Können Menschen mehr (Mathematik) als Computer?

Turing - Penrose - Searle

Achim Feldmeier, 23. Juni 2008

1931: Gödel beweist den Unvollständigkeitssatz.

1937: Turing beweist das Halteproblem.

1961: Lucas folgert: "Menschen können mehr Mathematik als Computer".

1994: Penrose nimmt dieses Argument auf in "Shadows of the Mind".

1994: Searle bespricht Penrose in "New York Review of Books".

Hier: Der Turingsche Beweis nach Ebbinghaus u.a. und der abstrakte Gödelsche Satz nach Smullyan.

Grundfrage: was ist ein Verfahren oder Algorithmus?

Church 1935: Jedes "Verfahren" lässt sich programmieren.

Ist bis heute unbewiesen. Weil der Begriff "Verfahren" unklar ist.

Turing entwickelt eine Assembler-Computersprache: Turingmaschine.

Beweist, dass es eine universelle Turingmaschine gibt, die jeden Algorithmus durchführt, also jede Turingmaschine "simuliert".

Ist heute empirisch klar: jeder Algorithmus ist programmierbar.

Ebbinghaus u.a. verzichten auf die komplizierte Turingsche Assemblersprache.

Wir setzen hier eine Programmiersprache wie Fortran voraus.

Ein Programm ist eine Abfolge von Zeilen in dieser Programmiersprache.

Übrigens: Computer heißen in der Mathematik nicht Rechenmaschinen, sondern Registermaschinen.

Sie rechnen nicht. Sie "speichern zwischen".

Input, Output, Stoppen

Seien A das Programmalphabet (Elementarzeichen) und A^* die daraus gebildeten Worte (Zeichenreihen).

Ein Wort $\zeta \in A^*$ dient auch als Input des Programms P.

Konkret: es wird vor oder zu Programmstart in das erste Register geschrieben.

Dass P bei Input von ζ irgendwann stoppt, wird geschrieben als

$$P : \zeta \to \text{stop}.$$

Wenn P nicht (d.h. nie) stoppt, schreibt man

$$P : \zeta \to \infty.$$

Dass P stoppt und vorher einen Output macht, schreibt man als (mit $\eta \in A^*$)

$$P : \zeta \to \eta.$$

Die Möglichkeit, dass P einen Output macht und nicht stoppt, wird nicht weiter formalisiert.

Dass ein Programm stoppt, werde durch eine STOP-Anweisung im Programm realisiert. (Diese muss es also in der Programmiersprache geben.)

Wir benutzen als Standardoutput $\square$: den Leerraum, blank, Vorrücken des Wagens.

Entscheidbarkeit

Def.: Sei $W \subset A^*$ eine Menge von Worten. Ein Programm P *entscheidet* W, wenn für alle $\zeta \in A^*$,

$$\begin{aligned} P : \zeta \to \square \qquad & \text{if } \zeta \in W, \\ P : \zeta \to \eta \qquad & \text{if } \zeta \notin W. \end{aligned}$$

Dabei wird in allem folgenden $\eta \neq \square$ vorausgesetzt.

Eine Menge W heißt *entscheidbar*, wenn es ein Programm P gibt, das sie entscheidet.

Kommentar: dies ist einer der Entdeckungen aus der Gödel-Tarski-Zeit: man kann nicht alle möglichen Mengen von Zahlen "ausdrücken" in einem strikten technischen Sinn: sie arithmetisch beschreiben: z.B. die Menge {2,4,6,8,...} als "teilbar durch 2 ohne Rest"; einen Algorithmus für sie angeben; sie "entscheiden".

Der Grund ist einleuchtend: die Menge aller Teilmengen von natürlichen Zahlen ist überabzählbar (Cantor). Die Menge aller Algorithmen, Programme aber ist abzählbar. Denn: man kann sie alphabetisch anordnen (siehe unten) und ihnen somit eine Abzählnummer zuordnen.

Beispiel: die Menge der geraden Zahlen ist entscheidbar. Ebenso die Menge der Primzahlen.

Gödelzahlen

Gödels geniale Idee. Durch Gödelzahlen werden die klassischen Paradoxien (Kreter, Barbier, Katalog) vermieden, ihre Selbstbezüglichkeit aber genutzt.

Weise Theoremen Zahlen zu. Theoreme über Theoreme werden dann zu Theoremen über Zahlen. Das sind aber einfach: Theoreme.

Wir ordnen jedem Programm (= Verfahren; entspricht Theorem) eine Zahl zu, indem

(i) wir dem Programm zunächst ein Wort ζ zuordnen: wir lassen alle Zeilenumbrüche weg

(ii) wir ζ eine Zahl zuordnen, nämlich seine Listennummer in einer alphabetischen Sortierung *aller* Worte in A^* — nicht nur der Programmworte.

Die Gödelzahlen sind also nur eine Teilmenge der natürlichen Zahlen.

Die Gödelzahl ist selbst wieder ein Wort aus A^*, z.B. 17.

Technische Anmerkung: A muss keine Zahlzeichen enthalten. Dann nimmt man einen bestimmten Buchstaben $\alpha \in A$ und schreibt ihn so oft hintereinander, wie die Gödelzahl lautet: $\alpha\alpha\alpha\alpha\alpha\alpha$....

Oder alternativ: 17 = siebzehn.

Die Menge aller Gödelzahlen ist entscheidbar

Wir schreiben die Gödelzahl eines Programms P als ζ_P. Sei

$$\Pi = \{\zeta_P \mid P \text{ ist ein Programm}\}.$$

Satz: Π ist entscheidbar. D.h. man kann ein Programm schreiben das entscheidet, ob irgendeine Zahl eine Gödelzahl ist, also ob an dieser Stelle im Wörterbuch aller Worte von A^* ein Programm steht.

Beweis: laut Ebbinghaus "offensichtlich": wir können die alphabetische Sortierung der Worte aus A beliebig weit treiben. Wenn also irgendeine Listenposition (Zahl) in diesem Wörterbuch gegeben ist, brauchen wir nur nachzusehen, was dort steht: ob das ein Programm ist, also eine erlaubte Kommandoabfolge der Programmiersprache. Ein Computer kann dies tun.

Eine unentscheidbare Menge

Satz: die Menge aller Gödelzahlen von Programmen, die bei Input ihrer eigenen Gödelzahl stoppen, ist unentscheidbar. Formal:

$$\Pi_{(\text{stop})} = \{\zeta_P \mid P \text{ ist Programm} \quad \wedge \quad P : \zeta_P \to \text{stop}\}$$

ist nicht entscheidbar.

Zum Beweis brauchen wir ein

Lemma: zu jedem Programm P, das eine Menge W *entscheidet*, also

$$\begin{aligned} P : \zeta \to \square \qquad & \text{if } \zeta \in W, \\ P : \zeta \to \eta \qquad & \text{if } \zeta \notin W, \end{aligned}$$

gibt es ein Programm P', das bei Input genau aus W *nicht stoppt*, also

$$\begin{aligned} P' : \zeta \to \infty \qquad & \text{if } \zeta \in W, \\ P' : \zeta \to \text{stop} \qquad & \text{if } \zeta \notin W. \end{aligned}$$

Beweis: ergänze P so, dass genau dann z.B. `TEST = .TRUE.` gesetzt wird, wenn $\square$ gedruckt wird. Ersetze dann die STOP-Zeile des Programms (die es laut Voraussetzung geben muss) durch eine Endlosschleife, z.B.

```
1 IF (TEST) GOTO 1
STOP
```

Die zweite Zeile ist nötig, damit das Progamm bei Nichtinput aus W stoppt. Das so modifizierte Programm ist das gesuchte P'.

Beweis des Satzes. Wir nehmen an, dass $P_{(\text{stop})}$ entscheidbar ist. Es gibt dann ein Programm P_0, so dass für alle P,

$$\begin{aligned} P_0 : \zeta_P \to \square \qquad & \text{if } P : \zeta_P \to \text{stop}, \\ P_0 : \zeta_P \to \eta \qquad & \text{if } P : \zeta_P \to \infty. \end{aligned}$$

Da "stoppen" (stop) und "nichtstoppen" (∞ = non stop) exklusiv sind, gibt es nach dem Lemma ein P_0', so dass

$$\begin{aligned} P_0' : \zeta_P \to \infty \qquad & \text{if } P : \zeta_P \to \text{stop}, \\ P_0' : \zeta_P \to \text{stop} \qquad & \text{if } P : \zeta_P \to \infty. \end{aligned}$$

Abgekürzt, indem wir die Exklusivität von ∞ und "stop" benutzen,

$$P_0' : \zeta_P \to \infty \quad \leftrightarrow \quad P : \zeta_P \to \neg\infty.$$

Jetzt kommt die berühmte *Diagonalisierung:* wähle aus allen P gerade P_0',

$$P_0' : \zeta_{P_0'} \to \infty \quad \leftrightarrow \quad P_0' : \zeta_{P_0'} \to \neg\infty,$$

also ein Widerspruch. Also ist die Voraussetzung falsch. Also ist $\Pi_{(\text{stop})}$ nicht entscheidbar.

Die Diagonalisierung entspricht der Selbstbezüglichkeit in den klassischen mengentheoretischen Paradoxa. Hier ist sie mathematisch einwandfrei.

Eine weitere unentscheidbare Menge

Dass ein Programm bei Input seiner eigenen Gödelzahl ein "Problem hat", mag noch dahingehen. Es gilt schlimmeres:

Satz:

$$\Pi_{\text{stop}} = \{\zeta_P \mid P \text{ ist Programm} \quad \wedge \quad P : \square \to \text{stop}\}$$

ist nicht entscheidbar. In Worten: es gibt keinen Algorithmus, der von jedem Algorithmus feststellen kann, ob er stoppt.

Wir beginnen den Beweis wieder mit einem

Lemma: Es gibt ein Verfahren, das jedem Programm P aus $\Pi_{(\text{stop})}$ ein Programm P^+ aus Π_{stop} zuordnet.

Das heißt, es gibt ein Verfahren $P \to P^+$, so dass

$$P^+ : \square \to \text{stop} \quad \leftrightarrow \quad P : \xi_P \to \text{stop}.$$

Beweis: Bestimme für ein gegebenes P zunächst dessen Gödelzahl ξ_P. Wie oben festgesetzt, nimmt P seinen Input aus Register 1. Ein geeignetes Programm P^+ ist dann das Programm P, vor dessen erster ausführbaren Zeile ein WRITE-Statement gesetzt wird, das den Wert ξ_P ins Register 1 schreibt. Eine bijektive Zuordnung $P \leftrightarrow P^+$ erhält man, indem man diesen Schreibbefehl $(\xi_P - 1)$ mal wiederholt.

Beweis des Satzes: Für vorgegebenes ξ_P (dies muss eine Gödelzahl sein, sonst ist P kein Programm und man ist nach Definition von Π_{stop} schon fertig) bestimmt man P aus dem Wörterbuch und ändert es wie angegeben zu P^+ ab. Wäre der Satz falsch, gäbe es ein Programm P_0, das entscheidet, ob $P^+ \in \Pi_{\text{stop}}$. Damit wäre aber auch entschieden, ob $P \in \Pi_{(\text{stop})}$. Dies ist ein Widerspruch zum vorigen Satz.

Literatur:

Ebbinghaus, Flum und Thomas, Einführung in die Mathematische Logik, diverse Auflagen und Verlage seit 1978, darunter Spektrum-Verlag und Springer. *Aus Vorlesungen an deutschen Unis entstanden. Daraus der Beweis des Halteproblems.*