefer liegt, als es auf den ersten Anhieb scheint.

Wir müssen uns allerdings vorweg von allen animistischen Vorstellungen distanzieren, wie sie etwa in dem Ausdruck «Elektronenhirn» sich zeigen. Eine Maschine, die denkähnliche Vorgänge ausführt, stellt durchaus nichts Geheimnisvolles dar, und das Prinzip ihrer Arbeitsweise lässt sich von jedem interessierten Laien ohne allzu grossen Aufwand an geistiger Anstrengung verstehen. Eine Verwandtschaft mit dem menschlichen Gehirn besteht höchstens in bescheidenem Ausmass. Es darf da etwa auf die Untersuchungen des grossen Mathematikers Johann von Neumann hingewiesen werden, der zum Ergebnis gelangte, dass mit grosser Wahrscheinlichkeit das menschliche Gehirn trotz einiger Ähnlichkeiten in wesentlichen Punkten anders gebaut ist als jede bisher konstruierte Maschine. Wenn wir also animistische Vorstellungen im Zusammenhang mit Maschinen durchaus ablehnen, lässt es sich allerdings der Anschaulichkeit wegen nicht ganz vermeiden, dass gelegentlich

eine Ausdrucksweise verwendet wird, die in dieser Beziehung nicht ganz konsequent ist.

Zu der Frage, ob Maschinen denken können, möchten wir im jetzigen Augenblick nur folgendes bemerken: Eine Antwort lässt sich erst geben, wenn zwei Vorfragen gelöst sind:

1. Was müsste eine Maschine leisten, damit wir ihr Denkfähigkeit zuerkennen?

2. Wo liegen die Grenzen der Leistungsfähigkeit von Maschinen?

Auf die erste Frage, was unter Denken zu verstehen ist, soll hier nicht näher eingegangen werden. Wir möchten nur zeigen, dass die Antwort nicht so einfach zu geben ist, indem wir fragen: Denken wir, wenn wir zwei ungleichnamige Brüche addieren? Das ist eine Tätigkeit, die einer Maschine durchaus keine Schwierigkeiten bereitet.

Über die zweite Frage, wo die Grenzen der Möglichkeiten von Maschinen liegen, ist es im jetzigen Zeitpnkt nicht möglich, eindeutige Angaben zu machen, die voll befriedigen. Auf alle Fälle ist deutlich festzuhalten, dass die Behauptung: «Eine Maschine kann nicht denken» unwissenschaftlich ist, sofern nicht klare Vorstellungen darüber bestehen, was für geistige Tätigkeiten mit dem Wort «Denken» erfasst werden sollen und was eine Maschine tatsächlich leisten kann.

Unter diesen Umständen soll die Frage offengelassen werden, ob Maschinen denken können oder nicht. Für denkähnliche Vorgänge in Maschinen wird deshalb nachstehend der Ausdruck «Simulation von Intelligenz» oder «künstliche Intelligenz» gebraucht, in Anlehnung an die im englischen Sprachgebiet gut eingeführte Wendung «artificial intelligence».

Vorerst ist es notwendig, sich kurz mit der prinzipiellen Arbeitsweise von Maschinen zu befassen, die in der Lage sind, Intelligenz zu simulieren. Wir beschränken uns dabei auf die heute im Vordergrund stehende Kategorie der elektronischen programmgespeicherten digitalen Rechenautomaten. «Digital» bedeutet in diesem Zusammenhang, dass die zu verarbeitenden Angaben – seien es nun Zahlen oder Buchstaben oder andere Symbole – in der Maschine durch das Vorhandensein oder Nichtvorhandensein physikalischer Grössen in geeigneter Kombination verschlüsselt wird. Den Gegensatz zu digitalen Geräten bilden die analogen, welche ziffernmässige Angaben stetig auf physikalische Grössen abbilden, sodass Zahlen nicht ziffernweise, sondern als Ganzes erfasst werden.

Der Ausdruck «programmgespeichert» weist daraufhin, dass die Befehlsfolge für die Lösung einer Aufgabe durch die Maschine, eben das «Programm», in der Maschine selber gespeichert ist, in einem Speicher, der sowohl Ausgangsgrössen, Resultate als auch die Befehle enthält. Die einzelnen Speicherstellen sind fortlaufend numeriert.

Primär sind die programmgespeicherten digitalen Rechenautoautomaten in der Lage, gewisse einfache Grundfunktionen durchzuführen, etwa eine Angabe aus einem Ort des Speichers an einen zweiten Ort zu bringen oder zwei Zahlen miteinander zu multiplizieren usw. Ein Befehl muss die Nummer der Speicherstelle tragen, an welcher die Maschine die zu verarbeitenden Grössen findet. Soweit deckt sich die Arbeitsweise der Maschine mit der Vorstellung einer starren Abwicklung eines festgelegten Arbeitsganges vollständig. Eine wesentliche Erweiterung bringen nun die bedingten Sprungbefehle. Diese erlauben es, bei Vorliegen bestimmter Umstände die Befehlsfolge zu durchbrechen und einen andern Programmteil durchzuführen. Ein derartiger bedingter Sprung könnte z.B. erfolgen, wenn eine bestimmte Grösse negativ ist. Dadurch wird es möglich, in Abhängigkeit von Ausgangsgrössen und Resultaten die Verarbeitung anzupassen und zwar auf praktisch beliebig komplexe Weise, da derartige Sprungbefehle beliebig häufig nacheinander durchgeführt werden können. Insbesondere ist es oft zweckmässig, eine Reihe von Instruktionen mehrfach durchlaufen zu lassen; das gilt vor allem für die Lösung vieler mathematischer Aufgaben.

Obwohl die bedingten Sprungbefehle eine reichere Strukturierung der Arbeitsvorgänge erlauben, ergibt sich wieder eine Arbeitsweise, die auf einer höheren Stufe vollständig determiniert ist und der Vorstellung eines starren Ablaufes entspricht.

Es zeigt sich aber, dass dieses Modell viel leistungsfähiger ist als es auf den ersten Blick aussieht und zwar deshalb, weil die Speicher für die Befehle und für die Ausgangsgrössen, Zwischenergebnisse und Resultate prinzipiell nicht voneinander verschieden sind. Diese Tatsache wirkt verwirrend. Es stellt sich die Frage, wie denn die Maschine Befehle von Daten unterscheiden kann. Die Antwort lautet, dass der Inhalt einer Speicherstelle es den meisten derartigen Maschinen nicht erlaubt, festzustellen, ob es sich um Angaben oder Befehle handelt. Normalerweise folgen sich die Befehle sequentiell im Speicher. Ausnahmen treten nur bei Sprungbefehlen auf.

Diese Eigenschaft, dass Daten und Befehle prinzipiell nicht unterscheidbar sind, macht es möglich, in der Maschine einen Befehl zuerst zu bilden und an eine Stelle zu bringen, wo er dann später durchgeführt wird. Diese am Anfang etwas schwer verständliche Eigenschaft erlaubt es der Maschine, in einem gewissen Sinne sich selber Befehle aufzubauen. Diese Ausdrucksweise kann aber zu einem Missverständnis Anlass bieten. Die Befehle, welche einen durchzuführenden Befehl erst aufbauen, sind vom Ersteller der Instruktionsreihe gebildet worden, Die in Verfolgung dieser Befehle durch die Maschine konstruierten Befehle sind also indirekt doch wieder vom Menschen vorausbestimmt.

In dieser Möglichkeit, Maschinenbefehle durch die Maschine in einem bestimmten Sinne selber erzeugen zu lassen, liegt die entscheidende Wurzel für die grosse Flexibilität der programmgespeicherten Rechenmaschinen. Hier liegt auch der eigentliche Grund dafür, dass heute die Grenzen der Leistungsfähigkeit der Maschinen nicht anzugeben sind. Der Mensch steht hier wohl zum ersten Mal vor der Tatsache, dass er eine Maschine konstruiert hat, von der er zwar weiss, wie sie aufgebaut ist, und dass sie bestimmte Aufgaben lösen kann. Er weiss aber nicht, wie weit die Leistungsfähigkeit der Maschine über das Erwartete und Geplante hinausgeht. Das Nichtüberblickbare liegt hier nicht etwa in der Geschwindigkeit, die auch in diesem Falle keine Hexerei darstellt, sondern in der Idee der Programmspeicherung, die es erlaubt, die Maschine gewissermassen auf sich selber anzuwenden, eine Art Rekursion höherer Ordnung durchzuführen.

Die Erzeugung des Programms durch die Maschine selber soll an einem einfachen Beispiele noch etwas klarer gemacht werden. Die Befehle für die Maschine sind in einer verschlüsselten Form aufzustellen. Die Schlüssel sind je nach Maschinentyp schwerer oder leichter zu merken. Auf alle Fälle wäre es viel bequemer, wenn der Auftrag für die Maschine in der Umgangssprache formuliert werden könnte, oder wenigstens in einer Kunstsprache, die dieser nahe steht. Dieses Problem ist heute grundsätzlich gelöst. Zum Beispiel kann als Programm geschrieben werden: Gewicht durch Volumen ergibt spezifisches Gewicht. Ein einmal erstelltes besonderes Programm erlaubt es dann, aus der für uns verständlichen Fassung das eigentliche Maschinenprogramm herzustellen und dann auszuführen. Der Maschine muss natürlich irgendwie mitgeteilt werden, was das Gewicht und das Volumen für sie bedeuten. Sie muss wissen, an welchen Speicherstellen sie das Gewicht und das Volumen

findet. Diese Mitteilung braucht sie aber nur ein für allemal. Sie «erinnert sich» dann gewissermassen immer, wo sie diese Grössen finden kann. Für das Ergebnis, das spezifische Gewicht hingegen, ist es nicht notwendig, der Maschine eine Mitteilung zu machen. Das Übersetzungsprogramm ist so aufgebaut, dass es für alle Namen, z. B. also Gewicht, spezifisches Gewicht, eine Liste führt mit der Nummer der entsprechenden Speicherstelle. Wenn im Programm, das in der Kunstsprache geschrieben ist, ein Name auftritt, sucht das Übersetzungsprogramm in der Liste, wo die entsprechende Information gespeichert ist. Findet das Übersetzungsprogramm in der Liste den Namen nicht, z. B. das spezifische Gewicht als Resultat, so ordnet es dem Namen den nächsten freien Speicherplatz zu und fügt Namen und Speichernummer der Liste bei. Das erlaubt später auch die errechneten Ergebnisse mit ihrem Namen anzurufen anstatt mit der Speichernummer. Der Mensch, der das Programm erstellt, weiss nicht, wo das spezifische Gewicht gespeichert ist, er hat aber der Maschine ein Zuordnungsschema mitgeteilt, auf Grund dessen sie deterministisch diese Aufgabe löst. – Nachdem die Maschine in einem längeren Prozess die Umwandlung des Befehls aus der Kunstsprache in die Verschlüsselung der Maschine durchgeführt hat, kann sie nun die eigentliche Rechnung, Gewicht durch Volumen, erledigen.

Die Umwandlung aus einer uns leichter verständlichen Sprache in den Code der Maschine kann als Übersetzung aus einer Sprache in eine andere aufgefasst werden. Im Prinzip kann die Maschine also übersetzen. Die Übersetzung bezieht sich aber hier auf Kunstsprachen, die viel weniger reich an Ausdrucksmöglichkeiten sind als natürliche Sprachen. Auf dem Gebiete der Übersetzung natürlicher Sprachen wird heute sehr viel gearbeitet. Es dürften zur Zeit einige 100 Forscher ihre Arbeitszeit vollständig dieser Frage widmen. Es zeigt sich, dass für natürliche Sprachen das Problem viel schwieriger ist. Viele Hindernisse werden sich wegräumen lassen. Auf diesem Gebiet wird jedoch der Mensch noch lange überlegen bleiben, da in vielen Fällen beim Übersetzen das Verständnis des Textes eine Voraussetzung bildet.

Trotz dieser grundlegenden Schwierigkeit ist jedoch zu erwarten, dass in absehbarer Zeit die Übersetzung technischer Schriften durch Maschinen häufig angewendet wird – allerdings wird es sich um Übersetzungen mittlerer Qualität handeln. Der damit erreichte Stand scheint hoch. Wahrscheinlich wird es aber schwer fallen, ihn in kurzer Zeit

wesentlich zu verbessern. Auf alle Fälle wird der Mensch hier höchstens einmal von der Maschine eingeholt, während er auf andern Gebieten von ihr übertroffen werden kann.

Unser Beispiel der Übersetzung von Kunstsprachen zeigt uns aber auch etwas anderes, nämlich, dass die programmgespeicherten elektronischen Rechenmaschinen nicht nur rechnen können, sondern ganz allgemein in der Lage sind, Zeichen nach Regeln zu verarbeiten, Symbole zu manipulieren. Der Ausdruck «Rechenmaschinen» erweist sich je länger je mehr als zu eng. Es zeigt sich, dass der zwar immer noch bedeutende Anteil an numerischen Arbeiten auf diesen Geräten abnimmt, hingegen die nicht-numerischen Anwendungen an Bedeutung gewinnen. Es ist z.B. falsch, anzunehmen, dass nur das Gebiet der numerischen Mathematik von diesen Maschinen erfasst wird. Sogar für Forschungsaufgaben auf dem Gebiet der reinen Mathematik werden sie zum mindesten Hilfsfunktionen übernehmen können. Es drängt sich deshalb auch eine andere Bezeichnung als Rechenmaschinen für die von uns hier betrachteten Geräte auf. Besser wäre z.B. der Ausdruck «Datenverarbeitungsanlage», der jedoch sprachlich nicht ganz befriedigt. Es gibt wohl keine gute und knappe Übersetzung des Ausdrucks «data processing». Wir wollen deshalb weiterhin den zu engen Ausdruck «Rechenmaschine» beibehalten.

In einem gewissen Sinne ist die Erkenntnis von der über das rein Rechnerische hinausgehenden Kraft der elektronischen Geräte für denjenigen, der ihren logischen Aufbau kennt, nicht fernliegend. Die Maschinen arbeiten ja so, dass elektronische Zeichen Grundverknüpfungen der Logik, z.B. «und», «oder», «nicht», unterworfen werden. Aus diesen Grundverknüpfungen können unter anderem Rechenoperationen konstruiert werden. Primäre Funktionen bilden aber die logischen Grundbeziehungen. Es ist deshalb auch in einem gewissen Grade naheliegend, dass die Untersuchungen der Logistiker für die Forschung auf diesem Gebiete von besonderer Bedeutung geworden sind. Für den Logistiker ist sogar die Frage nach den Grenzen der Fähigkeiten programmgespeicherter Rechengeräte im Prinzp vollständig geklärt. Alle diese Geräte sind grundsätzlich äquivalent der von Turing schon 1936 betrachteten Modellmaschine, die in der Lage ist, alles zu berechnen, was überhaupt sich berechnen lässt. Dass Grenzen bestehen, welche es nicht erlauben, alle Probleme in einem Kalkül zu lösen, haben die Untersuchungen von Gödel, Church und anderen gezeigt. Die Grenzen der

Entscheidbarkeit von Problemen im Rahmen einer Logik gelten ganz natürlich auch für die elektronischen Rechenmaschinen.

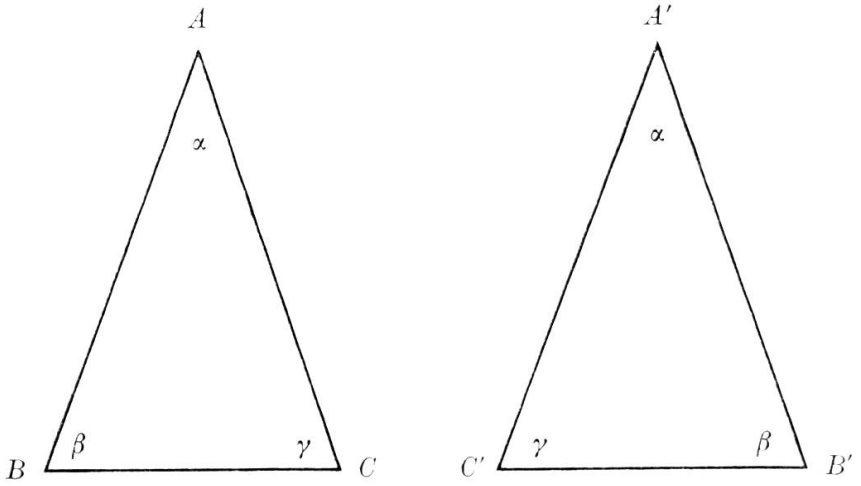
So wichtig und wertvoll diese Ergebnisse in vieler Beziehung auch sind, geben sie uns doch keine Antwort auf die uns hier interessierende Frage, ob Maschinen in der Lage sind, Aufgaben zu lösen, die *Intelligenz* voraussetzen. Wir wollen deshalb nun einige Beispiele näher betrachten.

Das wohl bekannteste Beispiel von Simulation von Intelligenz stellen die künstlichen Tiere dar, die sich ihren Weg zum Ziel in einem Labyrinth suchen. Wenn sie ihn durch systematisches Suchen gefunden haben, durchlaufen sie das nächste Mal den Weg direkt. Sie erinnern sich gewissermassen. Sie haben scheinbar etwas gelernt. Das erste Beispiel für ein derartiges Modell stammt von Shannon aus dem Jahre 1950. Eine Verbesserung besteht darin, dass die künstlichen Tiere auf nachträgliche Änderungen des Labyrinths «zweckmässig» reagieren, d.h. sie suchen von dem Augenblick, an dem sie eine Änderung feststellen, wieder einen Weg zum Ziel, wobei sie notfalls zum Ausgang zurückkehren. So verblüffend das Verhalten dieser künstlichen Tiere auch aussieht, die praktische Realisation stellt keine allzu grossen Anforderungen. Es ist nur notwendig, einen systematischen Suchprozess aufzubauen, ausserdem den Weg zu speichern, der beschritten wird, dafür zu sorgen, dass nicht mehrfach der gleiche Weg durchlaufen wird und nach Erreichen des Ziels vom Suchprogramm auf das direkte Durchlaufprogramm umzuschalten. Für den Fall, dass das Labyrinth nachträglich verändert wird, muss im Durchlaufprogramm die Möglichkeit der Rückkehr in das Suchprogramm gegeben sein. Der ganze Vorgang setzt natürlich eine geeignete Verbindung zwischen einem Rechengerät und der physikalischen Realisation des künstlichen Tieres voraus. Das Beispiel zeigt deutlich, dass es möglich ist, eine Maschine so zu programmieren, dass sie ein Vorgehen, das sich nicht bewährt, nicht mehr durchführt und umgekehrt ein Vorgehen beibehält, das Erfolg hat. Soweit Lernen also nur aus dem Anpassen an gemachte Erfahrungen besteht, kann es einer Maschine beigebracht werden, doch soll darauf hier nicht näher eingegangen werden.

Das nächste Beispiel stammt aus der Geometrie. Es gibt heute schon mehrere Programme, die in der Lage sind, Aufgaben der elementaren Geometrie zu lösen und Lehrsätze herzuleiten, die einem durchschnittlichen Maturanden zum mindesten gewisse Schwierigkeiten machen würden.

Eine geometrische Figur wird in einem elektronischen Rechenautomaten durch eine abstrakte Verschlüsselung festgehalten, auf die hier nicht näher eingegangen werden soll. Weiter sind die Verarbeitungsregeln und die Axiome der Geometrie ebenfalls im Programm zu berücksichtigen. Ausserdem erweist es sich häufig als zweckmässig, Hilfslinien oder ganze Hilfsfiguren zu betrachten. Auch diese Möglichkeit ist in derartigen Programmen eingebaut.

Die Basiswinkel in einem gleichschenkligen Dreieck sind bekanntlich einander gleich. Minsky hat von seinem Programm einen Beweis für diesen Satz erhalten, der sehr einfach ist. Normalerweise wird der Beweis geführt durch Ziehen der Höhe von der Spitze und Nachweis, dass die beiden Teil-Dreiecke kongruent sind. Die elektronische Maschine löst die Aufgabe so, dass sie ausser dem Dreieck ABC das Dreieck $A'B'C'$ betrachtet, das aus dem ersten durch eine Drehung aus der Ebene entsteht. Die beiden Dreiecke stimmen in drei Seiten überein, sind also kongruent. Daraus folgt, dass die Basiswinkel gleich sind.




In diesem Falle hat die Maschine einen Beweis gefunden, der dem Hersteller des Programms nicht bekannt war. Sie hat ihn also in einer Beziehung übertroffen.

Als weiteres Beispiel für künstliche Intelligenz wollen wir nun ein Programm betrachten, das Newell, Shaw und Simon aufgestellt haben. Dieses Programm ist in der Lage, Aufgaben einer gewissen Struktur auf verschiedenen Gebieten zu lösen. Da die Erklärung des Programms sich im Abstrakten bewegen muss, soll vorerst anhand eines Beispiels gezeigt werden, wie eine bestimmte Aufgabe praktisch gelöst wird. Die Darstellung beschränkt sich dabei auf das Wesentliche.

Wir stellen der Maschine die Aufgabe zu beweisen, dass

$$(\operatorname{tg} x + \operatorname{ctg} x) \sin x \cos x = 1$$

ist. Das Ziel besteht für die Maschine darin, eine Reihe von Transformationen der linken Seite der Gleichung zu finden, die sie in die rechte Seite überführt. Erlaubte Transformationen sind die Regeln der Schulalgebra und die elementaren Beziehungen zwischen den trigonometrischen Funktionen. Das *Hauptziel* der Maschine besteht also in der Überführung der linken Seite in die rechte. Sie versucht nun zuerst, ob es durch die einmalige Anwendung der erlaubten Umwandlungsregeln möglich ist, das Ziel zu erreichen. Das ist nicht der Fall. Dann setzt sich die Maschine eine Unteraufgabe, nämlich die Reduktion des Unterschieds zwischen den beiden Seiten. Als Unterschied ist z. B. zu verzeichnen, dass auf der linken Seite eine Tangensfunktion vorkommt, auf der rechten Seite nicht. Die algebraischen Mittel helfen nicht weiter. Ist es möglich, den Tangens auf der linken Seite wegzuschaffen mit Hilfe der zur Verfügung stehenden Beziehungen zwischen den trigonometrischen Funktionen? Die Maschine stellt fest, dass das möglich ist, da sich der Tangens als reziproker Wert des Cotangens ausdrücken lässt. Diese Ersetzung wird durchgeführt. Ein Teilziel ist erreicht. Der «Abstand» der linken Seite von der rechten Seite ist verkleinert. Die reduzierte Aufgabe lautet nun, die transformierte linke Seite in die rechte Seite überzuführen. Das gelingt abermals nicht in einem Schritt. Die Maschine stellt sich wiederum das Teilziel, den Abstand der beiden Seiten zu verkleinern. Auf der linken Seite kommt der Kotangens vor, auf der rechten Seite nicht. Algebraische Methoden ergeben keinen Fortschritt. Die Anwendung der Beziehung $\operatorname{cotg} x = \dfrac{\cos x}{\sin x}$ führt dann zum Erreichen des Teilziels. Es ergibt sich als neue linke Seite

$$\left( \frac{1}{\dfrac{\cos x}{\sin x}} + \frac{\cos x}{\sin x} \right) \sin x \cos x .$$

Nunmehr versucht die Maschine die Entfernung zwischen den beiden Seiten weiter zu verkleinern und stellt fest, dass sie dafür die Beziehung $\sin^2 x + \cos^2 x = 1$ benützen sollte. Diese Relation lässt sich dafür im Augenblick nicht anwenden. In mehreren Einzelschritten, die wir nicht mehr genau ausführen wollen, wird der Unterschied zwischen der linken

Seite und der anzuwendenden Beziehung durch algebraische Schritte verkleinert, bis die Transformation gelingt und die Maschine stolz «quod erat demonstrandum» schreibt.

Nach diesem Beispiel für eine Anwendung des allgemeinen Programms für die Lösung von Aufgaben, das von Newell, Shaw und Simon entwickelt worden ist, wollen wir es nun kurz in seiner vollen Allgemeinheit charakterisieren. Es handelt sich nicht um ein Programm, das etwa nur zur Lösung derartiger trigonometrischer Aufgaben aufgestellt wurde; die Lösungsmethode ist allgemein und wurde z. B. auch für Probleme der Logistik eingesetzt. Das Programm stellt ein Teilgebiet der Forschungsarbeit der Verfasser dar, welche nach ihren eigenen Worten das Verständnis intellektueller, anpassungsmässiger und schöpferischer Tätigkeit des Menschen anstrebt. Als Grundlage für das Programm dienten Beobachtungen an amerikanischen Studenten beim Lösen von Aufgaben auf dem Gebiete der symbolischen Logik. Die Studenten mussten laufend aussprechen, was sie beim Lösen der ihnen gestellten Probleme dachten. Das auf Grund dieser Beobachtungen erstellte Programm ist in der Lage, Probleme zu bewältigen, welche sich allgemein auf Objekte und Operatoren beziehen. In unserem trigonometrischen Beispiel sind Objekte die Zahlen und trigonometrischen Funktionen, Operatoren, die mit den Objekten erlaubten algebraischen und trigonometrischen Umformungen. Wenn wir als Beispiel das Schachspiel betrachten, so sind die Schachstellungen die Objekte und die erlaubten Züge die Operatoren, die neue Objekte ergeben. Für ein formales mathematisches System hingegen sind Objekte die Axiome und Theoreme, Operatoren die erlaubten Regeln für das Schliessen.

Das Programm bezieht sich auf allgemeine Objekte und allgemeine Operatoren. In der Anwendung auf ein bestimmtes Gebiet ist es deshalb notwendig, Angaben über die Eigenschaften der Objekte und Operatoren zusätzlich noch anzugeben. Das Trennen der Methoden für die Lösung von den Eigenschaften eines bestimmten Anwendungsgebietes zeigt, dass der Name «general problem-solving program» zu Recht gewählt wurde. Immerhin muss betont werden, dass nicht alle Aufgaben, die zu lösen sind, sich in dieser einfachen Form stellen. Das Programm ist allgemein in dem Sinne einer Anwendbarkeit auf viele Gebiete, nicht aber auf alle.

Wenn der Mensch ein Problem löst, so verwendet er dabei häufig heuristische Methoden, d. h. er benützt Verfahren, die nicht ganz exakt

sind und nicht sicher zum Ziele führen, Analogieschlüsse, Veränderung der Aufgabe usw. Auch die Maschine braucht etwas Ähnliches. Wir wollen aber dafür entgegen dem Vorgehen von Newell, Shaw und Simon nicht ebenfalls das Wort Heuristik verwenden, sondern wie mehrere Kritiker vorschlagen, den Ausdruck «Strategie». Es dürfte zweifellos besser sein, alles zu vermeiden, was leicht zu Missdeutungen führen kann. Der Ausdruck Strategie ist sachlich, während das Wort Heuristik vielleicht zu stark dazu verführt, Übereinstimmung mit dem Denken des Menschen beim Lösen von Aufgaben anzunehmen.

Was für Strategien sind jetzt im «general problem-solving program» eingebaut?

Einmal das Prinzip des Anstrebens von Teilzielen, d.h. es wird versucht, ein Ziel, das in einem Schritt nicht erreicht werden kann, durch ein leichteres Teilziel zu ersetzen. Dazu ist es notwendig, dass das Programm in der Lage ist, einen Fortschritt auf ein Ziel hin zu erkennen, d.h. es müssen auf irgendeine Weise Definitionen von Unterschieden gegeben sein. In unserem Beispiel der trigonometrischen Funktionen besteht z.B. ein Unterschied durch das Vorkommen trigonometrischer Funktionen auf der linken Seite; die auf der rechten Seite fehlen. Die Unterschiede können jedoch auf ganz verschiedenen Gebieten bestehen. Es kann ja nicht nur inbezug Vorhandensein von Elementen, sondern auch in ihrer Verknüpfung ein Unterschied bestehen, wie das Beispiel

$$(a-b)\,(a+b) = a^2-b^2$$

zeigt. Hier kommen auf beiden Seiten die gleichen Grössen vor, aber ihre Verknüpfung ist verschieden.

Beim Beweisen eines Satzes kann es bekanntlich unter Umständen zweckmässig sein, das Problem scheinbar zu komplizieren, indem etwa Hilfsvariable oder Zusatzglieder eingeführt werden. Das Programm von Newell, Shaw und Simon weist diese Möglichkeit nicht auf. Es handelt sich um ein geradliniges Vorgehen.

Die zweite Strategie, die in das Programm eingebaut ist, besteht im Wechselspiel zwischen Hilfsmitteln und Zielen. Die Art dieses Vorgehens kann vielleicht an besten mit dem sehr einfachen Beispiel erklärt werden, das Newell, Shaw und Simon in ihrer Arbeit verwendet haben:

Ich möchte meinen Sohn in den Kindergarten bringen. Was ist der Unterschied zwischen dem, was ich habe und dem, was ich will? Räumliche Entfernung. Was überwindet räumliche Entfernung? Mein

Auto. Mein Auto kann nicht fahren. Was ist notwendig, damit es wieder fährt? Eine neue Batterie. Wer hat neue Batterien? Eine Garage usw.

Man sieht hier schön den Wechsel zwischen Zielen und den zum Erreichen dieser Ziele notwendigen und zweckmässigen Hilfsmitteln. Diese primitive Art von – sagen wir nun nicht «Denken», sondern Verarbeitung – ist in das «general problem-solving program» ebenfalls eingebaut. Abstrakt ausgedrückt muss das Programm in der Lage sein, Differenzen zwischen den Objekten zu erkennen und die geeigneten Operatoren zu finden, welche diese Differenzen überwinden. Dabei wird es notwendig sein, eine Rangordnung der verschiedenen Arten von Differenzen zu erstellen, damit wichtige vorweg entfernt werden.

Soweit sind die im Programm berücksichtigten Strategien so aufgebaut, dass sie nur jeweils einen Schritt in Richtung auf das Ziel hin ausführen, ohne einen Gesamtplan der Lösung anzustreben. Eine weitere Strategie sucht nun einen derartigen Plan für eine Lösung herzustellen, indem gewisse Einzelheiten der Aufgabe weggelassen werden. Das so vereinfachte Problem wird gelöst und dann der Lösungsweg auf das ursprüngliche Problem übertragen. Das Verfahren führt nicht unbedingt zum Ziel, wie auch die bisher beschriebenen Strategien nicht, bringt aber in vielen Fällen eine wesentliche Verkürzung des Arbeitsganges, gelegentlich aber auch eine unnötige Verzögerung infolge der Untersuchung von Lösungen des vereinfachten Problems, die sich nicht auf das ursprüngliche Problem übertragen lassen. Diese Strategie ist bei unserem trigonometrischen Beispiel nicht angewendet worden.

In das «general problem-solving program» sind noch andere Strategien eingebaut, auf die hier nicht mehr näher eingegangen werden soll. Dem Aufstellen eines derartigen anspruchsvollen Programms stellen sich noch viele Schwierigkeiten entgegen. Wenn hier nur auf einige Grundzüge eingegangen werden konnte, dürfen deshalb die Schwierigkeiten für das Programmieren einer derartigen künstlichen Intelligenz nicht unterschätzt werden.

Wir wollen uns nun als letztes einem Ausblick auf Anwendungen für Forschungsaufgaben der reinen Mathematik zuwenden. Die bereits behandelten Beispiele haben wohl schon zur Genüge gezeigt, wie falsch es ist anzunehmen, dass elektronische Rechengeräte nur für numerische Aufgaben eingesetzt werden können. Man kann z.B. heute schon mit ihnen analytisch integrieren und differenzieren.

Schwierige mathematische Theoreme sind zurzeit durch Maschinen nicht zu beweisen. Es hat sich gezeigt, dass die Beherrschung von Kalkülen der Logik eine entscheidende Voraussetzung für Erfolge auf mathematischem Gebiet bilden. Dem amerikanischen Forscher Wang ist es im Jahre 1960 gelungen, 350 Sätze aus dem bekannten Buch von Whitehead und Russel «Principia mathematica» maschinell zu beweisen, wobei die Maschine für die ganze Arbeit gut 8 Minuten brauchte. Sie entdeckte dabei, dass ein Theorem nicht der Klassenlogik, sondern der Aussagenlogik zugehört, was bisher den Autoren und allen Lesern entgangen ist. Hingegen hat sie glücklicherweise kein Theorem als falsch bezeichnet.

Wang glaubt, dass zunächst einfachere Gebiete, wie die Schulalgebra und Schulgeometrie, erfasst werden können. Schwierigere Bereiche lassen sich dann stufenweise auch in Angriff nehmen. Es ist dabei zu berücksichtigen, dass für eine Maschine die Schwierigkeiten nicht gleich liegen wie für Menschen. Der Mensch ist im Vorteil auf Gebieten, wo Intuition und Anschauung ihm helfen. Für nichteuklidische Geometrie z.B. dürfte die Maschine einen Vorsprung besitzen.

Die Idee, die gesamte Mathematik auf das schematische Schliessen innerhalb eines Logikkalküls zurückzuführen, geht auf Leibniz zurück. Peano, Hilbert und andere haben diesen Gesanken weiterverfolgt. Hilbert wies darauf hin, dass die gesamte klassische Mathematik sich im Rahmen der elementaren Prädikatenlogik formalisieren lasse. Als zentrales Problem der mathematischen Logik ergibt sich somit das Problem, einen Algorithmus zu finden, der es erlaubt zu entscheiden, ob eine Formel der Prädikatenlogik gültig ist oder nicht. Church und Turing haben dann in den Dreissigerjahren dieses Jahrhunderts gezeigt, dass es keinen Algorithmus für das Entscheidungsproblem der Prädikatenlogik gibt. Daraus folgt, dass auch elektronische Rechenmaschinen nicht allgemein entscheiden können, ob eine mathematische Behauptung zutrifft oder nicht. Wenn es nun aber kein Entscheidungsverfahren gibt, so lassen sich wenigstens Beweisverfahren konstruieren, d.h. Kalküle, die für jede gültige Formel einen Beweis finden. Für ungültige Formeln suchen sie jedoch im allgemeinen unaufhörlich einen Beweis. Derartige Beweisverfahren lassen sich jedoch auf elektronischen Rechenmaschinen programmieren.

Es liegt uns nicht, Prophezeiungen anzustellen. Wir wollen deshalb nur erklären, dass mit dem heute erreichten Stand auf dem Gebiete

der mathematischen Logik eine Grundlage geschaffen ist, welche weitere Fortschritte in der Richtung der reinen Mathematik ermöglicht und sogar wahrscheinlich macht.

Zusammenfassend sei festgehalten, dass elektronische Rechenmaschinen – wenigstens in der in dieser Arbeit betrachteten Form – deterministisch gebaut sind. Die Vorstellung einer zwar flexiblen, aber doch in einem höheren Sinne starren, zum voraus festgelegten Arbeitsweise ist richtig. Mit einer derartigen Arbeitsweise lassen sich jedoch Aufgaben behandeln, die bisher dem selbständig denkenden Menschen vorbehalten schienen. Neue Gebiete erweisen sich einer rationalen Behandlung als zugänglich. Trotzdem bestehen natürlich Grenzen.

Wo z.B. Grenzen liegen, sei am Beispiel der Musik angedeutet. Soweit die Musik von Regeln beherrscht wird, soweit ist sie rational erfassbar. Auch gelegentliche systematische Abweichungen von Regeln bilden wieder ein Schema und lassen sich programmieren. Soweit die Komposition von Musikstücken also sich auf Regeln zurückführen lässt, lässt sie sich programmieren. Es ist wohl nicht nötig besonders zu betonen, dass nur damit Mozart und Bach keine Konkurrenz gemacht wird.

Es war das Ziel dieser Ausführungen, zu einem vertieften Durchdenken der sich hier stellenden Probleme anzuregen. Es trifft allerdings zu, wenn gesagt wird, eine Maschine kann nur das tun, was ihr von Menschen befohlen wird. Aber man kann z.B. einer Maschine befehlen, einen mathematischen Beweis zu suchen, etwas zu lernen, sich zweckmässig an Reaktionen der Aussenwelt anzupassen. Die Auswirkungen der uns durch diese neuen Maschinen gebotenen Möglichkeiten greifen nicht nur in unser materielles Leben ein, sondern müssen uns dazu veranlassen, unsere Vorstellungen über die Ausnahmestellung des denkenden Menschen einer genauen Prüfung zu unterziehen. Auch wenn wir das, was Maschinen leisten können, nur mit simulierter Intelligenz bezeichnen wollen, greift es doch eindeutig in Bereiche, die wir bis jetzt dem Menschen vorbehalten glaubten. Der Mensch hat es verstanden, mit geeigneten Werkzeugen und Maschinen seine eigenen körperlichen Kräfte und seine Sinnesorgane zu verstärken. Es wird ihm heute die Gelegenheit geboten, ähnliches auf intellektuellem Gebiete zu vollbringen. Die wirklich tiefen Probleme für den Menschen liegen jedoch nicht darin, ob er auf diesem oder jenem Gebiete von einer Maschine übertroffen wird, sondern wozu er die ihm auf bisher materiellem und

neu nun intellektuellem Gebiete gebotenen Möglichkeiten verwendet. Das Erkennen der richtigen Maßstäbe und Ziele kann dem Menschen keine Maschine abnehmen.

## Literaturhinweise

*Zemanek, H.:* Kybernetik. I. Teil: Maschinen mit Phantasie; II. Teil: Das Tier und die Maschine. Mathematik–Technik–Wirtschaft, 4, 1957.
*Verschiedene:* Machine translation of languages; 2. Auflage 1957.
*Davis, Martin:* Computability and Unsolvability, 1958.
*von Neumann, John:* The Computer and the Brain, 1958.
*Berkeley, Edmund C.:* Symbolic logic and intelligent machines, 1959.
*Newell, A., Shaw, J.C.* and *Simon, H.A.:* Report on a general problem-solving program; Information Processing, 1959.
*Kilburn, T., Grimsdale, R.L. and Sumner, F.H.:* Experiments in machine learning and thinking; Information Processing, 1959.
*Dunham, B., Fridshal, R. and Sward, G.L.:* A non-heuristic program for proving elementary logical theorems; Information Processing, 1959.
*Gelernter, H.:* Realization of a geometry theorem proving machine; Information Processing, 1959.
*Steinbuch, K.:* Pattern recognition and machine learning; Information Processing, 1959.
*Gilmore, P.:* A program for the production from axioms, of proofs for theorems derivable within the first order predicate calculus; Information Processing, 1959.
*Blachman, H.M.:* NPL symposium on the mechanization of thought processes; Communications of the Association for Computing Machinery, 2, 1959.
*Steinbuch, K.:* Lernende Automaten; Elektronische Rechenanlagen, 1, 1959.
*Samuel, A.L.:* Some studies in machine learning, using the game of checkers; IBM Journal of research and development, 3, 1959.
*Zemanek, H.:* Wesen und Grenzen des Automaten; Mathematik–Technik–Wirtschaft, 6, 1959.
*Prawitz, D., Prawitz, H. and Voghera, N.:* A mechanical proof procedure and its realization in an electronic computer; Journal of the Association for Computing Machinery, 7, 2, 1960.
*Davis, M. and Putnam, H.:* A computing procedure for quantification theory; Journal of the Association for Computing Machinery, 7, 3, 1960.
*McCarthy, J.:* Recursive functions of symbolic expressions and their computation by Machine, Part. I.; Communications of the Association for Computing Machinery, 3, 4, 1960.
*Wang, H.:* Proving theorems by pattern recognition I.; Communications of the Association for Computing Machinery, 3, 4, 1960.
*Gilmore, P.C.:* A proof method for quantification theory: its justification and realization; IBM Journal of research and development, 4, 1, 1960.
*Eier, R. and Zemanek, H.:* Automatische Orientierung im Labyrinth; Elektronische Rechenanlagen, 2, 1, 1960.
*Wang, H.:* Toward mechanical mathematics. IBM Journal of research and development, 4, 1, 1960.
*Verschiedene:* Advances in computers; vol. 1, 1960.

## Résumé

Après une brève présentation du fonctionnement fondamental des ensembles électroniques à programme enregistré, l'auteur, à l'aide de quelques exemples, montre jusqu'à quel point aujourd'hui les machines sont capables d'imiter l'intelligence.

## Summary

After a short presentation of the fundamental functional system of stored program electronic data processing machines, the author shows with help of a few examples how far today intelligence can be imitated by machines.

## Riassunto

Dopo una breve descrizione dei sistemi de lavoro fondamentali dei complessi elettronici a programmazione interna, l'autore mostra con alcuni esempi, fino a che punto oggi le macchine sono in grado di imitare l'intelligenza.