Künstliche Intelligenz

Computerprogramme spielen Schach, lernen aus Erfahrung, verstehen bruchstückhaft menschliche Sprache – kurz: dringen unaufhaltsam in die Sphäre des Geistes vor. Faszinierender noch als diese Intelligenzleistungen freilich ist das, was Programme, die erfolgreich intellektuelle Fähigkeiten des Menschen modellieren, über das Wesen des Denkens selbst enthüllen.

Von **David L. Waltz**

Der Gedanke, Computer könnten eines Tages die geistigen Fähigkeiten des Menschen erreichen oder gar übertreffen, ist so alt wie die Elektronenrechner selbst. In den Anfangstagen des Computerzeitalters entbehrte er freilich jeder empirischen Grundlage. Plausibles Schließen, Planen, Lernen, Fühlen, das Formulieren von Begriffen, der Gebrauch natürlicher Sprachen oder kreatives Denken war den damaligen Rechnern so fremd wie ihren mechanischen Vorfahren, den Puppen, aufziehbaren Spielzeugsoldaten und Musikboxen. Schlimmer noch, die Idee, daß Intelligenz selbst eines theoretischen Erklärungsrahmens bedürfe, kam kaum jemandem in den Sinn. Intelligenz galt als klar erkennbare Eigenschaft, und unter einer intelligenten Maschine stellte man sich ein Gerät vor, das praktisch über dieselben intellektuellen Fähigkeiten verfügen würde wie der Mensch. Daher hießen die frühen Rechner auch Elektronengehirne, und John von Neumann, einer der Architekten der noch heute gebräuchlichsten Art von Computern, stellte explizite Analogien zwischen dem Computer und dem menschlichen Gehirn her. Immer noch assoziieren viele Menschen Daten mit Wissen, den Ablauf eines Programms mit dem Fällen von Entscheidungen, die Folge der Operationen in einem Computer mit dem Gedankengang und die Akkumulation von Daten mit dem Lernprozeß.

In den vergangenen 20 bis 25 Jahren hat die neue Disziplin der „Künstlichen Intelligenz" von solchen naiven Analogien zwischen Computer und menschlichem Gehirn Abschied genommen und sich ernsthaft daran gemacht, den Begriff Intelligenz auf ein theoretisches Fundament zu stellen. Für die Praktiker dieser Disziplin ist der Computer ein Werkzeug, um neue Wege des Denkens über das Denken zu ergründen. Dabei haben Rechenprogramme eindrucksvoll demonstriert, daß Computer bei gewissen Tätigkeiten (auch solchen, von denen die meisten Menschen sagen würden, daß sie Intelligenz erfordern: das Schachspielen etwa) den meisten Menschen überlegen sind. Mit jüngst entwickelten Programmen gelang zudem der Nachweis, daß der Computer sogar umfangreiche Theorien über ein begrenztes Gebiet, die Arithmetik zum Beispiel, aus wenigen einfachen Axiomen entwickeln kann. Als Frucht der Versuche, Merkmale intelligenten Verhaltens mit jener Akribie zu beschreiben, die ein Computerprogramm verlangt, ist unser Verständnis der Intelligenz beträchtlich gewachsen. Zugleich wurde die Analogie zwischen den Leistungen des Computers und des menschlichen Gehirns entscheidend vertieft und erweitert.

Betrachten Sie etwa den Lernprozeß: Beim Menschen scheint Lernen mit einem Wachstum des Gehirns sowie Änderungen seiner Struktur verknüpft zu sein, während sich von der Hardware eines Computers nichts dergleichen behaupten läßt. Änderungen bleiben hier auf die Programmebene beschränkt: Programme akkumulieren und organisieren Daten oder modifizieren gegebenenfalls auch sich selbst. Wenn nun Forscher auf dem Gebiet der Künstlichen Intelligenz die Ansicht vertreten, komplexe Programme und insbesondere solche, die sich selbst während des Ablaufs ändern, könnten als probate Modelle des Lernprozesses dienen, so heißt das nicht, daß sich Lernen mittels solcher Programme in allen Einzelheiten nachvollziehen ließe. Denn natürlich kann ein Computer, in dem ein Lernprozeß abläuft, nicht das Feuern der Nervenzellen im Gehirn eines Lernenden simulieren. Dennoch läßt sich das Computer-Modell der Intelligenz dank seiner hohen Flexibilität auf jeder Genauigkeitsstufe formulieren, die man zur Beleuchtung wesentlicher Funktionen des Denkens für sinnvoll hält.

Ich will hier versuchen, einen Eindruck davon zu vermitteln, welche Arten von Ergebnissen sich mit Programmen in Künstlicher Intelligenz erzielen lassen. Die meisten der von mir diskutierten Beispiele sind wohlbekannte Lösungsan-

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (photo copyright © by www.schaakcomputers.nl/) (600 dpi)

sätze auf den Hauptproblemfeldern dieser Disziplin. Da es mir darum geht, die wesentlichen Beiträge aufzuzeigen, die sie jeweils geleistet haben, werde ich bevorzugt solche Programme diskutieren, deren Ablauf sich kurz und bündig beschreiben läßt. Was folgt, ist somit mehr als Einführung denn als Zusammenfassung des Gebietes gedacht.

Eines der Leitmotive in der Künstlichen Intelligenz war stets die Frage, wie man es am besten anstellt, den Bereich möglicher Handlungen im Hinblick auf wohldefinierte Ziele zu erforschen. Im allgemeinen eröffnet jeder unternommene Schritt neue Handlungsmöglichkeiten. Bei der Planung einer Folge von Handlungen gilt es daher, sich in ein stark verzweigtes Labyrinth möglicher Zustände vorzutasten, das man als Suchbaum bezeichnet (Bild 2). Gewöhnlich stellt man diesen Baum auf dem Kopf stehend dar, so daß er mit der Wurzel nach oben und den Ästen nach unten zeigt. Die Wurzel repräsentiert die Ausgangssituation, während die Äste für die Handlungen und die Astenden für die möglichen Ergebnisse der Handlungen stehen. Eines oder mehrere der Ergebnisse können dem Ziel der Suche entsprechen.

Heuristische Suche

Den Gesamtbereich möglicher Handlungen in allen möglichen Richtungen zu erforschen hieße, jeden Ast des Baums einzeln zu verfolgen. Dieses Vorgehen böte die Gewähr dafür, daß die optimale Strategie tatsächlich auch gefunden würde. In vielen Fällen ist der Suchbaum jedoch so ausgedehnt, daß sich eine erschöpfende Suche von selbst verbietet. Daher wenden die meisten Programme für solche Probleme heuristische Prinzipien, das heißt simple Faustregeln, an, mit deren Hilfe sie schon frühzeitig die aussichtsreichsten Äste auswählen, um nur diese weiterzuverfolgen. Solche heuristischen Prinzipien sind in den meisten Fällen sehr praktisch und wirkungsvoll und verkürzen die Suche ganz erheblich, nur kann eben niemand am Ende sicher sein, daß das Ergebnis wirklich das optimale ist. Dennoch verwendet man heuristische Suchverfahren üblicherweise für Programme, die beispielsweise einen Roboter in die Lage versetzen, planvoll mit Objekten im Raum zu hantieren oder sich selbst überlegt fortzubewegen.

Programme, die Spiele wie Schach, Dame oder Backgammon beherrschen, sind anschauliche Beispiele für den Einsatz heuristischer Suchverfahren. Viele haben mittlerweile Profi- oder Meisterstärke erreicht. Ein Programm namens „Mighty Bee“ („Mächtige Biene“), geschrieben von Hans Berliner von der Carnegie-Mellon-Universität, schlug 1979 sogar den Backgammon-Weltmeister (siehe „Ein Computer spielt Backgammon“ in Spektrum der Wissenschaft, August 1980). Das Hauptinteresse aber dürfte sich auf Programme konzentrieren, die Schach spielen. Praktisch alle seit den fünfziger Jahren ausgetüftelten Schachprogramme basieren auf einem von Claude E. Shannon bei den Bell-Laboratorien entwickelten heuristischen Suchmodell.

Wenn sich ein Schachprogramm einen Zug überlegt, muß es die Stellungen, die sich aus jedem der möglichen Züge ergeben, irgendwie bewerten. Das kann mit Hilfe eines Punktschemas geschehen, bei dem der Wert der einzelnen Figuren durch Zahlen ausgedrückt wird und die Bedrohung gegnerischer Figuren Pluspunkte bringt, während bedrohte eigene Steine zu einem Punktabzug führen (Bild 3). Bei vielen Programmen gehen weitere Kriterien wie die Stärke der Bauernstellung, das Ausmaß der Kontrolle über das Zentrum und die Zahl der entwickel-

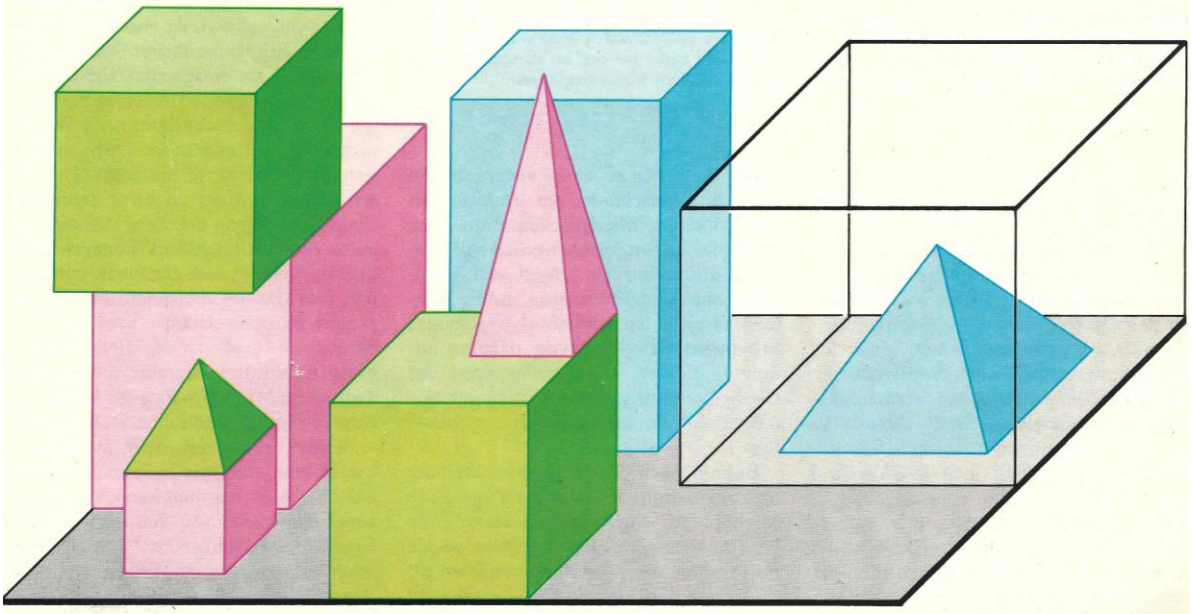

Bild 1: Die „Block-“ oder „Klötzchenwelt“ ist ein eng umgrenzter Mikrokosmos, der Forschern auf dem Gebiet der Künstlichen Intelligenz zum Studium der Möglichkeiten dient, Wahrnehmung, Denken und Handeln im Computer zu simulieren und zu koordinieren. Die Klötzchenwelt besteht aus einem Tisch, auf dem sich farbige Würfel, Kästen und Pyramiden befinden. Auch wenn sie nicht real ist, sondern nur im „Kopf“ eines Computers existiert, gelten in ihr die bekannten Gesetze der Alltagsphysik; eine Pyramide kann also beispielsweise nicht auf der Spitze stehen. Erfunden wurde die Klötzchenwelt von Terry A. Winograd am Massachusetts Institute of Technology als Aktionsraum für ein Computerprogramm namens SHRDLU. SHRDLU simuliert in seinem Reich Tätigkeiten, wie sie ein Mensch in unserer Welt auch verrichtet. So verstellt es auf eine in natürlicher Sprache gegebene Anweisung hin die Klötzchen nach einem Plan, den es sich selbst zurechtlegt. Ferner gibt es Auskunft über die Anordnung der Klötzchen zum gegenwärtigen oder zu einem früheren Zeitpunkt sowie über die bereits vorgenommenen oder noch beabsichtigten Handlungen und die Gründe dafür (siehe Bild 6). Andere Wissenschaftler haben gezeigt, wie sich auch das Erlernen neuer Tätigkeiten in der Klötzchenwelt nachbilden läßt. Ebenso wie dieses Bild sind auch die Bilder 4 und 6 an Abbildungen aus Winograds Artikel „Understanding Natural Language“ angelehnt, der in der Zeitschrift „Cognitive Psychology“ erschienen ist.

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (600 dpi)

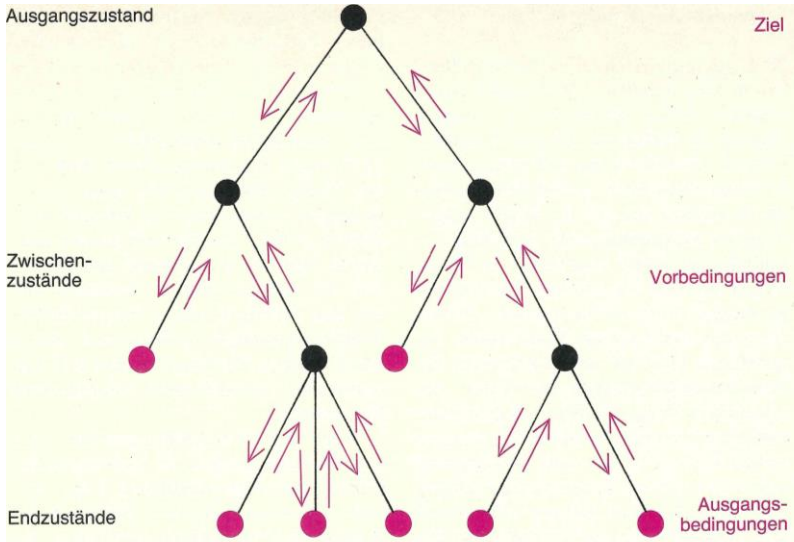

Bild 2: Viele Probleme in der Künstlichen Intelligenz laufen darauf hinaus, einen Suchbaum zu durchforsten. Die Suche beginnt am obersten Knoten oder der Wurzel des auf dem Kopf stehenden Baumes und verläuft zunächst entlang des äußersten linken Astes bis zum Endknoten (der Astspitze). Dann springt das Programm zurück zum ersten Verzweigungspunkt und nimmt sich den nächsten Ast vor. So werden von links nach rechts sukzessive sämtliche Zweige bis zu ihren Spitzen verfolgt. Man bezeichnet diese Vorgehensweise als Reihenprozeß. Je nach Art des Problems kann die Wurzel einen vorgegebenen Zustand verkörpern, oder aber ein gewünschtes Ziel. Im ersteren Fall repräsentieren die Verzweigungspunkte direkt unter der Wurzel die Ergebnisse der im Ausgangszustand möglichen Handlungen, also beispielsweise die Stellungen, die sich aus den in einer bestimmten Schachposition erlaubten Zügen ergeben. Die nächsttieferen Knoten stehen für die Ergebnisse darauffolgender Handlungen, also etwa die nach den möglichen Erwiderungen des Gegners erreichten Stellungen. Die Tiefe der Suche richtet sich dabei danach, wie weit das Programm vorausdenken soll. Verkörpert die Wurzel ein angestrebtes Ziel, so repräsentieren die benachbarten Knoten die Vorbedingungen von Handlungen, durch die sich das Ziel erreichen läßt. Auch für die Vorbedingungen existieren Vorbedingungen, symbolisiert durch die nächsttieferen Verzweigungspunkte. Die Suche endet bei den im Ausgangszustand bereits erfüllten Vorbedingungen.

ten Figuren in die Bewertung ein. Das Programm spielt so, daß es einen möglichst hohen Punktestand erreicht.

Die in einem Schachprogramm eingebauten heuristischen Prinzipien kontrollieren sowohl die Breite als auch die Tiefe des vom Programm analysierten Ausschnitts aus dem Suchbaum. Zunächst einmal berechnet das Programm die Punktzahl für jeden in der momentanen Stellung zulässigen Zug. Daraufhin könnte es sämtliche möglichen Erwiderungen des Gegners und anschließend die Gesamtzahl aller eigenen Entgegnungen ausfindig machen und bewerten. Schon bald aber muß es eine Auswahl unter den möglichen Zügen treffen, da die Anzahl der zu bewertenden Stellungen sonst ins Unermeßliche stiege. Diese Auswahl ist es, bei der die heuristischen Prinzipien zum Tragen kommen. Das Programm könnte sich beispielsweise darauf beschränken, den Zug mit der höchsten Punktzahl weiterzuverfolgen oder sich jene, sagen wir, fünf Zugmöglichkeiten näher anzusehen, die am ehesten Erfolg versprechen. Für die ausgewählten Züge analysiert es den Suchbaum dann bis zu einer vorgegebenen Tiefe und entscheidet am Ende erneut anhand seiner heuristischen Prinzipien, ob die Suche fortgesetzt werden soll oder nicht. Ein Abbruch erfolgt erst dann, wenn eine verhältnismäßig stabile Stellung erreicht ist. Schließlich macht das Programm den Zug, von dem es annimmt, daß er – optimales Spiel des Gegners vorausgesetzt – zum höchstmöglichen Punktestand für es selbst führt.

Beim Schach gibt es in jeder Stellung im Durchschnitt 35 mögliche Züge. Eine erschöpfende Suche bis zu einer Tiefe von nur drei Zügen pro Spieler würde daher schon die Bewertung von über 1,8 Milliarden Stellungen erfordern. Obwohl inzwischen rigorose Methoden zur Beschneidung des Suchbaums entwickelt wurden, tun die meisten Schachprogramme auch heute im Grunde nichts anderes, als den gestutzten Suchbaum in extenso zu durchforsten. Weltmeister im Computer-Schach ist zur Zeit „Belle", ein von Ken Thompson und Joe Condon von den Bell-Laboratorien entwickeltes Programm. Es läuft auf einem Computer, dessen Hardware speziell für Schachberechnungen ausgelegt ist, und prüft im Durchschnitt 160 000 Stellungen pro Sekunde. Belle spielt in Turnieren, als hätte sie eine Elozahl von 2160 (ab 2000 darf ein Spieler den Titel „Schachmeister" führen). Damit wird sie in der Spielstärke nur noch von wenigen Großmeistern (mit Elozahlen ab 2200) überboten.

Künstliche Intelligenz und Psychologie

Die Spielweise von Belle und ähnlichen Schachprogrammen illustriert eine der wichtigsten Fragen, die sich den Vertretern der Künstlichen Intelligenz heute stellt. Wenn eines der Hauptziele ihrer Arbeit darin besteht, intelligentes menschliches Verhalten nachzuahmen, auf welcher Ebene soll diese Simulation dann erfolgen? In gewissem Sinn simuliert ein Programm, das – egal wie – Schach auf Meisterebene spielt, zwar ohne Frage das Verhalten eines menschlichen Schachmeisters. Besteht das Ziel allerdings darin, die strategischen Überlegungen eines menschlichen Spielers nachzuahmen, dann muß ein Programm wie Belle als kompletter Fehlschlag gewertet werden.

Vieles deutet darauf hin, daß erfahrene Schachspieler völlig anders vorgehen als Belle. Ein menschlicher Spieler denkt strategisch. Er entscheidet sich für ein Ziel, etwa die Eroberung einer bestimmten Figur, und sucht dann nach Wegen, dieses Ziel zu erreichen. Dazu muß er sich möglicherweise zunächst Teilziele setzen und prüfen, ob diese erreichbar sind. Tests haben ergeben, daß sich ein guter Spieler höchstens hundert Züge überlegt, ehe er sich für einen entscheidet. Nur die aussichtsreichsten Fortsetzungen werden verfolgt – aber das bis zu einer relativ großen Tiefe. Belle dagegen prüft in den drei Minuten, die ihr beim Turnier im Mittel pro Zug zur Verfügung stehen, rund 29 Millionen Stellungen.

Belles Erfolg auf dem Schachbrett macht klar, daß intelligentes Verhalten auf Prozessen beruhen kann, die grundverschieden sind von den menschlichen Erkenntnismechanismen. Einige Forscher behaupten nichtsdestoweniger, die Wissenschaft von der Künstlichen Intelligenz sei, grob gesprochen, letztlich nur ein Zweig der Psychologie des Menschen. Sie gehen davon aus, daß jede erfolgreiche Simulation einer intellektuellen Leistung des Menschen durch ein Computerprogramm einen „Existenzbeweis" für das Computermodell der menschlichen Intelligenz erbringe. Auch wenn sich der Ablauf des Programms so sehr von mentalen Prozessen unterschei-

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (photo copyright © by www.schaakcomputers.nl/) (600 dpi)

de, daß es keinen direkten Beitrag zur Psychologie zu leisten vermöge, könne es doch zumindest gewichtige Aspekte der menschlichen Intelligenz simulieren, etwa die Schwierigkeit, ihre Leistungen genau vorherzusagen. Schachprogramme spielen oft besser als ihre Erfinder, und so ist es höchst irreführend zu erklären, Computer seien nur einer beschränkten Intelligenz fähig, weil sie nichts anderes zu tun vermöchten, als was der Programmierer ihnen eingetrichtert habe. In vielen Fällen aber weiß ein Programmierer gar nicht, was sein Programm alles kann, bevor es nicht gelaufen ist.

Ein schlagendes Beispiel dafür liefert ein Programm, das Donald Michie und Tim Niblett von der Universität Edinburgh sowie Ivan Bratko von der Edward-Kardelj-Universität in Ljubljana aufgestellt haben. Es analysiert speziell Schachendspiele, wo sich die meisten Schachprogramme wesentlich schlechter schlagen als im Mittelspiel. Für das Endspiel gibt es einen Satz von Regeln, die jeder bessere Schachspieler einfach im Kopf haben muß. Interessanterweise hat nun das Programm der drei Wissenschaftler gezeigt, daß in gewissen Endspielstellungen, von denen man früher glaubte, sie führten zum Remis, eine Seite den Sieg erzwingen kann. Die dazu nötigen Regeln sind allerdings so kompliziert, daß sie sich ein Mensch nur schwer einprägen kann. Der Computer tut sich da leichter.

Planung

Auch zielgerichtetes Planen – wiewohl kein hervorstechendes Merkmal von Schachprogrammen – gehört zu jenen Intelligenzleistungen, die im Prinzip leicht zu programmieren sind. Wieder läuft die Aufgabe auf das Erforschen einer auf dem Kopf stehenden baumartigen Struktur hinaus (Bild 2).

Die Wurzel verkörpert diesmal das Ziel, und die Knoten, von denen die Äste abzweigen, repräsentieren Vorbedingungen, die erfüllt sein müssen, ehe das Ziel erreichbar ist. Jede Vorbedingung kann ihrerseits zum Teilziel mit eigenen Vorbedingungen – verkörpert durch weitere Verzweigungspunkte –

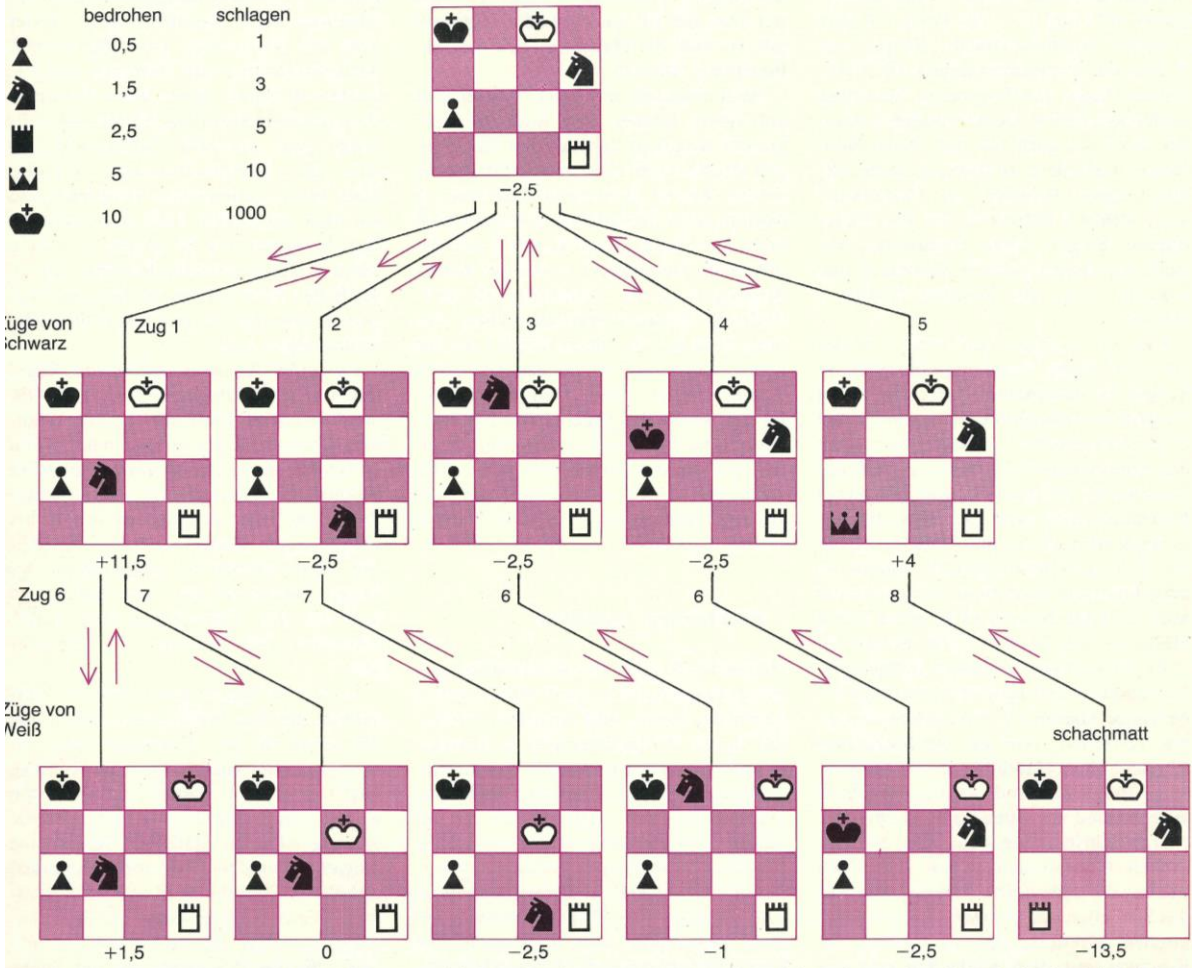

Bild 3: Die Baumstruktur einer Stellung auf einem vereinfachten Schachbrett zeigt, wie ein Computerprogramm (es führt die schwarzen Steine) seinen nächsten Zug ermittelt. Nacheinander untersucht es alle ihm möglichen Züge und die jeweiligen Erwiderungen des Gegners. Dabei durchkämmt es den Suchbaum bis zu einer Tiefe, die von heuristischen Regeln bestimmt wird. Für jede Stellung errechnet es nach einem Bewertungsschema (links oben) eine Punktzahl. Der Ausgangsstellung werden so −2,5 Punkte zugeordnet: die Summe aus 1004 Punkten für die schwarzen Figuren, −1005 Punkten für die weißen Steine und −1,5 Punkten für die Bedrohung des schwarzen Springers. Am Ende wählt das Programm den Zug, der auf der tiefsten untersuchten Ebene bei optimalem Spiel von Weiß zur höchsten Punktzahl für Schwarz führt. Hier spielt es Zug 1 und erwartet, daß Weiß mit Zug 7 antwortet. Die heuristischen Regeln veranlassen das Programm, die Suche entlang eines Astes abzubrechen, sobald es einen Gegenzug von Weiß findet, der zu einem schlechteren Punktestand für Schwarz führt als die beste weiße Erwiderung auf einen bereits untersuchten Zug von Schwarz. Der beste Gegenzug von Weiß auf Zug 1 ergibt 0 Punkte für Schwarz. Da das Programm zu den Zügen 2, 3, 4 und 5 Gegenzüge von Weiß findet, die zu Stellungen mit weniger als 0 Punkten für Schwarz führen, verfolgt es diese Züge nicht weiter.

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (600 dpi)

werden. Die Spitzen des Baumes stellen die im Ausgangszustand erfüllten Vorbedingungen dar. Mit jedem Knoten ist zugleich eine Handlung verbunden, durch die sich ein Ziel oder Teilziel erreichen läßt. Das Programm durchforstet alle Äste des Baumes, bis sämtliche Vorbedingungen erfüllt sind. Danach entwirft es einen Aktionsplan in Form eines Pfades, der von den Spitzen des Baumes zur Wurzel führt.

Ein Programm, das ein anschauliches Beispiel für diese Vorgehensweise in einem sorgsam umgrenzten Bereich liefert, hat Terry A. Winograd in den späten sechziger und frühen siebziger Jahren am Massachusetts Institute of Technology entwickelt. Es heißt SHRDLU (nach den siebt- bis zwölfthäufigsten Buchstaben im englischen Alphabet), und sein Betätigungsfeld ist die „Block-" oder „Klötzchenwelt" (Bild 1) – ein Tisch, auf dem einfache dreidimensionale Körper wie Würfel und Pyramiden stehen. (In Wirklichkeit kann das Programm allerdings weder mit realen Gegenständen hantieren, noch sie auch nur auf einem Bildschirm darstellen; es führt alle seine Manipulationen lediglich „in Gedanken" aus.) SHRDLU antwortet auf die eingetippten Fragen eines Benutzers, verstellt auf dessen Geheiß Klötzchen und berichtet über das Resultat (Bilder 5 und 6).

Um zu vestehen, auf welche Weise SHRDLU einen Handlungsablauf plant, wollen wir annehmen, daß es nur zwei Tätigkeiten beherrscht, die STAPELN und HERUNTERNEHMEN heißen sollen. Auf das Kommando STAPELN hin ergreift das Programm (in Gedanken, wie gesagt) ein Klötzchen, hebt es in die Höhe, bewegt es über ein anderes und stellt es darauf ab. Bei HERUNTERNEHMEN nimmt es umgekehrt ein Klötzchen, das auf einem anderen liegt, bewegt es zu einer freien Stelle auf dem Tisch und setzt es dort ab.

In Wirklichkeit umfassen Tätigkeiten wie STAPELN und HERUNTERNEHMEN eine ganze Hierarchie von untergeordneten Aktionen. Auf der nächsttieferen Ebene stehen Handlungen wie ERGREIFEN, BEWEGEN und ABSETZEN. Aber auch sie sind aus einer Reihe noch einfacherer Schritte aufgebaut. Die elementarsten Operationen wären schließlich solche, die sich durch Steuersignale für das Schließen der „Finger" eines mechanischen Greifers, das Feststellen des Kontakts mit dem Klötzchen und das Drehen eines Gelenks ausdrücken ließen. Die Hierarchie der Anweisungen des Programms spiegelt dabei genau die hierarchische Gliederung menschlicher Tätigkeiten wider, angefangen von den im Bewußtsein formulierten Zielen bis hinab zur akribischen Steuerung der Muskelbewegungen. Von nun an werde ich diese Details außer acht lassen und die vom Programm entworfenen Pläne nur noch auf der höchsten Ebene diskutieren.

Die Aktion STAPELN ist an zwei Vorbedingungen geknüpft: Die Oberseite jedes der beiden Gegenstände, die den Stapel bilden sollen, muß frei sein. Ebenso hängt auch HERUNTERNEHMEN von zwei Vorbedingungen ab, nämlich daß auf dem Objekt, von dem etwas heruntergenommen werden soll, überhaupt ein anderes Klötzchen liegt und daß dessen Oberfläche frei ist. Bei der Suche nach einem gangbaren Weg zu einem Ziel oder Teilziel vergleicht das Programm dieses Ziel oder Teilziel mit dem Resultat einer bestimmten Handlung, das heißt mit dem Zustand der Klötzchenwelt, der aus der betrachteten Handlung hervorgeht. Deckt sich dieser Zustand mit dem, der für das Ziel oder Unterziel gilt, so wird die Handlung Teil des Aktionsplans (Bild 4).

Angenommen, ein roter Würfel sitzt auf einem blauen. Wie muß das Programm vorgehen, um mit den Handlungen STAPELN und HERUNTERNEHMEN die beiden Würfel zu vertauschen? Nun, es beginnt beim Zielzustand und sucht nach einer Handlung, deren Resultat sich damit deckt. Dann steckt es sich die Vorbedingungen dieser Handlung als neue Teilziele. Konkret: Mit der Aktion STAPELN ließe sich der blaue Würfel auf den roten stellen (und damit das Ziel erreichen), wenn die Oberseiten beider Würfel frei wären. Im gegebenen Ausgangszustand ist die Oberseite des roten Würfels frei, die des blauen aber nicht. Also steckt sich das Programm als neues Teilziel, die Oberseite des blauen Würfels frei zu machen.

Rückwärtiges Verketten

Als nächstes sucht es nach Handlungen, mit denen sich die festgestellten Teilziele erreichen lassen, und bestimmt wiederum deren Vorbedingungen – solange, bis es bei den Ausgangsbedingungen angekommen ist. Die Oberseite des blauen Würfels ließe sich beispielsweise mit der Aktion HERUNTERNEHMEN freimachen. Da deren Vorbedingungen im Ausgangszustand erfüllt sind, ist die Suche in unserem Fall beendet. Diesen Prozeß, bei dem ausgehend vom Zielzustand im Rückwärtsgang eine Folge von Handlungen bis hin zum Ausgangszustand konstruiert wird, bezeichnet man als rückwärtiges Verketten.

Zur Aufstellung des gewünschten Aktionsplanes braucht das Programm jetzt nur noch die Reihenfolge der gefundenen Handlungen umzukehren. Als erstes wird also durch HERUNTERNEHMEN des roten Würfels die Oberfläche des blauen freigemacht und dann durch STAPELN der blaue Würfel auf den roten gesetzt. Entscheidend dabei ist, daß von einem Freimachen der Oberfläche des blauen Würfels in der Anweisung an das Programm nicht die Rede war. Es ergibt sich einfach als Nebeneffekt.

Eine der faszinierendsten Anwendungen des rückwärtigen Verkettens bilden jüngst entwickelte Computerprogramme – sogenannte Expertensysteme –, die über das Wissen von Spezialisten verfügen und es in der direkten Kommunikation mit dem Benutzer auch ebensogut an den Mann zu bringen verstehen wie der Spezialist selbst. Solche Programme haben bereits in der medizinischen Diagnose, bei der Erkundung von Bodenschätzen sowie als Steuerberater Einsatz gefunden. Wenn es beispielsweise um die Diagnose einer Krankheit geht, tastet sich das Programm von allgemeinen Auskünften über das Befinden des Patienten zu immer spezielleren Untersuchungsbefunden zurück, bis die Indizienkette eine plausible Schlußfolgerung über die Krankheitsursache erlaubt. Statt eines Aktionsplans wird hier also ein Plan entwickelt, nach dem sich aus den Aussagen des Benutzers am Ende auf eine Krankheit schließen läßt. An die Stelle der Aktionen treten dabei im Programm gespeicherte Regeln für mögliche Schlußfolgerungen.

Ein gutes Beispiel einer solchen Regel findet sich im Diagnose-System MYCIN, das Edward H. Shortliffe von der Stanford-Universität geschrieben hat. „Wenn es sich bei der Infektion um eine primäre Bakteriämie handelt und die kultivierte Probe aus einer normalerweise sterilen Körperregion stammt und die vermutliche Eintrittsstelle der Organismen der Magen-Darmtrakt ist, dann handelt es sich bei den Organismen sehr wahrscheinlich (70 Prozent) um Bacteroides."

Da solche Regeln große formale Ähnlichkeit mit den Erklärungen haben, die Menschen für ihre Entscheidungen anführen, lernen auch computerunkundige Experten relativ schnell, ihre eigenen Regeln computergerecht zu formulieren. Wichtig ist auch, daß jede Regel eine Angabe über den Grad ihrer Zuverlässigkeit (70 Prozent im zitierten Fall) enthält. Diese Angabe dient als Entscheidungsgrundlage für heuristische Prinzipien, die das Programm auf der Suche nach einer plausiblen Krankheitsursache leiten, und wird auch dem Benutzer mitgeteilt, damit er sich ein eigenes Bild vom Wert der Computer-Diagnose machen kann. Außerdem muß das Programm erklären, warum es eine bestimmte Schlußfolgerung gezogen hat. Niemand würde einem Computer ver-

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982)

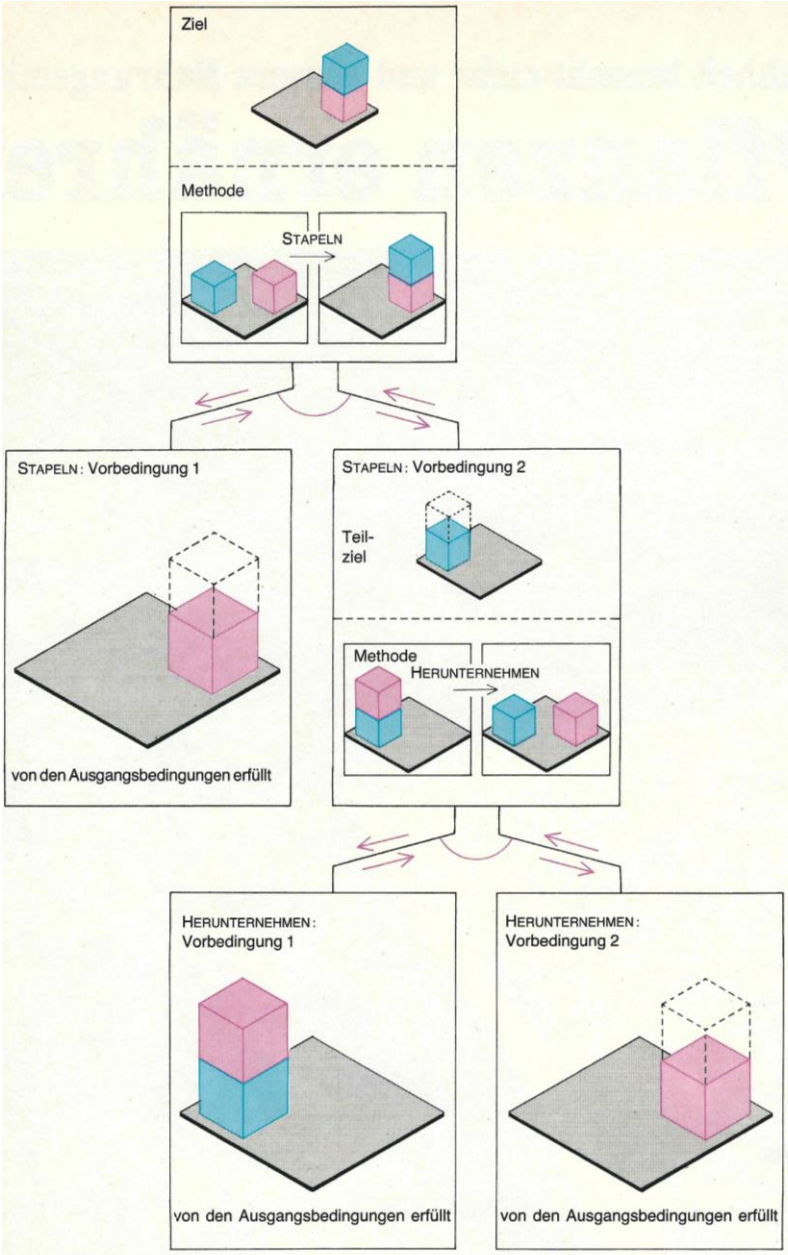

Bild 4: Die Aufstellung eines Aktionsplans in der Klötzchenwelt erfordert das Erkennen eines Ziels und die Auswahl einer Folge von Handlungen, mit denen sich das Ziel vom Ausgangszustand aus erreichen läßt. Im gezeigten Beispiel liegt im Ausgangszustand das rote Klötzchen auf dem blauen, und das Ziel besteht darin, das blaue Klötzchen auf das rote zu stellen. Um einen Weg vom Ausgangs- zum Zielzustand zu finden, unternimmt das Programm eine Suche, die der Erforschung einer Baumstruktur entspricht. Ausgehend von der Wurzel des Baumes, also dem Ziel, sucht das Programm in seinem Tätigkeitsrepertoir nach einer Handlung, deren Ergebnis mit dem Zielzustand übereinstimmt. Hier ist es die Aktion STAPELN. Sie wird daher dem Zielzustand zugeordnet. Wie die beiden von der Wurzel abzweigenden Äste anzeigen, läßt sich diese Operation allerdings nur durchführen, wenn zwei Vorbedingungen erfüllt sind: Die Oberseite sowohl des roten als auch des blauen Klötzchens muß frei sein. Die erste Voraussetzung ist bereits im Ausgangszustand gegeben, die zweite nicht. Daher verwandelt das Programm die Vorbedingung 2 in ein Teilziel und sucht nun nach einer Handlung, mit der es die Oberfläche des blauen Klötzchen freibekommt. Es findet sie in der Aktion HERUNTERNEHMEN. Deren zwei Vorbedingungen – daß sich auf dem blauen Klötzchen überhaupt ein anderes befindet und daß die Oberseite des roten Klötzchens frei ist – werden beide vom Ausgangszustand erfüllt. Damit braucht das Programm nur noch die Reihenfolge der gefundenen Handlungen umzukehren, und der Aktionsplan steht. Ist die Erreichung eines gewünschten Ziels an mehrere Vorbedingungen geknüpft, so werden die entsprechenden Äste wie hier durch einen Bogen miteinander verbunden.

trauen, der nur seine Diagnose ausdruckt und keine Begründung dafür mitliefert.

Lernen in der Klötzchenwelt

Solange ein Computer seine Kenntnisse nicht durch „Erfahrung" vertiefen kann, wird seine Leistung allein vom Wissen, der Voraussicht und Gründlichkeit des Programmierers bestimmt. Lernen zählt daher zu den zentralen Anliegen der Künstlichen Intelligenz. Eine Reihe von Wissenschaftlern haben Winograds Klötzchenwelt als Experimentierfeld zur Erforschung von Lernstrategien benutzt.

Eine Art des Lernens besteht darin, sich einen einmal entwickelten Plan für ähnliche Situationen in der Zukunft zu „merken". Richard E. Fikes und Nils J. Nilsson am Stanford Research Institute haben diese Idee erstmals in ihrem Programm STRIPS (Kürzel für: **S**tanford **R**esearch **I**nstitute **P**roblem **S**olver) angewendet. Angenommen, ein Programm soll einen Zustand herbeiführen, bei dem ein braunes Klötzchen auf einem grünen liegt, und findet genau die umgekehrte Situation vor. Nachdem es bereits früher einmal ein rotes und ein blaues Klötzchen vertauscht hat, verfügt es im Speicher über eine computergerechte Darstellung des damals aufgestellten Plans. Damit es darauf zurückgreifen kann und sich nicht daran stört, daß die neuen Klötzchen anders gefärbt sind als die alten, muß diese interne Darstellung allerdings allgemein genug gehalten sein. Zu diesem Zweck werden die Namen der zu vertauschenden Klötzchen als variable Größen gespeichert. Die gesamte Aktion, sie sei hier VERTAUSCHEN genannt, geht als feste Abfolge aus den einfacheren Tätigkeiten ERGREIFEN, BEWEGEN und ABSETZEN mit den entsprechenden Vorbedingungen und Resultaten in das Tätigkeitsrepertoire des Programms ein. Da der Ausgangszustand der braunen und grünen Klötzchen die Vorbedingungen der Operation VERTAUSCHEN erfüllt, braucht das Programm keine Suche anzustellen, sondern kann das Ziel direkt erreichen.

Eine andere Art, Erfahrungen zu sammeln, ist das Lernen durch Versuch und Irrtum. Es wird von einem Programm namens HACKER nachgebildet, das Gerald J. Sussman vom Massachusetts Institute of Technology geschrieben hat. (Im Informatiker-Jargon ist ein Hacker jemand, der die meiste Zeit mit dem Schreiben von Computer-Programmen zubringt.) HACKER besteht aus einem Planungssystem ähnlich SHRDLU, „Kritikern", die die Planung überwachen und Probleme registrieren, und „Debuggern" (wörtlich: „Entlausern"), die neue Regeln aufstellen, um eine Wiederho-

lung der von den Kritikern entdeckten Fehler zu verhindern.

Angenommen, ein rotes Klötzchen liegt auf einem blauen, und ein grünes befindet sich daneben frei auf dem Tisch. HACKER wird aufgefordert, einen Stapel aus drei Klötzchen zu bilden, wobei das rote oben, das grüne in der Mitte und das blaue unten liegen soll. Das Programm hat dafür nur die Operationen STAPELN, HERUNTERNEHMEN und VERTAUSCHEN zur Verfügung. Nach seinem bekannten Suchschema zerlegt es das Ziel zunächst in Teilziele. Beim ersten Teilziel hat es dabei die Wahl zwischen zwei Möglichkeiten: Es kann entweder das rote Klötzchen auf das grüne oder das grüne auf das blaue legen. Angenommen, es entscheidet sich zufällig für das falsche Teilziel, nämlich das rote Klötzchen auf das grüne zu setzen. Das erreicht es durch die Aktion STAPELN. Das nächste Teilziel muß dann darin bestehen, das grüne Klötzchen auf das blaue zu stellen. Das grüne Klötzchen aber läßt sich solange nicht ergreifen, wie das rote obenauf liegt. Also muß das Programm das rote Klötzchen wieder vom grünen herunternehmen und dabei das gerade erreichte Teilziel zerstören.

Sobald die Kritiker nun entdecken, daß ein Teilziel zerstört wurde, rufen sie die Debugger herbei. Diese sehen sich die Sache an und halten nach anderen Planungsmöglichkeiten Ausschau. Dabei stellen sie in unserem Fall fest, daß keine Errungenschaft rückgängig gemacht werden muß, wenn als erstes das grüne Klötzchen auf das blaue gelegt wird. Daraus können sie schließen, daß sich überflüssige Operationen – wenigstens manchmal – durch Änderung der Reihenfolge von Teilzielen vermeiden lassen. HACKER merkt sich diese Erkenntnis als neue, generelle Entlausungsmethode. Auf diese Weise lernte Sussmans Programm schließlich, die Teilziele für beliebig hohe Stapel in die richtige Reihenfolge zu bringen.

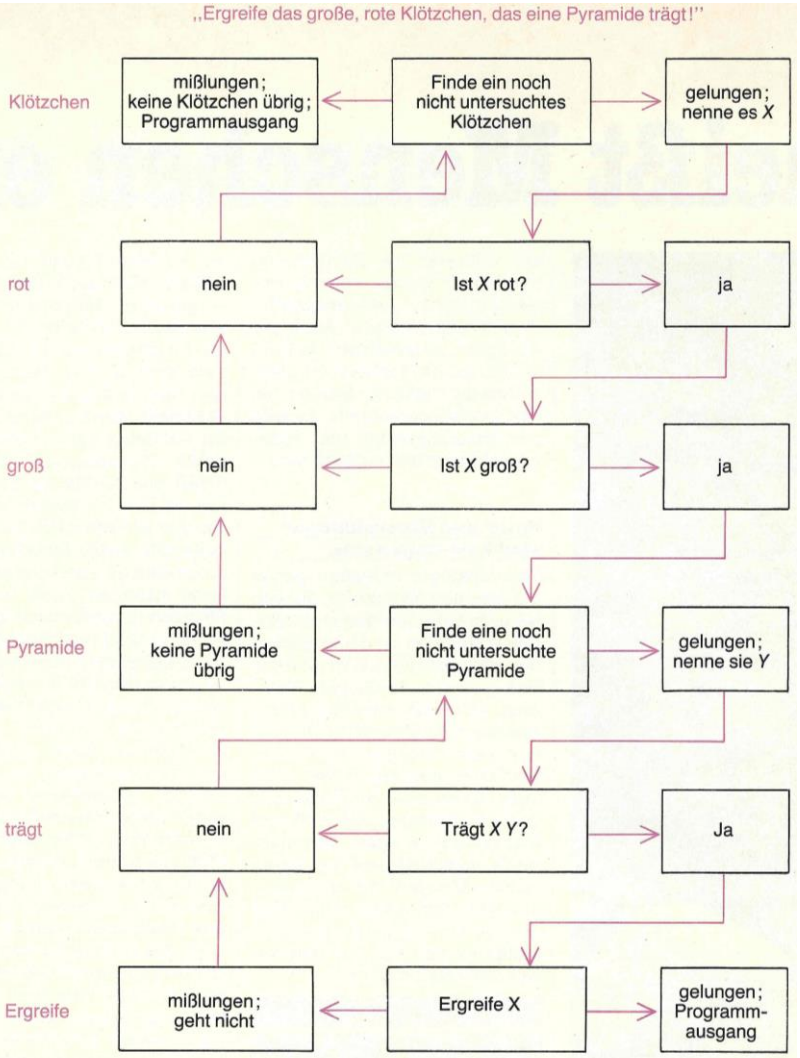

Bild 5: SHRDLU versteht die ihm in natürlicher Sprache erteilten Anweisungen, indem es Prozeduren wie der in diesem Flußdiagramm gezeigten folgt. Die Schlüsselworte in dem Kommando „Ergreife das große rote Klötzchen, das eine Pyramide trägt" setzen eine Folge von Schritten in Gang, durch die bestimmt wird, welches Klötzchen gemeint ist. Anschließend stellt das Programm einen Plan auf, um den Befehl auszuführen.

Erlernen von Begriffen

Eine Reihe von Programmen verleihen einem Computer die Fähigkeit, eine Handlung von sich aus sachgerechter durchzuführen oder zu einer angemesseneren Darstellung eines Begriffs zu gelangen. Eines der ersten Programme, das seine Leistungsfähigkeit selbsttätig zu steigern vermochte, war ein Dameprogramm, das Arthur L. Samuel von der International Business Machines Corporation um 1960 schrieb. Es lernte in Anlehnung an die Mechanismen der Evolution. Samuel ließ jeweils zwei Versionen des Programms, die eine Stellung nach einem leicht unterschiedlichen Punktsystem bewerteten, gegeneinander antreten. Der Sieger in diesem direkten Wettbewerb zweier Bewertungssysteme diente als Ausgangsbasis für weitere Variationen.

In den späten sechziger Jahren schrieb Patrick H. Winston vom Massachusetts Institute of Technology ein Programm, das nach und nach seine Vorstellung von bestimmten Begriffen verbessert. Ein Lehrer präsentiert dem Programm eine sorgfältig abgestufte Folge von Szenen, die Beispiele oder Gegenbeispiele für den vom Programm zu lernenden Begriff sind. Das Programm analysiert jeweils die Szene und bildet eine „Hypothese" darüber, welche ihrer Komponenten und welche Beziehungen zwischen ihnen wesentliche Aspekte des Begriffs beinhalten. Anhand jeder neuen Szene überprüft und modifiziert es seine früheren Hypothesen.

Angenommen, das Programm bekommt eine einfache Zeichnung dargeboten und gesagt, es handle sich um ein Beispiel für einen Bogen (Bild 7). Es stellt fest, daß das dargestellte Objekt aus drei Klötzchen besteht, von denen zwei senkrecht auf dem Tisch stehen, während das dritte flach darübergelegt ist. Obwohl das Programm vielleicht noch viele andere Merkmale erraten hätte, die für Bögen im allgemeinen gelten, sorgt man bewußt dafür, daß es sich zunächst nur einen sehr ungenauen Begriff von einem Bogen macht, damit es ihn

Würdest du bitte einen Stapel aus den beiden roten Klötzchen und entweder einem grünen Würfel oder einer Pyramide bilden?
O.K.

Welcher Würfel liegt auf dem Tisch?
Der große grüne, der die rote Pyramide trägt.

Lege einen kleinen Würfel auf den grünen Würfel, der eine Pyramide trägt.
O.K.

Lege die kleinste Pyramide darauf.
O.K.

Wieviele Gegenstände liegen auf grünen Würfeln?
Ich bin mir nicht sicher, was du mit dem Wort „auf" in der Wendung „auf grünen Würfeln" meinst. Meinst du:
1 – direkt auf der Oberfläche
2 – irgendwo obendrauf?

2
Drei

Hast du irgendeine Pyramide angefaßt, bevor du die grüne auf den kleinen Würfel gelegt hast?
Ja, die grüne

Wann hast du sie angefaßt?
Als ich einen Stapel aus dem roten Würfel, einem großen, roten Klötzchen und einem großen, grünen Würfel bildete.

Warum?
Um sie los zu werden.

Warum hast du das getan?
Um die Oberfläche des roten Würfels frei zu machen.

Warum hast du die Oberfläche dieses Würfels frei gemacht?
Um ihn auf einen großen, grünen Würfel zu legen.

Warum hast du das getan?
Um einen Stapel aus dem roten Würfel, einem großen, roten Klötzchen und einem großen, grünen Würfel zu bilden.

Warum hast du das getan?
Weil du es mich geheißen hast.

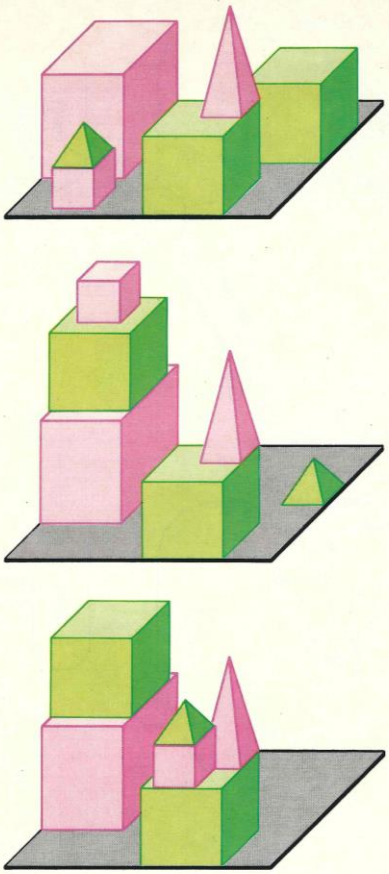

Bild 6: Die extreme Beschränktheit der Klötzchenwelt ermöglicht dem Programm SHRDLU eine Fülle flexibler Reaktionen auf Wünsche und Fragen des Programmierers. Auf dessen Anweisung kann es beispielsweise einen Plan entwickeln, um die Gegenstände in seiner Welt in der gewünschten Weise umzustellen, und sodann die erforderlichen Manipulationen vornehmen. Hier überführt es die Klötzchenwelt entsprechend den Kommandos des Programmierers (schwarze Schrift links oben) aus dem rechts oben gezeigten Ausgangszustand in die darunter abgebildeten Konfigurationen. Außerdem beantwortet es (rote Schrift) Fragen nach der gegenwärtigen Anordnung der Klötzchen, nach früher ausgeführten Handlungen sowie nach den Gründen für diese Handlungen. Bei mehrdeutigen Fragen oder Anweisungen kann es von sich aus Rückfragen stellen. SHRDLU war eines der ersten Programme, das Sprachverständnis und die Simulation planvoller Tätigkeiten miteinander verband.

anhand der späteren Szenen weiter präzisieren kann.

Das zweite Bild ist kein Bogen. Wieder erstellt das Programm eine Beschreibung des Gesehenen und vergleicht sie mit der aus dem ersten Beispiel abgeleiteten Hypothese. Wenn sich die zweite Szene nicht zu stark von der ersten unterscheidet, kann das Programm so ein wenig mehr von dem herausfinden, was das Wesentliche an einem Bogen ausmacht. Besteht der einzige Unterschied zwischen erstem und zweitem Bild beispielsweise darin, daß das obere Klötzchen nun unten auf dem Tisch liegt, so schließt das Programm daraus, daß bei einem Bogen die beiden senkrechten Klötzchen stets das dritte tragen müssen. Es hat also gelernt, daß das dritte Klötzchen im ersten Bild nicht zufällig auf den beiden anderen lag, und diese Erkenntnis wird in die Hypothese über den Begriff Bogen aufgenommen. Mit jeder neuen Zeichnung erfährt die Darstellung dieses Begriffs eine weitere Präzisierung, so daß das Programm schließlich lernt, selbst neue Beispiele von Bögen zu erkennen.

Jüngst hat Douglas B. Lenat von der Stanford-Universität ein Programm entwickelt, das den Namen AM (für **a**utomatisierte **M**athematik) trägt und fähig ist, aus rund hundert Grundbegriffen der Mengenlehre neue Begriffe und Lehrsätze zu formulieren. Heuristische Prinzipien legen fest, unter welchen wohldefinierten Bedingungen ein neuer Begriff aufgestellt werden darf. Ist ein solcher Begriff dann geschaffen, beginnt das Programm, ihn weiter zu untersuchen. Während eines Laufs nahm sich das Programm beispielsweise den Begriff Teiler vor. Es fand viele Zahlen mit vier oder mehr Teilern (so die Zahl 6 mit den Teilern 1, 2, 3 und 6), doch die heuristischen Regeln bewogen es, sich auf Zahlen mit nur wenigen Teilern zu beschränken.

Bei der Untersuchung der Zahlen mit genau drei Teilern stellte das Programm fest, daß alle betrachteten Beispiele auch Quadratzahlen waren. Überdies entdeckte es, daß die Quadratwurzel einer Zahl mit drei Teilern stets eine Zahl mit genau zwei Teilern ist. Auf Grund der Übereinstimmung zwischen dem Begriff „Quadratwurzel von Zahlen mit drei Teilern" und dem Begriff „Zahl mit zwei Teilern" erhöhte das Programm die Priorität beider Begriffe auf seiner Agenda und beschloß, Zahlen mit genau zwei Teilern näher zu untersuchen. So begann es, das ergiebige Feld der Primzahlen – wie der Mathematiker Zahlen mit genau zwei Teilern nennt – zu erforschen. In nur einer Stunde Laufzeit reproduzierte es mehrere wohlbekannte Vermutungen über Primzahlen und stellte die Hypothese auf, daß sich jede natürliche Zahl eindeutig in ein Produkt von Primzahlen zerlegen lasse.

Fortschreitende Einschränkung

Die Art von Suche, die ich bei Schach- und Planungsprogrammen geschildert habe, ist ein serieller oder Reihenprozeß, bei dem die einzelnen Äste und Knoten nacheinander verfolgt und geprüft werden. Schneller käme das Programm voran, wenn es unabhängige Teilaspekte der Suche simultan nebeneinander ausführen könnte. Ein gut Teil der Signalverarbeitung im Gehirn läuft beispielsweise derart parallel ab, und viele Experten sind der Ansicht, daß Programme, die eine Parallelverarbeitung vorsehen, zu einer viel besseren Simulation der sensorischen Fähigkeiten des Menschen imstande wären als solche, die nur seriell vorgehen können. Ein Parallelprozeß, den man sich bei einer Reihe experimenteller Programme in der Künstlichen Intelligenz heute zunutze macht, ist die sogenannte fortschreitende Einschränkung. Eines der ersten Beispiele dafür habe ich 1972 in meiner Doktorarbeit vorgelegt. Ich schrieb mein Programm in LISP, einer Programmiersprache, die heute auf dem Gebiet der Künstlichen Intelligenz fast ausnahmslos verwendet wird.

Zu den Dingen, die der Mensch spielend beherrscht, zählt die Fähigkeit, eine zweidimensionale Zeichnung als dreidi-

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982)

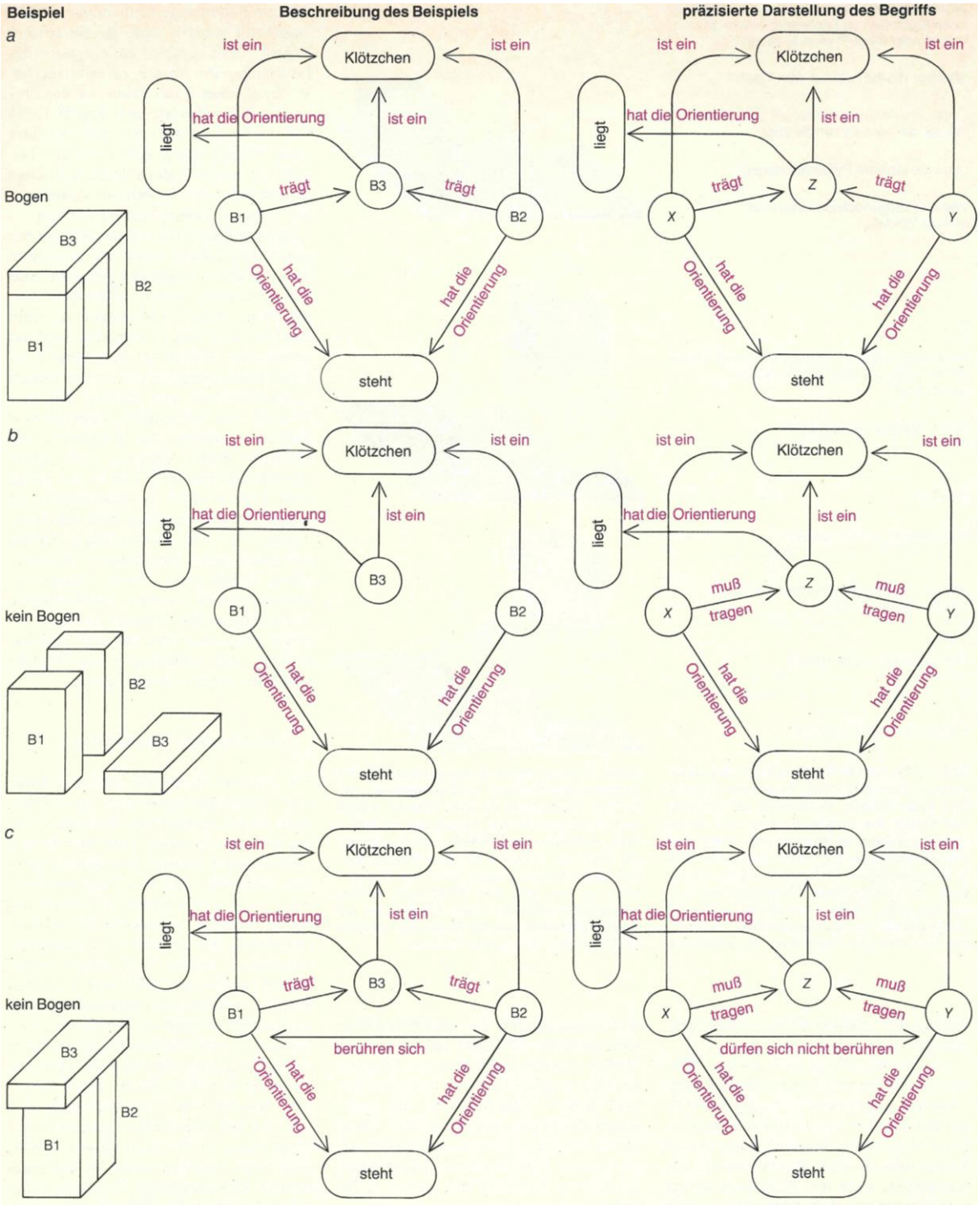

Bild 7: Das Lernen von Begriffen erfordert die Abstraktion von den zufälligen Eigenschaften und das Aufspüren der wesentlichen Merkmale der Verkörperungen eines solchen Begriffs. Auch Computer können Begriffe lernen, indem sie eine Hypothese über die für den Begriff wesentlichen Aspekte eines bestimmten Beispiels bilden und diese Hypothese anhand späterer Beispiele oder Gegenbeispiele revidieren und präzisieren. Um etwa einem Programm beizubringen, was ein Bogen ist, präsentiert ihm der Lehrer das Bild eines Bogens (*a*). Das Programm beschreibt das Bild mit Hilfe eines Bedeutungsnetzes, das die Beziehungen zwischen den bereits im Programm gespeicherten Begriffen wiedergibt. Die erste Hypothese ist ein verallgemeinertes Bedeutungsnetz, in dem die Variablen durch die Bestandteile des vorliegenden Bildes ersetzt wurden. Anschließend zeigt der Lehrer dem Computer zwei Bilder, die keine Bögen darstellen (*b*, *c*). Indem das Programm auch diese Bilder beschreibt und mit dem ersten Bild vergleicht, kann es anhand der Unterschiede erkennen, welche Aspekte des ersten Bildes wesentlich für einen Bogen waren und welche nicht. Entsprechend modifiziert es seine Hypothese. Die Folge von Beispielen muß dabei sorgfältig durchdacht sein. Ein Beispiel, das sich zu stark von seinem Vorgänger unterscheidet, würde das Programm verwirren.

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (600 dpi)

mensionales Bild zu deuten. In meiner Doktorarbeit setzte ich mir zum Ziel, ein Programm zu entwickeln, das die gleiche Fertigkeit besitzt. David A. Huffman vom Massachusetts Institute of Technology und der verstorbene Max B. Clowes von der Universität Sussex hatten bereits gezeigt, daß sich gewisse einfache Zeichnungen dann als dreidimensionale Bilder interpretieren lassen, wenn es gelingt, sie mit Hilfe eines Satzes von 16 zulässigen Schnittpunkttypen konsistent zu bezeichnen (Bild 8). Dabei mußten die dargestellten Objekte allerdings ebene Oberflächen besitzen, und an keinem Schnittpunkt durften mehr als drei Ebenen aneinanderstoßen. Jede von einem Schnittpunkt ausgehende Kante war entweder als Randkante oder als konvexe (auf den Betrachter hin zeigende) oder konkave (vom Betrachter weg zeigende) Innenkante zu klassifizieren. (Als Innenkanten gelten dabei alle Kanten, an denen zwei sichtbare Flächen aneinanderstoßen, während bei Randkanten nur eine Fläche sichtbar ist.)

Eine Darstellung läßt sich dann konsistent bezeichnen oder – wie man sagt – „etikettieren", wenn, nachdem alle Schnittpunkte klassifiziert sind, sämtliche Kanten nur eine Bezeichnung (ein „Etikett" oder „Label") tragen. Ergibt sich aus jeder möglichen Klassifikation zweier durch eine Kante verbundener Schnittpunkte dagegen stets eine widersprüchliche Bezeichnung für die Kante, so kann die Zeichnung kein reales dreidimensionales Objekt repräsentieren. Ein Beispiel für eine Zeichnung, die sich nicht konsistent etikettieren läßt, ist die sogenannte Teufelsgabel, die auf den ersten Blick wie ein dreidimensionales Gebilde aussieht, sich bei genauerem Hinsehen aber als verwirrende räumliche Paradoxie entpuppt (Bild 9). Das beschriebene Etikettierungsschema erlaubt einem Computer zu erkennen, daß es sich bei der Zeichnung nicht um die Darstellung eines dreidimensionalen Objektes handeln kann, und hilft auch menschlichen Betrachtern, die Ursache der Paradoxie zu durchschauen.

Die ersten Programme zur Klassifikation der Schnittpunkte einer Zeichnung führten eine serielle Baumsuche nach konsistenten Etiketten durch. Sie griffen sich einen Schnittpunkt heraus und ordneten ihm sämtliche möglichen Etiketten zu. Dann gingen sie weiter zum nächsten Schnittpunkt und prüften, welche der für ihn möglichen Bezeichnungen mit denen für den ersten Schnittpunkt konsistent waren. Für jede dieser Etiketten ermittelten sie wiederum die zulässigen Bezeichnungen für einen dritten, dem zweiten benachbarten Schnittpunkt. So wurden nach und nach alle Schnittpunkte durchgegangen, bis jeder Ast des Suchbaums erforscht war. Ein Ast endete, wenn keine konsistente Bezeichnung mehr möglich war oder alle Schnittpunkte bereits ein Etikett trugen. (Bei manchen Zeichnungen gibt es mehrere Arten, die Schnittpunkte konsistent zu klassifizieren. Solche Bilder erscheinen auch menschlichen Beobachtern mehrdeutig.)

Diese Methode funktioniert, solange die Zahl der Schnittpunkttypen klein ist. Mein Programm aber sollte auch einspringende Ecken, Schatten und andere Arten von Kanten unterscheiden können, mit denen frühere Programme nicht zurechtgekommen waren. Dazu sind allerdings wesentlich mehr Etiketten erforderlich: für bestimmte Schnittpunkte etwa 100, für andere über 1000. Bei nur vier Schnittpunkten gäbe es also bereits über 1000^4 oder eine Billion Bezeichnungsmöglichkeiten. Auch wenn bei einer vollständigen Suche nicht jede dieser Möglichkeiten berücksichtigt werden müßte (weil sich beispielsweise viele davon bereits als inkonsistent herausgestellt hätten, bevor noch der letzte Schnittpunkt an die Reihe käme), wäre ein solches Vorhaben undurchführbar.

Daher beschloß ich, daß mein Programm von vornherein möglichst viele unzulässige Bezeichnungen ausschließen sollte. Dazu bestimmt es zunächst die zulässigen Etiketten für jeden Schnittpunkt und betrachtet dann Schnittpunktpaare, die durch eine Linie miteinander verbunden sind (Bild 10). Normalerweise stellt sich dabei heraus, daß bestimmte Bezeichnungen für den einen Schnittpunkt mit keiner der möglichen Bezeichnungen für den Nachbarschnittpunkt verträglich sind (also grundsätzlich zu einer widersprüchlichen Doppeletikettierung der Verbindungslinien führen würden). Sie können daher an dieser Stelle bereits ausgeschieden werden. Da sich solche Paarvergleiche unabhängig voneinander vornehmen lassen, ist es möglich, diesen Bearbeitungsschritt parallel durchzuführen. Nach der ersten Ausscheidungsrunde werden neue Paare gebildet und diese wieder parallel auf Konsistenz der Etiketten geprüft. So schrumpft die Zahl möglicher Bezeichnungen von Runde zu Runde.

Was mir ursprünglich vorgeschwebt hatte, war nicht mehr und nicht weniger, als den Suchbaum mit Hilfe meines Paarausscheidungsverfahrens auf handliche Maße zu stutzen. Bei der ersten Zeichnung, an der ich mein Verfahren erprobte, stellte sich jedoch heraus, daß es vom ganzen Baum nur noch die Wurzel übrig ließ, das heißt die Zahl der Etiketten pro Schnittpunkt auf genau eins zusammenstrich. Die Suche selbst hatte sich damit erübrigt.

Das war eine Riesenüberraschung. Zunächst glaubte ich sogar, im Programm müsse irgendwo ein Fehler stekken, als es immer wieder eindeutige Etikettierungen lieferte. Heute kann ich von Glück sagen, daß ich für mein Verfahren ein Computerprogramm schrieb und es nicht etwa per Hand ausprobierte. Denn dann hätte ich seine bemerkenswerte Eigenschaft, eine Baumsuche überflüssig zu machen, höchstwahrscheinlich gar nicht entdeckt. Selbst für die einfachste Handetikettierung wäre es nämlich erforderlich gewesen, gut und gern 10 000 Schnittpunkt-Bezeichnungen zu untersuchen. Bei dieser Sisyphus-Arbeit hätten sich Fehler wohl kaum vermeiden lassen. Die Möglichkeit, durch ein Computerprogramm eine unerwartete Entdeckung zu machen, ist etwas, das in Nicht-Informatikerkreisen nur selten gewürdigt wird, obwohl es gar nicht so selten vorkommt.

Auch die anschließenden Versuche, mein Programm auf realistischere Bilder anzuwenden, machten deutlich, daß das Experiment wesentlicher Bestandteil der Forschung auf dem Gebiet der Künstlichen Intelligenz ist. Obwohl sich das Programm nämlich bei den Bildern, für die es konzipiert war, als elegant und effizient erwies, ließ es sich nur schlecht auf Bilder von Objekten ausdehnen, die eine gekrümmte Form, strukturierte oder glänzende Oberflächen sowie andere bei realen Gegenständen verbreitete Besonderheiten aufwiesen.

Die eindrucksvollsten jüngsten Erfolge auf dem Gebiet der „Computervision" hat der verstorbene David C. Marr vom Massachusetts Institute of Technology erzielt, indem er die Operationen modellierte, die der visuelle Cortex im Gehirn ausführt. In meiner Arbeit benutzte Techniken haben sich jedoch für viele Probleme – so die Analyse elektronischer Schaltkreise oder die räumliche Verschmelzung von zwei Ansichten derselben Szene – als ausgesprochen nützlich erwiesen. Außerdem helfen sie, Einzelheiten in einem Bildausschnitt mit solchen in einem benachbarten Ausschnitt zu vergleichen oder die Mehrdeutigkeit bestimmter sprachlicher Wendungen aufzulösen.

Sprachverständnis

Das Kommunikationsmedium zwischen Mensch und Computer bezeichnet man oft als „Sprache", obwohl es mit dem Verständigungsmittel, dessen wir Menschen uns bedienen, nur wenig gemein hat. Die Aufgabe, Computer so zu programmieren, daß sie natürliche Sprache verstehen, zählt zu den schwierigsten Herausforderungen an die Disziplin der Künstlichen Intelligenz. Selbst das primitivste Sprachverständnis erfordert um-

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982)

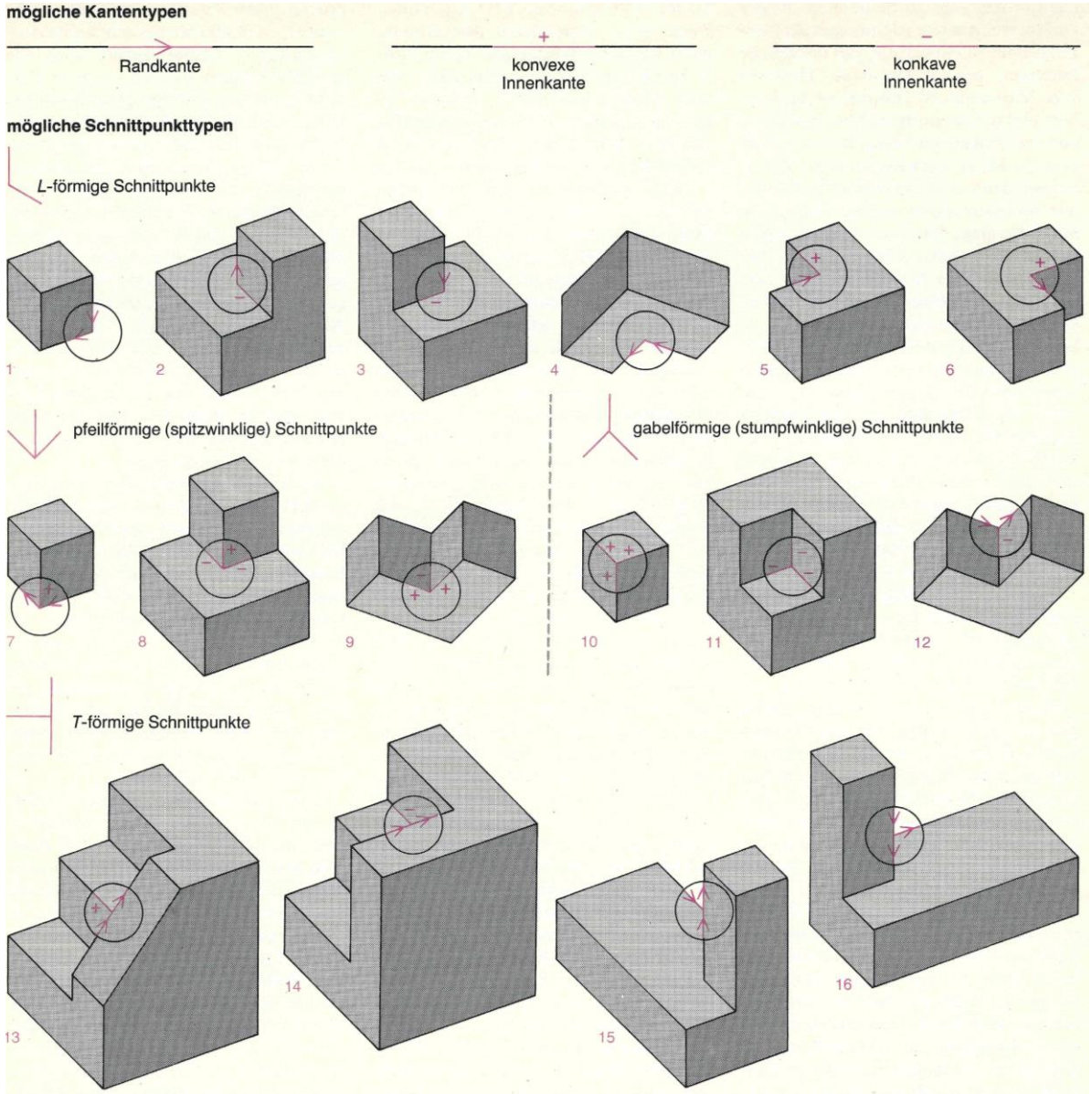

Bild 8: Um eine zweidimensionale Zeichnung als perspektivische Ansicht eines räumlichen Objektes zu erkennen, muß ein Programm ein Etikettierungsschema auf die Schnittpunkte der Zeichnung anwenden. Dabei geht es davon aus, daß jede Linie entweder eine Randkante oder aber eine auf den Betrachter zuspringende (konvexe) beziehungsweise von ihm weg zeigende (konkave) Innenkante darstellt. Bei Randkanten ist nur eine der beiden anstoßenden Flächen sichtbar, und der Pfeil, durch den solche Kanten gekennzeichnet werden, wird so orientiert, daß die sichtbare Fläche, in Pfeilrichtung betrachtet, rechts liegt. Insgesamt gibt es 4 verschiedene Kategorien und 16 verschiedene Typen von Schnittpunkten. Jeder Schnittpunkt in der Zeichnung muß in konsistenter Weise (Bild 9) einem dieser Typen zugeordnet – man sagt „etikettiert" – werden, ehe das Programm aus der Zeichnung das zugrundeliegende dreidimensionale Objekt rekonstruieren kann. Das von David A. Huffman vom Massachusetts Institute of Technology und dem verstorbenen Max B. Clowes von der Universität Sussex entwickelte Klassifikationsschema ist allerdings nur auf Objekte mit ebenen Oberflächen anwendbar, bei denen an keinem Schnittpunkt mehr als drei Ebenen aufeinandertreffen.

fangreiche und komplizierte Programme, und auch die besten Programme, die es auf diesem Sektor gibt, sind jeweils auf ein enges Sachgebiet oder einen kleinen Themenkreis beschränkt und erfassen im allgemeinen nur die naheliegendste Bedeutung. Dennoch ist dies ein wichtiges Forschungsfeld – nicht zuletzt, weil ein Programm, das Sprache verstünde, offensichtlich einen ganz wesentlichen Aspekt des menschlichen Denkens zu simulieren vermöchte.

Hinter den ersten, in den fünfziger Jahren unternommenen Versuchen, Computern ein Sprachverständnis zu vermitteln, stand die Absicht, die Übersetzung zu automatisieren. Die Idee war, anhand eines gespeicherten Lexikons für jedes Wort das Äquivalent in der Fremdsprache zu suchen und dann mittels einfacher Regeln Wortstellung und Satzbau anzugleichen. Alle diese Versuche endeten in der Sackgasse. Nachdem der Satz „Der Geist ist willig, aber das Fleisch ist

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (600 dpi)

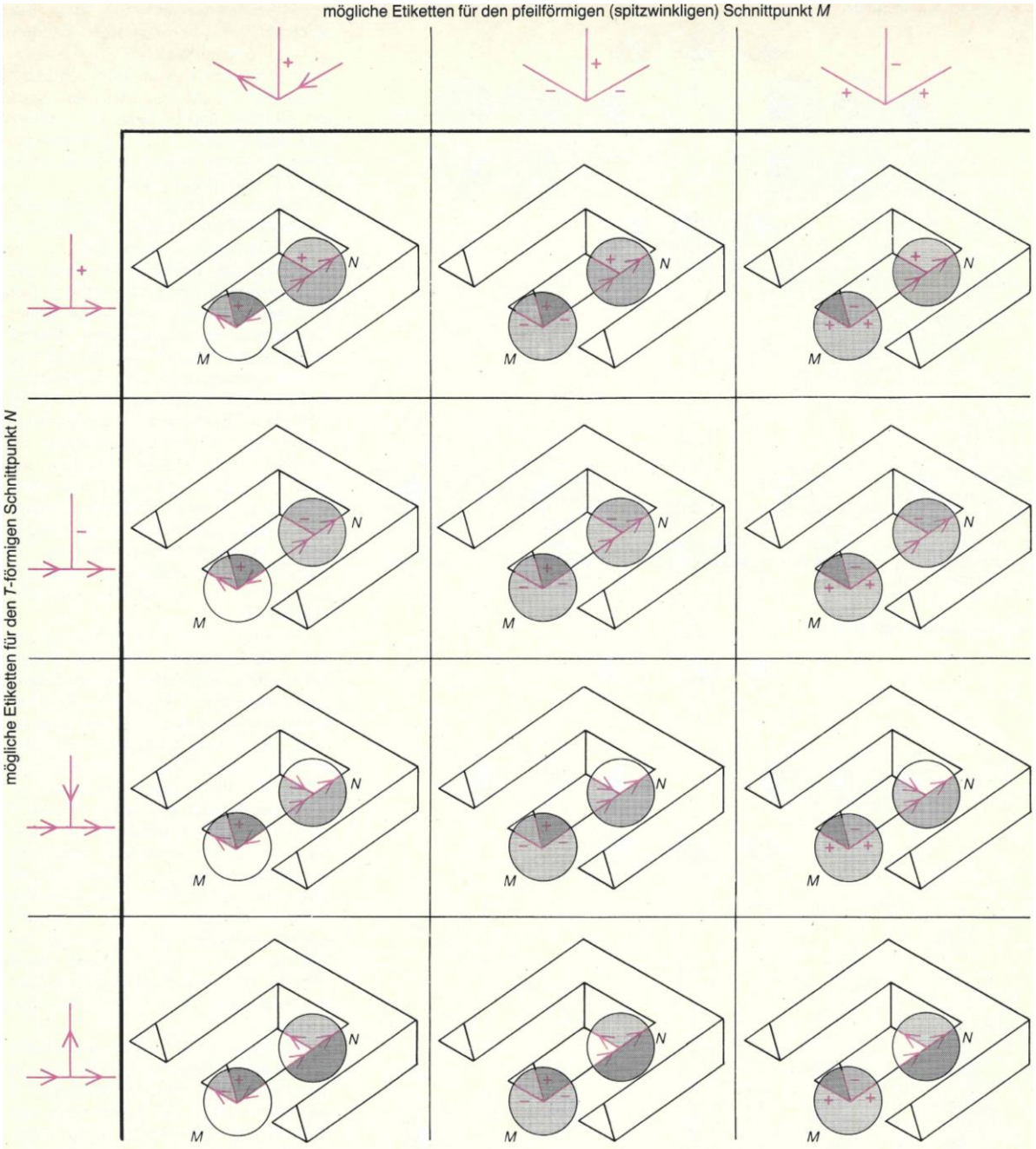

Bild 9: Daß die Teufelsgabel keine zweidimensionale Ansicht eines dreidimensionalen Objektes sein kann, zeigt sich, wenn man das in Bild 8 beschriebene Etikettierungsschema darauf anwendet. Für die Schnittpunkte *M* und *N* lassen sich keine konsistenten Etiketten finden: Die Kombination sämtlicher möglicher Schnittpunkttypen führt stets zu widersprüchlichen Bezeichnungen für die Verbindungslinie zwischen *M* und *N*.

schwach" ins Russische übertragen und dann wieder rückübersetzt worden war, hieß er, so geht die Sage: „Der Wodka ist stark, aber das Steak ist vergammelt". Für eine gute Übersetzung war ein inhaltliches Verständnis unerläßlich. Bis Mitte der sechziger Jahre hatte man diesen Ansatz daher ganz fallengelassen.

Einen anderen Ansatz verfolgte Joseph Weizenbaum vom Massachusetts Institute of Technology mit seinem 1966 geschriebenen Programm ELIZA. ELIZA umgeht jede wirkliche linguistische Analyse und bedient sich stattdessen eines festen, aber geschickt erdachten Antwortschemas, durch das es ein Sprachverständnis vortäuscht, das auf viele Leute erstaunlich echt wirkt. Das Antwortschema lehnt sich dabei an die Gesprächstechnik eines Psychiaters an: Das Programm sucht in den Aussagen des „Patienten" nach Worten oder Wortmustern, anhand derer es aus einem Vorrat gespeicherter Sätze oder Satzmuster pas-

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (600 dpi)

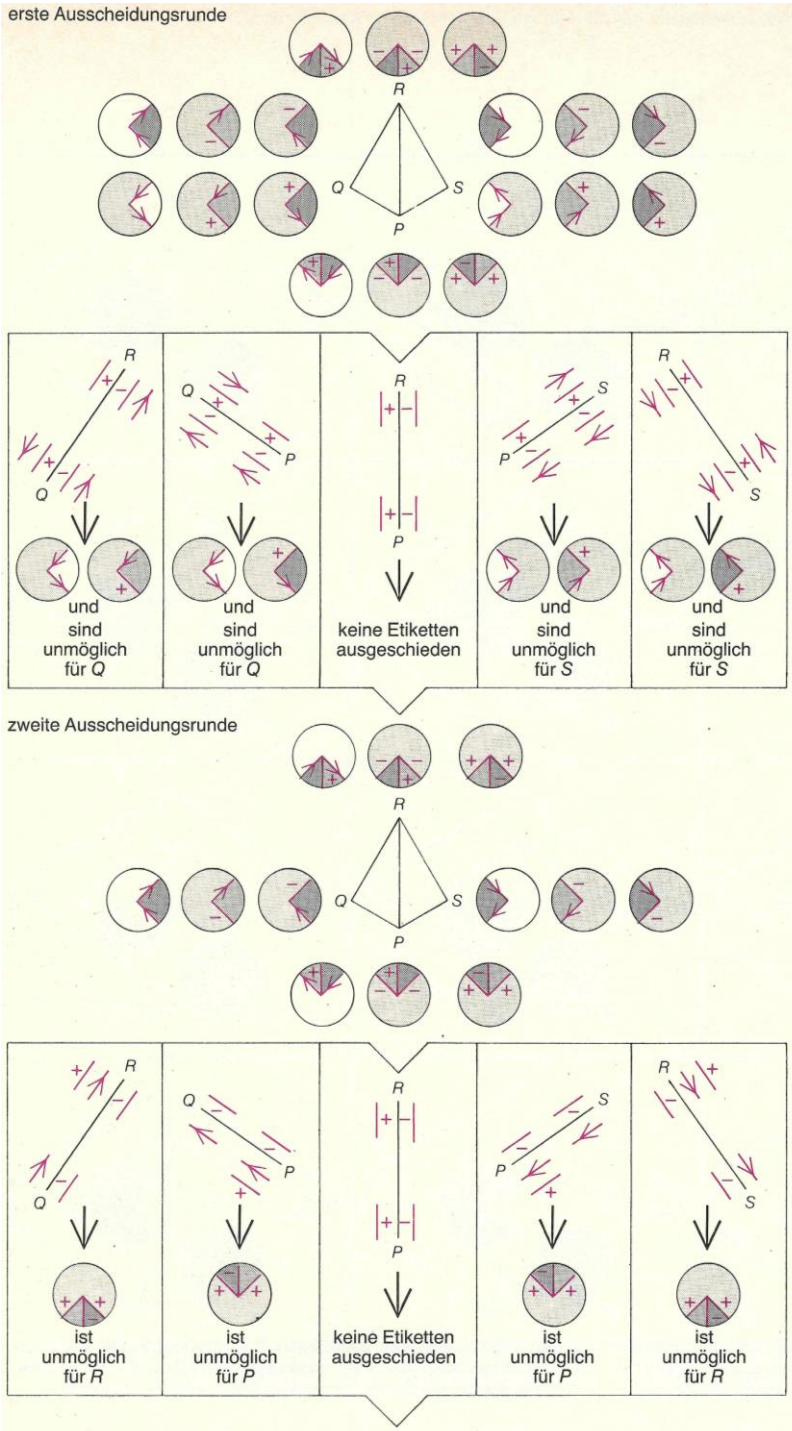

Bild 10: Mit der Methode der fortschreitenden Einschränkung lassen sich auch kompliziertere Bilder analysieren, da so eine parallele Etikettierung der Schnittpunkte möglich ist. Das Programm ordnet jedem Schnittpunkt alle zulässigen Etiketten zu und vergleicht sie mit sämtlichen zulässigen Bezeichnungen für jeden Nachbarschnittpunkt. Läßt sich für ein Etikett, das einem Schnittpunkt zugeordnet wurde, kein konsistentes Etikett für den Nachbarschnittpunkt finden (das zu einer widerspruchsfreien Bezeichnung der Verbindungslinie zwischen den Punkten führen würde), so wird dieses Etikett für den betreffenden Schnittpunkt ausgeschieden. Da sich zwei Nachbarschnittpunkte jeweils unabhängig auf die Verträglichkeit ihrer Etiketten prüfen lassen, kann diese Prüfung für alle Schnittpunktpaare gleichzeitig erfolgen. Mit jeder Ausscheidungsrunde wird die Zahl der möglichen Schnittpunktbezeichnungen weiter eingeschränkt. Die grauen Schattierungen sind für den Etikettenvergleich unmaßgeblich; sie dienen allein der optischen Verdeutlichung der Schnittpunkttypen.

sende Antworten auswählt. Fällt beispielsweise das Wort „Mutter", so kramt es einen Standardsatz hervor wie: „Erzählen Sie mehr von Ihrer Mutter". Tippt der Patient ein: „Ich fühle mich ein bißchen müde", kann ELIZA einen Teil dieses Satzes in ihre Antwort einbauen und etwa fragen: „Warum fühlen Sie sich ein bißchen müde?"

Obwohl das Programm viele der eingegebenen Worte ignoriert, braucht es eine umfangreiche Bibliothek für die Unmenge von Bedeutungsmustern, die es kennen muß, um mit der großen Zahl möglicher Aussagen seines menschlichen Gegenübers zurechtzukommen. Weizenbaum vertrat später die Ansicht, das Programm demonstriere, ein wie schlechtes Kriterium die Simulation menschlichen Verhaltens für die Beurteilung der Intelligenz eines Computerprogramms sei. Obwohl ELIZA, so sein Argument, nur über ein äußerst primitives und oberflächliches Sprachverständnis verfüge, ließen sich viele Leute von den lebensechten Antworten dazu verleiten, der Maschine ihre persönlichen Probleme mitzuteilen, als sei sie ein wirklicher Psychiater.

Um 1970 trat Roger C. Schank von der Yale-Universität mit Programmen hervor, die in natürlicher Sprache formulierte Sätze über menschliche Tätigkeiten verarbeiten konnten. Sie beruhen auf dem, was Schank Grundbausteine (Primitiva) begrifflicher Abhängigkeit nennt. Zu diesen Grundbausteinen gehören unter anderem MTRANS (die Menge aller Tätigkeiten, bei denen ein Transfer von Gedankeninhalten stattfindet, wie erzählen, hören, schreiben und lesen), ATRANS (die Menge sämtlicher Tätigkeiten, die mit einem Transfer von Eigentum verbunden sind, wie kaufen, verkaufen, geben und nehmen) sowie ATTEND (für Wahrnehmungstätigkeiten wie betrachten, zuhören, riechen und schmecken).

Man kann die Bedeutung eines Satzes nun in einem Diagramm darstellen, das auf solchen begrifflichen Abhängigkeiten basiert (Bild 11). Dabei zeigt sich, daß ähnlich aussehende Sätze unterschiedlicher Bedeutung ganz verschiedenartige Diagramme besitzen. Umgekehrt sind die Diagramme unterschiedlich formulierter, aber inhaltlich identischer Sätze gleich oder ähnlich. Schließlich bestimmen die begrifflichen Abhängigkeiten auch die „Erwartungen", die ein Programm hegt, nachdem es einen Satz teilweise analysiert hat. So erwartet es nach dem Satzfragment „Hans gab" sowohl einen Namen für den Empfänger als auch ein Wort für den übergebenen Gegenstand. Das Programm schafft im voraus freie Felder für die erwarteten Ausdrücke.

David L. Waltz: Künstliche Intelligenz

(Quelle: Spektrum der Wissenschaft – Dezember 1982)

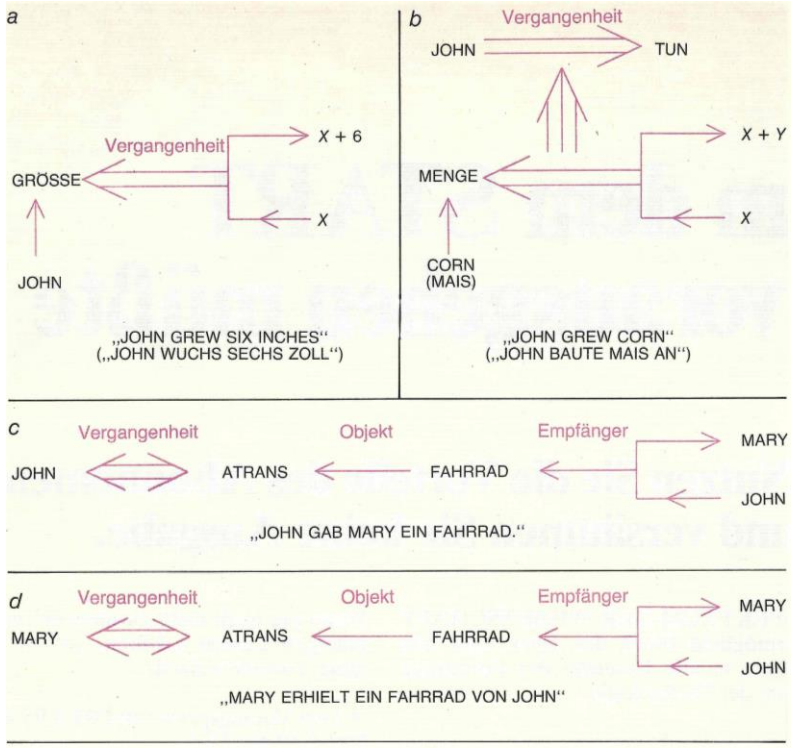

Bild 11: In Diagrammen begrifflicher Abhängigkeit wird die Bedeutungsstruktur eines Satzes computerintern so dargestellt, daß Sätze, die sich äußerlich ähneln, aber eine unterschiedliche Bedeutung haben, verschiedenartige Diagramme bekommen (*a*, *b*), während Sätze mit verwandtem Inhalt, aber äußerlich verschiedener Struktur zu gleichen oder sehr ähnlichen Diagrammen führen (*c*, *d*). Die hier abgebildeten Diagramme zeigen, daß der englische Satz „John grew six inches" für den Computer bedeutet: „Johns Größe stieg von irgendeinem Wert X auf den Wert $X + 6$ Zoll" (*a*). Unter dem Satz „John grew corn" dagegen versteht der Rechner: „John tat irgend etwas, wodurch die Menge an Mais von irgendeiner Zahl X auf die Zahl $X + Y$ anstieg" (*b*). Den Satz „John gab Mary ein Fahrrad" faßt das Programm auf als: „John übertrug den Besitz eines Fahrrades von sich selbst auf Mary" (*c*). (ATRANS steht für Verben, die die Übertragung von Eigentum beinhalten.) Der Satz „Mary erhielt ein Fahrrad von John" wird genauso dargestellt (*d*), nur daß nun Mary als Urheberin vermerkt ist. Urheber dieses Darstellungssystems ist Roger C. Schank von der Yale-Universität.

Rahmen und Drehbücher

Schanks Programm spiegelte einen allgemeinen Wandel in der Auffassung darüber wider, wie Sprachverständnis entsteht. Während früher die Meinung vorherrschte, daß Bedeutungsinhalte von unten nach oben konstruiert würden, der Weg also von der Definition der einzelnen Worte über ihre Bedeutung innerhalb von Satzteilen bis hinauf zu ihrer wahrscheinlichen Funktion im ganzen Satz führe, glaubt man heute, daß es sich vielmehr um einen abwärts gerichteten Prozeß handelt, bei dem Worte als Schlüsselreize für den Abruf von Erwartungen aus dem Gedächtnis und als Indizien für oder gegen die Berechtigung dieser Erwartungen dienen.

1974 äußerte Marvin L. Minsky vom Massachusetts Institute of Technology die Auffassung, nicht allein die Sprache, sondern alles Denken beruhe möglicherweise auf Prozessen, die von Erwartungs- und Wissensstrukturen – sogenannten Rahmen – abhingen. Wie ein Grundbaustein begrifflicher Abhängigkeit besteht auch ein Rahmen aus einem Kernelement und einer Reihe von Leerstellen. Jede Leerstelle entspricht einem Aspekt des vom Rahmen implizit definierten Begriffs. Nach Minsky besteht eine wichtige Funktion eines Rahmens darin, ein stereotypes Grundmuster zu verkörpern. Das Stereotyp wiederum ist ein intuitiv einleuchtendes Modell des Prozesses, durch den sich der Mensch nicht explizit gegebene Informationen über eine Situation hinzudenkt.

Eine Reihe von Sprachverständnis-Programmen greifen heute auf diese allgemeine Strategie zurück. So kann ein Programm, das Wendy G. Lehnert von der Yale-Universität geschrieben hat, Fragen zu Geschichten wie der folgenden beantworten.

„John nahm den Bus von New Haven nach New York. Unterwegs wurde ihm das Portemonnaie gestohlen. Er ging in eine Gaststätte und bestellte Spaghetti. John konnte nicht bezahlen und wusch daher Teller."

Obwohl nur gesagt wird, daß John Spaghetti bestellte, würden die meisten Menschen aus dem Fortgang der Geschichte schließen, daß er sie auch aß. Lehnerts Programm ist gleichfalls in der Lage, solche vernünftigen Schlüsse zu ziehen. Es tut das mit Hilfe eines speziellen, Drehbuch genannten Rahmens. Darin werden dem Computer jene Erwartungen, die die meisten Menschen im Hinterkopf haben, wenn von einem Gaststättenbesuch die Rede ist, explizit zugänglich gemacht.

Ein Programm namens FRUMP, das Gerald F. DeJong II an der Yale-Universität entwickelt hat, kann Meldungen der Nachrichtenagentur United Press International zusammenfassen. Es enthält Drehbücher über eine Vielzahl von Ereignissen wie Erdbeben, Verkehrsunfälle und Staatsbesuche. DeJong, der inzwischen zur Universität von Illinois in Urbana-Champaign übergewechselt ist, erweitert sein Programm zur Zeit dahingehend, daß es in der Lage ist, seine eigenen Drehbücher zu schreiben. Konfrontiert mit dem Textfragment „Marie wollte ein neues Radio; sie ging zur Bank", würde das Programm seine Drehbücher über das Erwerben neuer Dinge sowie über Bankbesuche zu Rate ziehen und nach Informationen durchforsten, die beides verbinden. So käme es dahinter, daß neue Radios Geld kosten und daß Leute zur Bank gehen, um Geld abzuheben. Daraus würde es „folgern", daß Marie zur Bank ging, um Geld für den Kauf des Radios zu holen. Ganz ähnlich könnte das Programm, ohne daß es etwas von Entführungen wüßte, einen Plan aufstellen, um herauszufinden, warum jemand auf die Idee kommen könnte, eine Person zu „entwenden". Nach Beendigung der Analyse könnte der Plan verallgemeinert und als Entführungsdrehbuch gespeichert werden.

Viele andere Aspekte des Sprachverständnisses wurden in jüngster Zeit ins Visier genommen. So hat man versucht, das Verständnis von Metaphern, von ungenannten Absichten und Überzeugungen des Erzählers sowie von den Gefühlen und Motiven der Figuren einer Geschichte mit Computerprogrammen zu simulieren. Andere Programme sollen mit widersprüchlichen Informationen zurechtkommen oder die Plausibilität von Aussagen abschätzen können.

Plausibilitätsbetrachtungen sind oft Vorbedingung für das Verständnis natürlicher Sprache. Bevor ein Programm Metaphern, Humor, Lügen oder Über-

treibungen verstehen kann, muß es fähig sein zu erkennen, daß die wörtliche Bedeutung eines Satzes in der gegebenen Situation möglicherweise nicht plausibel ist – so wenn das Prädikat nicht zum Objekt paßt wie in „Hans schluckte die Kritik" – oder daß ein Satz eine physikalisch unmögliche Aussage macht wie: „Bei dieser Nachricht sprang Marie drei Meter hoch". Ich persönlich halte die Beschäftigung mit Plausibilitätsbetrachtungen für einen ersten Schritt auf dem Weg zu einer Simulation des gesunden Menschenverstandes.

Gesunder Menschenverstand

Von den vielen Einwänden gegen die derzeitigen Bemühungen um die Künstliche Intelligenz trifft vielleicht der, es sei ihr bisher nicht gelungen, den sogenannten gesunden Menschenverstand zu modellieren, am ehesten den Kern der Sache. Eine Schwierigkeit bei der Simulation des gesunden Menschenverstandes besteht darin, daß ein entsprechendes Programm Wahrnehmung, Denken und Handeln kombinieren und in der Lage sein muß, alles drei auf einmal zu tun; denn intelligenter Umgang mit Begriffen schließt Wahrnehmung, Denken und Handeln gleichermaßen ein. Die besten derzeitigen Programme mit Sprachverständnis wissen im wahrsten Sinne des Wortes nicht, wovon sie reden; ihr einziges Bindeglied zur Welt ist die Sprache.

Ihre größten Erfolge konnte die Künstliche Intelligenz dort erringen, wo es gelang, abgeschlossene Bereiche zu definieren – etwa in der Klötzchenwelt von SHRDLU. Meiner Ansicht nach hat die Erfahrung jedoch gezeigt, daß sich solche Programme nicht einfach erweitern und verallgemeinern lassen. Die Fähigkeit, widersprüchliches und unvereinbares Wissen, reale Bilder oder die Sprache in ihrem ganzen Facettenreichtum zu handhaben, wird mit den bislang angewandten Baukasten-Methoden nicht zu erreichen sein.

Das heißt freilich nicht, daß Baukasten-Methoden beim Ergründen allgemeiner Prinzipien der Intelligenz nicht äußerst nützlich sein könnten. Gleichwohl bedarf es wesentlich besserer Modelle des menschlichen Bewußtseins, ehe Systeme, die auch nur ansatzweise über gesunden Menschenverstand verfügen, in greifbare Nähe rücken. Ich gehe davon aus, daß diese Aufgabe mich und viele andere noch lange in Atem halten wird.

David L. Waltz: Künstliche Intelligenz

NB: Aus Scientific American, Oktober 1982

(Quelle: Spektrum der Wissenschaft – Dezember 1982) (600 dpi)

First Published on January 26, 2017 – Size: 35.923 KB