KURT GÖDEL

und die

GRENZEN DES WISSENS

Mit Beiträgen von

Jana Bohnstengel
Lena Borgmann
Sören Dobberschütz
Jesco Humpola
Ulrich Krause
Ann–Kristin Petersen
Janina Ried

Herausgegeben von Prof. Dr. Dr. Ulrich Krause

UNIVERSITÄT BREMEN

Vorwort

Das vorliegende Bändchen ist entstanden im Rahmen meiner Veranstaltung „Vom mathematischen Denken“, Sommersemester 2006, im Studiengang Mathematik der Universität Bremen. Die Studentinnen Jana Bohnstengel, Lena Borgmann, Ann-Kristin Petersen, Janina Ried (alle 4. Semester) und die Studenten Sören Dobberschütz (6. Semester), Jesco Humpola (4. Semester) haben mit ihren Beiträgen jeweils einen Schein in der Rubrik „General Studies, Schlüsselqualifikation, BGW“ erworben, der im Diplomstudiengang Mathematik obligatorisch ist.

Vorausgegangen war ein Vortrag über „Gödel und die Grenzen des Wissens“, den ich anlässlich des 100. Geburtstags von Kurt Gödel am 28. April 2006 im Haus der Wissenschaft in Bremen gehalten habe. Aufgabe der Studierenden war es, zu Teilthemen des Vortrags eigene Nachforschungen anzustellen und jeweils ein schriftliches Referat auszuarbeiten. Bei dem letzten Beitrag des Bändchens handelt es sich um eine erweiterte Fassung des letzten Teils meines Vortrags. Absicht des Bändchens ist es, einem weiteren Leserkreis einige bahnbrechende Ideen näherzubringen, die teils von Gödel selbst stammen oder die sich teils auf ihn beziehen und zwar bis in die jüngste Zeit hinein.

Ich möchte mich an dieser Stelle herzlich für das Engagement der Studierenden und für ihre sehr schönen Beiträge bedanken. Zusätzlich danke ich Herrn Dobberschütz sehr für seine Bereitschaft, die einzelnen vorgelegten Texte zu einem Ganzen zusammenzufügen.

Bremen im August 2006 Ulrich Krause

Inhaltsverzeichnis

Kapitel 1

Semantische Paradoxien und Gödels Unvollständigkeitssatz

Jana Bohnstengel

1.1 Einführung

Der Artikel wird sich zum größten Teil mit logischen bzw. semantischen Paradoxien und Vollständigkeit beschäftigen. An den Beginn stelle ich zunächst eine kurze Biografie von Kurt Gödel. Danach werde ich eine Einführung in logische bzw. semantische Paradoxien anhand von einigen Beispielen geben. Im Anschluss werden einige Theorien zur Vermeidung von Paradoxien wie z.B. Tarskis Hierachie der Sprache oder Kripkes Theorie der semantischen Abgeschlossenheit diskutiert. Zum Schluss werde ich auf Gödels Unvollständigkeitssatz eingehen und erklären, was damit gemeint ist und welche Konsequenzen sich daraus ergeben. Ein Zusammenhang zwischen semantischen Paradoxien und Gödels Unvollständigkeitssatz besteht insofern als, Gödel selbst in seiner berühmten Arbeit „Über formal unentscheidbare Sätze der *Principia mathematica* und verwandeter Systeme I“ von 1931 schreibt:

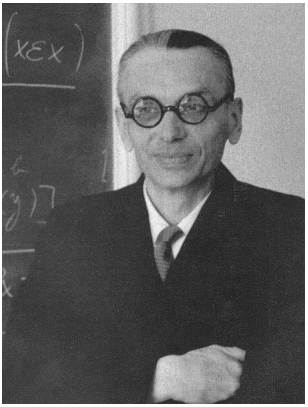

Abbildung 1.1: Kurt Gödel

> „Die Analogie dieses Schlusses mit der Antinomie Richard springt in die Augen; auch mit dem Lügner besteht eine nahe Verwandtschaft, . . . “[21][S. 148]

> „Wir haben also einen Satz vor uns, der seine eigene Unbeweisbarkeit behauptet.“[21][S. 150]

(vgl. Kapitel 4.2 Gödels Unvollständigkeitssatz)

1.2 Kurt Gödel

Kurt Gödel gilt als der bedeutendste mathematische Logiker des 20. Jahrhunderts.
Er hat maßgebliche Beiträge im Bereich der Prädikatenlogik sowie zum klassischen und intuitionistischen Aussagenkalkül geleistet. Nun zu seinem Werdegang, Kurt Friedrich Gödel wurde am 28.4.1906 im österreich-ungarischen Brünn geboren. Er war der zweite Sohn von Rudolf August

Gödel und Marianne Gödel, geborene Handschuh. Seine Eltern gehörte dort der deutschen Minderheit an. Sein Vater ist aus Wien nach Brünn gekommen, um dort Direktor einer Textilfabrik zu werden. In dieser Position ist er zu Wohlstand gekommen.[21]

Mit sechs oder sieben Jahren bekam Kurt rheumatisches Fieber und trotz vollständiger Genesung, glaubte er, dass er von der Krankheit eine dauerhafte Schädigung des Herzens bekommen hat. Dieses waren die frühesten Anzeichen von Kurt Gödels ständiger Sorge um seine Gesundheit.[21]

Von 1912 bis 1916 ging Kurt zur evangelischen Privat-Volks- und Bürgerschule. Ab 1916 ging er dann zum kaiserlich-königlichen Staatsrealgymnasium mit deutscher Unterrichtssprache in Brünn.[5]

Nach dem ersten Weltkrieg wurde Brünn 1918/1919 Teil der neu gegründeten Tschechoslowakischen Republik. Gödel sprach kaum Tschechisch und fühlte sich unwohl in dem neuen Staat.[34]

Nach seinem Schulabschluss 1924 zog er nach Wien, um dort Mathematik, Physik und Philosophie zu studieren und nahm 1929 die österreichische Staatsbürgerschaft an.

1926 bekam Gödel Kontakt zu dem Wiener akademischen Zirkel, der von Moritz Schlick ins Leben gerufen wurde. Durch diesen Kontakt wurde Gödels Interesse an den grundlagentheoretischen Fragen der Mathematik geweckt.[8]

Nach einem Umzug lernte Kurt Gödel 1928 Adele Nimbursky, geb. Porkert, die in der Nachbarschaft lebte, kennen. Es wird gesagt, dass sie im „Nachtfalter“, einem Wiener Nachtlokal, arbeitete. Gegen diese Beziehung schien Kurts Mutter Einwände gehabt zu haben und so heirateten sie erst 10 Jahre später.[8]

Am 6.7.1929 promovierte Gödel bei Hans Hahn mit seiner Arbeit „Über die Vollständigkeit des Logikkalküls“. Die Doktorwürde wurde ihm am 6. Februar 1930 verliehen.

Im Jahr 1931 veröffentlichte Gödel in seiner Habitilation seine zwei Unvollständigkeitssätze. Der erste Satz besagt, dass unter gewissen Bedin-

gungen eines Systems es in diesem System unentscheidbare Sätze[1] gibt. Der zweite Unvollständigkeitssatz besagt, dass die Konsistenz eines Systems im Rahmen eines reichhaltigen Systems nicht beweisbar ist (vgl. Kabitel 4.2 Gödels Unvollständigkeitssatz).[60]

Am 17.2.1933 wurde Gödel Privatdozent an der Universität Wien. Er erhielt dort einen Lehrauftrag und konnte dafür Gebühren bei den Studenten kassieren.[21]
In Wien bekam Gödel nie eine darüber hinausgehende Anstellung.

Auf Grund der bahnbrechenden Ergebnisse wurde Gödel an das Institute for Advanced Study in Princeton eingeladen. Dort war er erstmals im akademischen Jahr 1933/ 1934. In dieser Zeit traten seine psychischen Erkrankungen zum ersten Mal in Form von depressiven Stimmungen und hypochondrischen Zwangsvorstellungen auf. Im Frühjahr 1934 kehrte er nach Wien zurück. Zu dieser Zeit hatte er schon eine weitere Einladung nach Princeton erhalten.[21]

Doch auf Grund des Todes seines Mentors Hans Hahn und der Verschlechterung seiner Gesundheit verbrachte Gödel im Herbst 1934 eine Woche im Sanatorium in Purkersdorf bei Wien.

In den Jahren von 1935 bis 1938 hielt sich Gödel mehrfach in den USA auf. In dieser Zeit scheinen sich seine phobischen Neigungen verhärtet zu haben. So entwickelte er z.B. eine starke Angst vor Vergiftung. Deshalb verbrachte er einige Zeit in einer psychatrischen Klinik.[30]

Nach dem Anschluss von Österreich an das Dritte Reich wurde das Bildungssystem umgestellt und so unter anderem auch die Privatdozentur abgeschafft. Deshalb verlor Gödel seine Stellung und musste sich um eine adäquate akademische Stelle im deutschen Bildungssystem bemühen. Seine Anträge wurden aber nur schleppend bearbeitet. Als er zusätzlich noch von einem Trupp junger Nazis angegriffen wurde, weil sie ihn für einen jüdischen Intellektuellen hielten, und er zur Wehrmacht einberufen wurde, fasste er den Entschluss nach Amerika auszuwandern. Dank

[1] „Ein Satz ist unentscheidbar“ bedeutet, dass weder der Satz noch seine Negation in dem System herleitbar ist.

seiner amerikanischen Freunde (wie z.B. Albert Einstein, John von Neuman u.a.) konnte er nach einem scheinbar aussichtslosen Papierkrieg am 18.1.1940 mit seiner Frau Adele nach Amerika auswandern. Die Reise ging mit dem Zug von Wien über Berlin und Moskau nach Wladiwostock und von Yokohama mit dem Dampfschiff nach San Francisco, wo sie am 4.3.1940 eintrafen. Von dort reisten sie weiter nach Princeton.[30]

Dort arbeitete er am Institute for Advanced Study (IAS). 1946 wurde er ständiges Mitglied am IAS. Dort bekam er schließlich 1953 eine Professur. Ein Grund, weshalb er vorher als ungeeignet eingestuft wurde, war sein merkwürdiges Verhalten, das er auf Grund seiner phobischen Neigungen an den Tag legte.[34]
1948 wurde Gödel amerikanischer Staatsbürger.
1952 wurde Gödel von der Harvard-Universität die Ehrendoktorwürde verliehen und 1953 wurde er in die National Academy of Science aufgenommen.[8]

In den vierziger und fünfziger Jahren vereinsamte Gödel wegen seiner Gemütskrankheit zunehmend. Er hatte in Princeton nur wenige Freunde. Zu ihnen gehörte u.a. Albert Einstein und John von Neumann.

Seine letzten Lebensjahre verbrachte Kurt Gödel in seinem Haus in Princeton und in verschiedenen Sanatorien. Seine Krankheit hatte sich so weit verschlimmert, dass er sich nur noch durch die Führsorge seiner Frau halbwegs normal ernährte. Adele musste im Juni 1977 wegen einer Operation für einige Monate ins Krankenhaus. In dieser Zeit starb außerdem Morgenstern, ein Freund von Gödel.[8]

Adele wurde am 18.12.1977 aus dem Krankenhaus wieder entlassen und konnte Kurt Gödel zwei Tage später überzeugen, sich ins Krankenhaus Princeton-Hospital einliefern zu lassen. Dort starb er am 14.1.1978 an „durch Persöhnlichkeitsstörung verursachter Unterernährung und Auszehrung“.[30, 8]

1.3 Logische Paradoxien

1.3.1 Was ist eine Paradoxie?

Zu Beginn ein Beispiel einer semantischen Paradoxie aus Don Quijote von M. de Cervantes. Don Quijote lässt an seiner statt seinen Schildknappen Sancho Pansa als Statthalter der Insel Barataria zurück. Als Statthalter ist er auch Vorsitzender des Gerichts. Eines Tages wird ihm folgendes Problem vorgelegt. Es gibt ein Gesetz, das besagt, dass jeder, der eine Brücke von einem Kirchspiel ins andere überqueren möchte, vorher erklären muss, warum er die Brücke überqueren möchte. Diejenigen, die die Wahrheit sagen, dürfen die Brücke überqueren und die, die lügen, werden an einen Galgen gehängt.
Dieses Gesetz erfüllte lange Zeit seinen Nutzen, aber eines Tages kam ein Mann an die Brücke, der sagte, dass er nur aus dem Grund die Brücke überqueren möchte, um dort an dem Galgen zu sterben.[16]

Es gibt nun die beiden Möglichkeiten, dass der Mann die Brücke überqueren darf und freigelassen wird oder dass er gehängt wird. Wenn der Mann die Brücke überqueren darf, muss er nach dem Gesetz gehängt werden, weil er dann gelogen hat (da er als Grund angegeben hat, dass er die Brücke überqueren möchte, um gehängt zu werden). Wenn er allerdings gehängt wird, muss er die Brücke überqueren dürfen (und freigelassen werden), weil er den wahren Grund für die Überquerung der Brücke angegeben hat. Also muss er, wenn er die Brücke überqueren darf, gehängt werden und wenn er gehängt wird, muss er die Brücke überqueren dürfen (und freigelassen werden), weil er die Wahrheit gesagt hat. Er muss also sowohl gehängt, wie nicht gehängt werden.

Es stellt sich also heraus, dass in diesem Fall das Gesetz ausgesetzt werden muss. In diesem Fall können also die Prämissen verworfen werden.

Aber die Möglichkeit der Verwerfung der Prämissen besteht nicht immer.

Ein anderes Beispiel für eine semantische Paradoxie ist der sogenannte „Lügner“:

> „Dieser Satz ist falsch.“

Wenn wir nun annehmen, dass dieser Satz falsch ist, dann ist er wahr, weil das ja die Aussage des Satzes ist. Wenn wir aber davon ausgehen, dass der Satz wahr ist, dann ist er gleichzeitig falsch, weil dieses die Aussage des Satzes ist. Also muss der Lügner sowohl wahr wie falsch sein.

Nun werde ich zu einem Beispiel kommen, dass weniger paradox ist als der „Lügner“. Diese handelt von dem Kreter Epemenides. Es besagt, dass Epemenides in seiner Enttäuschung sagte, dass alle Kreter Lügner sind. Lügner ist in diesem Zusammenhang jemand, der immer lügt.

Wenn man nun annimmt, dass Epemenides die Wahrheit gesagt hat, muss er gelogen haben, weil er selbst Kreter ist und er nach seiner eigenen Aussage, somit selbst ein Lügner ist.

Es ist aber möglich, dass Epemenides gelogen hat. Daraus folgt nicht unbedingt, dass er gleichzeitig die Wahrheit gesagt hat, weil die Verneinung von „Alle Kreter sind Lügner.“ „Nicht alle Kreter sind Lügner.“ (bzw. „Es gibt mindestens einen Kreter, der kein Lügner ist.“) ist.

Ein weiteres Beispiel ist das sogenannte *Russellsche Paradoxon*. Davon werde ich zwei Varianten vorstellen, die ursprüngliche, mengentheoretische und eine anschauliche.

Die mengentheoretische Paradoxie zeigte Russell Gottlob Frege auf, der sich das Ziel gesetzt hatte, für die Arithmetik ein vollständiges Axiomsystem aufzubauen. Russell leitete das folgende Paradoxon aus Freges Axiomsystems ab.

> Sei A die Menge, die alle Mengen und nur diejenigen enthält,
> die sich nicht selbst enthalten.

Nun ist die Frage, ob A sich selbst enthält oder nicht. Wenn A sich nicht selbst enthält, dann muss sie sich selbst enthalten, weil sie alle Mengen enthält, die sich nicht selbst enthalten. Wenn sie sich aber selbst enthält, dann enthält A nicht nur die Mengen, die sich nicht selbst enthalten. Also darf sie sich nicht selbst enthalten. A muss sich also sowohl selbst enthalten als auch nicht selbst enthalten.[18]

Die anschauliche Version handelt von einem Barbier in Tombstone. Dieser Barbier rasiert alle diejenigen und nur die Männer, die sich nicht selbst rasieren. Dabei stellt sich die Frage, wer den Barbier rasiert, denn wenn er sich selbst rasiert, darf er sich nicht selbst rasieren, weil er nur diejenigen rasiert, die sich nicht selbst rasieren. Wenn er sich aber nicht selbst rasiert, muss er sich selbst rasieren, weil er alle Männer rasiert, die sich nicht selbst rasieren. Also muss er sich sowohl selbst rasieren, wie nicht selbst rasieren.

Eine weitere mengentheoretische Paradoxie ist von Cantor. Diese sagt aus, dass keine Menge größer sein kann, als die Menge von allen Mengen. Aber wenn man die Potenzmenge von dieser Mengen nimmt, ist dieser wiederum größer. Dementsprechend hat man eine Menge gefunden die größer ist als die Menge aller Mengen.[9]

Nun komme ich zu „*Richard's paradox*“, auf das Gödel in seiner Arbeit „Über formal unentscheidbare Sätze der *Principia mathematica* und verwandeter Systeme I“ von 1931 verweist (vgl. Einleitung).

> Let E be the class of decimals definable in a finite number of words, and let its members be ordered as the first, second, third ... etc. Now let N be the number such that if the nth figure in N is $m+1$, or 0 if $m = 9$. Then N differs from every member of E, and yet has been defined in a finite number of words.[9]

Anders ausgedrückt: Es gibt unendlich viele Zahlen. Von diesen sind einige von Menschen schon charakterisiert worden. Es gibt aber auch eine Menge von Zahlen, die noch nicht von Menschen charakterisiert wurden. Diese Menge hat ein kleinstes Element. Hierdurch ist diese Zahl aber charakterisiert.

Ein Paradoxon besteht also, wenn aus anscheinend annehmbaren Prämissen durch ein plausibles Argument eine unannehmbare Schlussfolgerung folgt.[16] Um mit Paradoxien umzugehen, gibt es drei Möglichkeiten. Man kann die Prämissen verwerfen, zeigen, dass die Überlegungen,

die zur Paradoxie geführt haben, falsch waren oder die Schlussfolgerung, dass es eine Paradoxie ist, akzeptieren. Für alle drei Herangehensweisen werde ich Beispiele geben.

1.3.2 Lösungsversuche für Paradoxien

Zu Beginn werde ich einige Lösungsmethoden kurz vorstellen.

Ein Ansatz besteht darin, Selbstbezüglichkeit zu verbieten. Aber dieses schränkt die möglichen Sätze zu weit ein und geht gleichzeitig nicht weit genug, da dann auch Sätze wie „Dieser Satz ist in Deutsch." nicht aussagbar wäre und z.B. Russells Paradoxie weiter bestehen würde.[16]
Diesen Ansatz kann man dann so erweitern, dass Selbstbezüglichkeit in Verbindung mit semantischen Begriffen verboten ist. Aber dann bestehen Antinomien wie „Der nächste Satz ist falsch. Der vorherige Satz ist wahr." weiterhin.
Weiter ist anzumerken, dass auch einige mathematische Argumente Selbstbezüglichkeit verwenden. Ein Beispiel für ein solches mathematisches Argument ist Gödels Beweis der Unvollständigkeit.[9] Weitere Beispiele sind Fixpunktprobleme.

Ein weiterer Ansatz von Bochvar ist die Verwerfung der Zweiwertigkeit, also dass es nicht nur wahr und falsch gibt, sondern auch sinnlos bzw. paradox [9]. Allerdings besteht in diesem Fall immer noch eine Paradoxie, nämlich der sogenannte „verstärkte Lügner",

> „Dieser Satz ist entweder falsch oder sinnlos."

Bei diesem Satz besteht eine Paradoxie, weil unter Annahme, dass der Satz wahr ist, folgt, dass er entweder falsch oder sinnlos ist. Unter der Annahme, dass der Satz falsch ist, folgt allerdings, dass er wahr ist, weil seine Aussage ist, dass er entweder falsch oder sinnlos ist. Wenn wir davon aussgehen, dass der Satz sinnlos ist, ist er wahr, weil dieses wiederum die Aussage des Satzes ist. Dementsprechend besteht eine Paradoxie.

Außerdem würde dieser Ansatz zu einer Veränderung der logischen Prinzipien führen [9].

Aber selbst wenn man die Zweiwertigkeit verwirft, kann man noch eine andere Antinomie finden. So führt z.B.

„Dieser Satz ist nicht wahr."

zu einem Widerspruch. Dabei wird nicht die Zweiwertigkeit[2] verwendet, sondern nur die Theorie des ausgeschlossene Dritten, das bedeutet, dass immer entweder die Aussage oder das Gegenteil wahr ist.

Im Folgenden werde ich noch drei weitere umfangreichere Theorien betrachten.

1.3.3 Tarskis Hierarchie der Sprache

Alfred Tarski

Alfred Tarski (ursprünglich Alfred Tajtelbaum) war ein polnischer Mathematiker und Logiker. Er wurde am 14. Januar 1902 in Warschau geboren. 1926 wurde Tarski Dozent an der Universität Warschau. 1939, als die Deutschen in Polen einmarschierten, befand sich Tarski gerade in Amerika und entschloss sich dann in den USA zu bleiben. 1942 bekam er schließlich eine Anstellung an der University of California, Berkeley. In Berkeley verstarb er am 26. Oktober 1983.[33]

Tarski schuf mit „Der Wahrheitsbegriff in den formalisierten Sprachen"(1933) die Grundlagen für die logische Semantik. Desweiteren veröffentlichte er Arbeiten zum Bereich der mathematischen Logik, Mengenlehre, Metamathematik, Modelltheorie, Algebra und Geometrie.[53]

Im folgenden wird Tarski's Hierarchie der Sprache betrachtet.

Hierarchie der Sprache

Tarskis Idee war eine Sprach-Hierarchie aufzubauen, um Paradoxien zu vermeiden. Dabei ist es nicht erlaubt Wahrheit und Falschheit von Sätzen

[2] Bei der Zweiwertigkeit wird davon ausgegangen, dass falsch das gleiche bedeutet wie nicht wahr. Es liegt eine zweiwertige Logik vor.

der gleichen Stufe auszudrücken, sondern man darf Wahrheit nur von Sätzen niedrigerer Stufe ausdrücken.

Auf der Stufe 0 ist der Wahrheitsbegriff noch nicht interpretiert. Es gibt also nur Aussagen die wahr oder falsch (bzw. semantische Begriffe allgemein) nicht enthalten. Dieses sind z.B. Sätze, wie „Schnee ist weiß.“. Auf der Stufe 1 wird nun die Wahrheit/ Falschheit von Sätzen der Stufe 0 weiter interpretiert. Hierzu gehören Sätze wie „Schnee ist weiß, ist wahr.“.[16] Auf der Stufe 2 werden nun Wahrheit/ Falschheit von Aussagen der Stufe 1 interpretiert. Diese Konstruktion wird fortgesetzt. Sätze dürfen dabei immer nur Wahrheit/ Falschheit von Sätzen niedrigerer Stufe ausdrücken. Der Lügner („Dieser Satz ist falsch.“) ist also nach Tarskis Hierarchie nicht ausdrückbar, er ist sinnlos. Der Satz kann nur in folgender Form auf der Stufe M gebildet werden: „Dieser Satz ist falsch-in-O.“, wobei M die Metasprache der Objektsprache O ist.[9]

Bei dieser Theorie ist ein Problem allerdings, dass sie *ad hoc* ist. Es wird nicht erklärt, warum Selbstbezüglichkeit zu Paradoxien führt. Es scheint zwar so zu sein, wie man in den obigen Beispielen gesehen hat.

Ein weiteres Problem ist, dass es nicht klar ist, warum ein Satz wie der Lügner, der sinnvoll gebildet ist, sinnlos sein soll.[16]

Außerdem treten bei dieser Theorie auch im Alltagsgebrauch einige Probleme auf. Wenn z.B. Aristoteles sagt, dass Platon die Wahrheit sagt und Platon sagt, dass Aristoteles lügt, dann muss in der Hierarchie der Sprache Aristoteles Aussage über Platons stehen. Aber gleichzeitig muss Platons Aussage über der des Aristoteles stehen. Die beiden Aussagen schaukeln sich also immer weiter hoch.[16]

In so einem Fall kann es also passieren, dass man glaubt, etwas Sinnvolles zu sagen, aber in Wirklichkeit sagt man nichts Sinnvolles, weil ein anderer vorher eine Aussage geäußert hat, die sich auf jene Person bezieht. Tarskis Hierarchie der Sprache ist also nur eine rein formale Lösung.

1.3.4 Saul Kripkes Theorie der semantisch abgeschlossenen Sprache

Saul Kripke

Saul Aaron Kripke ist ein 1940 in Bay Shore, New York, geborener amerikanischer Philosoph und Logiker. Er studierte in Harvard und war von 1968 bis 1976 Professor an der Rockefeller-Universität in New York. Ab 1976 war er in Princeton als Philosophieprofessor am CUNY Graduate Center.[36]

Er wurde 2001 mit dem renommierten Schock-Preis für Logik und Philosophie ausgezeichnet. Kripkes Arbeiten hatten großen Einfluss in dem Bereich der Logik und Sprachphilosophie.[36]

Semantische Abgeschlossenheit

Bei der Theorie der semantisch abgeschlossenen Sprache von Saul Kripke werden die Bedingungen für Wahrheit und Falschheit getrennt, das bedeutet, dass Falschheit nicht einfach „nicht Wahrheit“ bedeutet.

Desweiteren hat diese Theorie Ähnlichkeit mit der Hierachie der Sprache von Tarski. Bei Kripke wird allerdings die Sprache wirklich erweitert. Das bedeutet, dass die Metasprache einer jeden Objektsprache immer sowohl die Objektsprache als auch weitere Aussagen enthält.[9]

Bei dieser Konstruktion ist ein Paar $\langle S_1, S_2 \rangle$ gegeben. Zu Beginn sind S_1 und S_2 leer, das bedeutet, dass die Wahrheitsprädikate noch uninterpretiert sind.

Nun werden im ersten Schritt alle wahren Aussagen S_1 und alle falschen Aussagen S_2 zugeordnet. Diese Zuordnung ist partiell, weil Sätze wie „Schnee ist weiß, ist wahr.“ noch nicht zugeordnet sind, also noch kein Wahrheitsprädikat haben. Nun werden S_1 und S_2 erweitert. Dieses geht z.B. durch Kleenes starke Matrizen. Diese Konstruktion bedeutet, dass z.B. Aussagen wie „A oder B“ wahr sind, wenn A, B oder beide wahr

sind, falsch, wenn beide falsch und uninterpretiert sind, wenn beide noch keinen Wahrheitswert besitzen.[16] [3]

Der Erfolg dieser Konstruktion hängt von der Existenz eines Fixpunktes ab. Fixpunkt bedeutet, dass an diesem Punkt die Metasprache gleich der Objektsprache ist, also dass keine Erweiterung hinzukommt.

Dieser Fixpunkt existiert, weil wenn man die Erweiterung als Funktion Φ auffasst, ist diese Funktion monoton wachsend. Diese Funktion ist monoton wachsend, denn: $\langle S_1, S_2\rangle \leq \langle S_1', S_2'\rangle$ bedeutet $S_1 \subset S_1'$ und $S_2 \subset S_2'$. Für $\langle S_1, S_2\rangle \leq \langle S_1', S_2'\rangle$ folgt nun, $\Phi(\langle S_1, S_2\rangle) \leq \Phi(\langle S_1', S_2'\rangle)$, weil durch Φ die Menge der wahren und die Menge der falschen Aussagen erweitert wird. Da $S_i \subset S_i'$ gilt, gilt auch, dass die Erweiterung von S_i in der Erweiterung von S_i' enthalten ist, weil alle sich ergebenen wahren/ falschen Aussagen, die sich aus S_i ergeben, sich auch aus S_i' ergeben. Diese Funktion ist also monoton wachsend. Außerdem wird angenommen, dass diese Funktion stetig ist. Φ ist also eine monoton wachsende stetige Funktion und hat als solche einen minimalen Fixpunkt. Für diesen gilt: $\Phi\,\langle S_1, S_2\rangle = \langle S_1, S_2\rangle$ (siehe [16])

An den Fixpunkten haben alle paradoxon Sätze keinen Wahrheitswert. Aber nicht alle Sätze ohne Wahrheitswert sind Paradox. So kann dem Satz „Dieser Satz ist wahr.“ ein konsistenter Wahrheitswert zugeordnet werden, obwohl er am Fixpunkt keinen Wahrheitswert hat.[9]

Diese Theorie hat also Ähnlichkeit mit der dreiwertigen Logik von Bochvar, weil in diesem Fall ebenfalls Sätze nicht entweder wahr oder falsch sind, sondern sie können auch uninterpretiert sein. Paradoxe Sätze haben kein Wahrheitsprädikat.[9]

Sätze wie „der Lügner“ haben also keinen Wahrheitswert. Sie heißen

[3] Andere Erweiterungen sind z.B. „A ist wahr“. Diese ist wahr, wenn A wahr ist, falsch, wenn A falsch und unbestimmt, wenn A unbestimmt ist.
nicht-A: wahr wenn A falsch; falsch, wenn A wahr und unbestimmt, wenn A unbestimmt
A und B: wahr, wenn A und B wahr sind; falsch, wenn A oder B falsch; unbestimmt, wenn A oder B unbestimmt.
$(\exists x)Fx$: wahr, wenn Fx für ein x wahr ist, falsch, wenn Fx für alle x falsch ist, sonst unbestimmt.

unfundiert.
Bei dieser Theorie haben paradoxe Sätze in der Objektsprache wieder keinen Wahrheitswert, sondern einen Wahrheitswert kann ihnen nur in der Metasprache zugeordnet werden.

1.3.5 Akzeptanz von Widersprüchen

Es gibt auch noch die Möglichkeit Paradoxien zu akzeptieren. Also zu akzeptieren, dass Aussagen sowohl wahr wie falsch sein können. Diese Theorie ist nur vernünftig, wenn man nicht zu viele Antinomien hinnehmen muss.

Nun werde ich die Folgen der Hinnahme von Paradoxien untersuchen. Nehmen wir an, es gilt sowohl A wie nicht-A, dann folgt, dass auch A oder B (A $\vee$ B, wobei B eine beliebige Aussage ist (also kann es auch eine falsche sein)) wahr ist. Dann folgt wiederum aus nicht-A, dass B wahr sein muss.
Diese Schlussform nennt sich *Ex falso quodlibet* (übersetzt: aus Falschem folgt Beliebiges). Durch diese Schlussform bekommt man also, dass jede beliebige Aussage wahr ist. Diese ist keine akzeptable Schlussfolgerung, also muss die Schlussform *Ex falso quodlibet* verworfen werden.[16]

Doch auch dann kann man durch einen Satz wie „Wenn dieser Bedingungssatz wahr ist, dann ist B wahr.“, wobei B wieder für eine beliebige Aussage steht, wieder bekommen, dass jeder beliebige Satz wahr ist.
Wenn der Bedingungssatz wahr ist, dann folgt, dass B wahr ist, weil aus einem wahren Wenn-Satz der Dann-Satz folgt. [4] Dieses ist aber gerade das, was der Bedingunssatz sagt. Also ist gezeigt, dass der Bedingungssatz wahr ist. Daraus folgt aber auch ein wahrer dann-Satz.[16]
Durch diese Schlussform können also ebenfalls alle beliebigen Aussagen hergeleitet werden. Deshalb muss die obige Schlussform verworfen werden. Ein Grund für die Verwerfung der Schlussform ist die zweifache Verwendung des Bedingungssatzes. Dieser Satz wird einmal als konditionale

[4] Diese Schlussform heißt *modus ponendo ponens*(Wenn A, und wenn A, dann B, dann B).

Prämisse des *modus ponendo ponens* und zum anderen als Wenn-Satz dieser konditionalen Prämisse verwendet, dabei wurde der Bedingungssatz nur einmal aufgeschrieben. Wenn man diese Schlussform explizit hinschreibt, ergibt sich:

> Wenn dieser Bedingungssatz wahr ist, dann, wenn dieser Bedingungssatz wahr ist, ist B wahr.

Nun ist der Schluss von B nicht mehr möglich, weil nun zwei verschiedene Bedingungssätze dort stehen.
Es wird also verboten, zwei Anwendungen einer Annahme durch eine zu ersetzen. Dieses wird als *Kontraktion* bezeichnet.[16]

Zur Herleitung der Paradoxien wurde die *reductio ad absurdum*[5] verwendet. Diese Schlussform ist eng mit der *Kontraktion* verbunden. Eine grundlegende Form der *reductio* ist die *consequentia mirabilis*[6]. Zu „nicht-A“ ist „wenn A, dann B“ (B beliebige Aussage) äquivalent. Also ergibt sich für die *consequentia mirabilis* „Wenn A, dann, wenn A, dann B; also, wenn A, dann B“. Dieses ist wiederum ein Beispiel für eine *Kontraktion*.
Also folgt aus der Verwerfung der *Kontraktion*, dass keine Paradoxien mehr existieren, weil so auch die *reductio ad absurdum* bzw. die *consequentia mirabilis* verworfen werden.[16]

Bei Akzeptierung von Widersprüchen folgt also entweder, dass jede beliebige Aussage wahr ist, oder dass keine Paradoxien mehr bestehen und einige Schlussformen der Logik verworfen werden müssen. Diese Folgen sind nicht akzeptable und somit ist diese Theorie der Akzeptanz von Widersprüchen fragwürdig.

[5] Man beginnt mit der Annahme A und kann einen Widerspruch folgern. Also schließt man, dass nicht-A gilt.

[6] Wenn A, dann nicht-A; also nicht-A

1.3.6 Russells Typentheorie

Russell vertrat die Ansicht, dass es einen gemeinsamen Grund für semantische und mengentheoretische Paradoxien gibt. Dieser lag für ihn im *vicious circle principle* (Teufelskreisprinzip).
Dieses ist:

> „Whatever involves all of a collection must not be one of the collection“[9] [S. 141]

oder anders ausgedrückt

> „If, provided a certain collection had a total, it would have members only definable in terms of that total, then said collection has no total.“[9] [S. 141]

Als formale Lösung der Paradoxien hat Russell die *Typentheorie* entwickelt. Zum einen gibt es die einfache Typentheorie, die zur Vermeidung von mengentheoretischen Paradoxien ist, und durch die erweiterte Typentheorie werden semantische Paradoxien vermieden.

In der einfache Typentheorie gibt es eine Hierarchie; Individuen (Typ 0), Mengen von Individuen (Typ 1), Mengen von Mengen von Individuen (Typ 2) usw.. Dabei werden die Elementen mit einem Index beschrieben. x_0 bedeutet also Element von Typ 0, x_1 Element von Typ 1, etc.. Eine Aussage $x \in y$ ist nur dann richtig geformt, wenn der Index von x um eins kleiner ist als der von y. Also ist speziell $x_n \in x_n$ nicht richtig geformt und somit ist es nicht ausdrückbar, Mitglied von sich selbst zu sein.[40]

In der erweiterte Typentheorie wird dieses Schema auf Sprachen übertragen. Dabei wird wie in Tarskis Hierarchie zwischen der Objektsprache und der Metasprache unterschieden. Bestimmte Aussagen sind also nur in der Metasprache ausdrückbar.

Die Paradoxien sind also in der Typentheorie nicht ausdrückbar. Sie sind nicht richtig geformt und somit sinnlos.

1.3.7 Zusammenfassung der Theorien

In den vorher betrachteten Theorien konnte man erkennen, dass alle Theorien zu Problemen führten oder nur rein formale Lösungen sind. Es scheint also nicht möglich zu sein, eine Theorie zu ermitteln, durch die die Paradoxien vermieden werden können und gleichzeitig keine Probleme bei der Verwendung der Sprache auftreten.
Deshalb stellt sich die Frage, ob die Paradoxien vielleicht aus der unpräzisen Sprache entstehen. Vielleicht ist es nötig unsere semantischen Begriffe genauer zu präzidieren.

Der Zusammenhang dieser semantischen Paradoxien mit Gödels Unvollständigkeitssatz besteht darin, dass Gödel als anschauliche Beweisidee des Unvollständikeitssatzes eine Aussage wie z.B. „Diese Aussage ist nicht beweisbar.“ in einem System gebildet hat und dadurch gezeigt hat, dass nicht jeder Satz beweisbar ist, weil wenn „Diese Aussage ist nicht beweisbar.“ in dem System beweisbar wäre, wäre es wahr und somit nicht beweisbar, weil das die Aussage ist. Wenn er dagegen in dem System nicht beweisbar wäre, wäre er wahr, obwohl er nicht beweisbar ist. Somit gibt es in einem System Sätze, die nicht beweisbar und nicht widerlegbar sind.

Das bedeutet also, dass durch den Unvollständigkeitssatz die alten philosophischen Paradoxien in ein mathematisches System übersetzt werden.

Auf diesen Unvollständigkeitssatz werde ich nun noch weiter eingehen.

1.4 Unvollständigkeit

1.4.1 Hinführung - geschichtlicher Hintergrund

Die Geschichte beginnt mit David Hilbert, der als der bedeutendste Mathematiker seiner Zeit gilt. Er stellte das sogenannte Hilbertsche Programm 1921 auf. In diesem vertrat er die Ansicht, dass es zum einen ein Verfahren geben müsste, durch das für jeden logischen Ausdruck durch

endlich viele Schritte entschieden werden kann, ob dieser Ausdruck wahr oder falsch ist. Dieses wird als Entscheidungsproblem bezeichnet.[39]

Weiter enthält dieses Programm die Aufforderung, die Widerspruchsfreiheit für das arithmetische System nachzuweisen.[60]

Außerdem enthält dieses Programm, die Aufforderung für ein gegebenes System von Axiomen die Vollständigkeit zu zeigen. Vollständigkeit bedeutet, dass jeder wahre Satz eines Systems in dem System ableitbar ist.

Gödel beschäftigte sich mit dem letzten Punkt, also mit der Vollständigkeit.

1929 promovierte Gödel mit der Arbeit „Über die Vollständigkeit des Logikkalküls“. Er scheint auf das Vollständigkeitsproblem der Logik durch Carnap, einem Schüler von Frege, aufmerksam geworden zu sein. Mit diesem Vollständigkeitssatz schien Gödel zunächst Hilberts Programm zu unterstützen. Bei Vollständigkeit handelt es sich darum, dass ein Satz φ aus einem Axiomsystem Γ durch endlich viele Schritte hergeleitet werden kann.[30]

Gödels Vollständigkeitssatz besagt nun,

> „..., dass unsere Logikkalküle keiner Erweiterung bedürfen und fähig sind: wenn φ in jeder Struktur gilt, die alle Axiome aus Γ erfüllt, dann gibt es einen Beweis von φ aus Γ.“[30]

Aus diesem Satz folgt wiederum der Kompakteitssatz, der Folgendes besagt:

> „... wenn jede endliche Teilmenge eines Axiomsystems Γ konsistent ist (d.h. kein Beweis von $0 = 1$ existiert), dann ist Γ selbst erfüllbar.“[30]

Doch Gödel brachte mit seinen Unvollständigkeitssätzen, die er 1931 veröffentlichte, Hilberts Programm, bzw. seine Forderung nach der Widerspruchsfreiheit des Axiomsystems (z.B. der Zahlentheorie), zu Fall.

1.4.2 Gödels Unvollständigkeitssatz

Gödel stellte seinen ersten Unvollständigkeitssatz im September 1931 im Rahmen einer Tagung der Deutschen Mathematiker Vereinigung (DMV) in Bad Elster vor. Bei dieser Tagung konnten viele Mathematiker wie z.B. Ernst Zermelo den Beweisen nicht folgen und viele erkannten auch nicht, was dieser Satz für die Mathematik bedeutet.[8]

Hilbert hatte spätestens Anfang 1931 von Gödels Unvollständigkeitsatz erfahren und erkannte gleich die Bedeutung für sein Programm.

Sein erster Unvollständigkeitssatz besagt, dass für ein korrektes und reichhaltiges System (d.h., dass alle ableitbaren Aussagen wahr sind und dass die elementare Zahlentheorie und Logik im System formulierbar sind) folgt, dass es unvollständig ist, also dass es Aussagen gibt, die unentscheidbar sind (d.h. weder die Aussage noch ihre Negation ist im System ableitbar). Es gibt also Sätze in einem System, so dass weder der Satz noch seine Negation in dem System beweisbar ist.[60]

Mit seinem zweiten Unvollständigkeitssatz liefert er ein Beispiel für einen solchen Satz. Dieser Unvollständigkeitssatz besagt nämlich, dass für ein konsistentes und reichhaltiges System (d.h., dass keine Aussage zusammen mit ihrer Negation ableitbar ist und dass die elementare Zahlentheorie und Logik im System formulierbar sind) folgt, dass die Konsistenz (d.h. alle ableitbaren Aussagen sind wahr) des Systems nicht im System ableitbar ist.[60]

Als Beweisidee des ersten Unvollständigkeitsatzes erzeugt Gödel durch die sogenannte „Gödelisierung [7]“ einen Satz in der Zahlentheorie, der „Dieser Satz ist (in diesem System) nicht beweisbar“ aussagt. Dieser Satz kann nicht in dem System beweisbar sein. Denn wenn er beweisbar und damit wahr wäre, wäre er nicht beweisbar. Wenn er aber falsch wäre, wäre er beweisbar und dementsprechend wahr.
Das bedeutet also, dass in einem System beweisbar und wahr nicht äquivalent sind.

[7] Gödelisierung bedeutet, dass Ausdrücke der formalen Sprache mit natürlichen Zahlen identifiziert werden.

Gödel schrieb dazu:

> The occasion for comparing truth and demonstrability was an attempt to give a relative model-theoretic consistency proof of analysis in arithmetic. Th[at] leads almost by necessity to such a comparison. For an arithmetical model of analysis is nothing else but an arithmetical $\in$-relation satisfying the comprehension axiom:
>
> $$(\exists n)(x)[x \in n \equiv \Phi(x)].$$
>
> Now, if in the latter „$\Phi(x)$“ is replaced by „$\Phi(x)$ is provable,“ such an $\in$-relation can easily be defined. Hence, if truth were equivalent to provability, we would have reached our goal. However (and this is the decisive point) it follows from the correct solution of the semantic paradoxes that „truth“ of the propositions of a language *cannot be expressed* in the same language, while provability (being an arithmetical relation) *can*. Hence true $\not\equiv$ provable.[5]

Es kann allerdings ein anderes System geben, in dem dieser Satz bewiesen werden kann. In diesem System gibt es aber auch wieder unbeweisbare Sätze.

Eine weitere Konsequenz des ersten Unvollständigkeitssatzes ist, dass wenn φ ein unentscheidbarer Satz in dem Axiomsystem Γ ist, dann bildet sowohl Γ mit dem Axiom φ ein widerspruchsfreies System, als auch Γ mit nicht-φ ein widerspruchsfreies System.[8]

Gödels Satz hat auch noch weitere Folgerungen. Eine Folgerung ist z.B., dass das Wahrheitsprädikat formal nicht definierbar ist, weil sonst „Dieser Satz ist nicht wahr.“ formalsprachlich darstellbar wäre.[30]

Eine mathematische Folgerung ist, dass es niemals ein System geben kann, in dem alle mathematischen Sätze beweisbar sind.

Eine weitere Folge von Gödels Unvollständigkeitssätzen ist, dass Hilberts zweites Problem, nicht lösbar ist.[27]

Hao Wang, ein Vertrauter Gödels, schlug für die Folgen des Unvollständigkeitssatzes folgende umgangssprachliche Fassung vor:[8]

- Die Mathematik ist unerschöpflich: Sie kann nicht vervollständigt werden.
- Jede widerspruchsfreie formale Theorie der Mathematik enthält notwendigerweise unentscheidbare Aussagen.
- Kein Computer(-programm) kann alle und nur die wahren Aussagen der Mathematik beweisen.
- Kein formales System der Mathematik (das wenigstens die Theorie der natürlichen Zahlen umfasst, also ein reichhaltiges System) kann sowohl widerspruchsfrei als auch vollständig sein.

Wang schrieb über die Bedeutung des Unvollständigkeitsaxiom „außerhalb der Mathematik“:

> „Die Tatsache der algorithmischen Unerschöpflichkeit zeigt, Gödel zufolge, dass entweder der menschliche Geist allen Computern überlegen ist, oder dass die Mathematik nicht vom menschlichen Geist geschaffen ist; oder aber sie zeigt beides zugleich. Es ist daher einleuchtend, dass das Theorem sowohl für die Philosophie des Geistes als auch für die Philosophie der Mathematik von Bedeutung ist.“[8]

22 Semantische Paradoxien und Gödels Unvollständigkeitssatz

Kapitel 2

Berechenbarkeit und Intelligenz im Sinne von Alan Turing

Lena Borgmann

2.1 Einleitung

Dieser Abschnitt handelt von den Werken Turings, die im Kontext von Gödels Arbeiten entstanden. Turing beschäftigte sich hierbei mit der Konstruktion von theoretischen Maschinen, die dazu geeignet sind, berechenbare von nicht berechenbaren Funktionen zu unterscheiden.

Außerdem geht es um die Fragestellung, in wieweit Maschinen denken können. Turing, zählt sich zu den Befürwortern und formuliert diese Frage mit Hilfe des Turing-Tests zunächst zur besseren Beantwortung um. In Zusammenhang mit dieser Frage werden eine Reihe von Gegenargumenten genannt.

2.2 Turings Leben

Alan Turing wurde 1912 in London geboren. Sein Vater, britischer Staatsdiener in Indien, wollte Alan Turing und seine Mutter den Gefahren Indiens nicht aussetzen. Während sie in London lebten, pendelte sein Vater.[32]

Abbildung 2.1: Alan Turing [56]

Während seiner Schulzeit wurde Turings Begabung von einigen Lehrern erkannt und gefördert. In dem Internat in Dorset, auf das er später ging, wurde sein Drang zur Naturwissenschaft allerdings nicht sehr unterstützt, da das Internat den Geisteswissenschaften näherstand. Seine Noten in den Fächern der Geisteswissenschaften wurden schlechter, so dass er das College wechselte und für die Jahre 1931 bis 1934 auf ein College in Cambridge ging. Während seiner Schulzeit beschäftigte er sich unter anderem sehr intensiv mit den Arbeiten von Einstein.[32]

Im Jahr 1936 erschien eine seiner wichtigsten Arbeiten „On Computable Numbers, with an Application to the Entscheidungsproblem“. Diese Arbeit baut auf die von Kurt Gödel 1931 erarbeiteten Theorien auf und setzt sie in Zusammenhang mit der Turing-Maschine.

Für zwei Jahre ging Turing nach Princeton an die Universität, um dort seinen Doktortitel zu erwerben. Anschließend kehrte er nach Cambridge

zurück und diskutierte mit Ludwig Wittgenstein darüber, ob die Mathematik überbewertet sein oder ob sie absolute Wahrheit zutage bringen könne.

In den folgenden Jahren des zweiten Weltkriegs arbeitete Turing daran, die deutschen Verschlüsselungen Enigma und Fish zu dechiffrieren. Dazu wurde der erste digitale, programmierbare, elektronische Röhren-Computer Colossus entwickelt, der mit Hilfe vieler anderer Wissenschaftler 1943 die Fish-Verschlüsselung entzifferte. Ebenso trug er entscheidend dazu bei, den Enigma-Code zu dechiffrieren, was den Weltkrieg um zwei Jahre verkürzte.[26]

Nach dem Krieg arbeitete Turing in Teddington, Großbritannien, an der Entwicklung weiterer Computer und programmierte erste Software. 1948 ging er dann nach Manchester, um dort zu lehren, und wurde stellvertretender Direktor der Computerabteilung der Universität. Er arbeitete unter anderem an einem Schachprogramm, das allerdings auf Grund zu leistungschwacher Rechner nicht getestet werden konnte. Bei einem von ihm persönlich durchgeführten Test verlor er gegen seinen Gegner. Ein Zug dauerte dabei rund 90 Minuten.[32]

Er arbeitete in den letzten Jahren an mathematischen Problemen der Biologie, bevor er wegen seiner Homosexualität angeklagt wurde. Nach seiner Verurteilung kam er in eine psychiatrische Anstalt, in der er sich 1954 das Leben nahm, indem er sich mit Zyanid vergiftete.[42]

2.3 Die Turing-Maschine

Die Turing-Maschine ist ein mathematisches Modell, das Turing 1936 entwickelte. Sie dient dazu, berechenbare von nichtberechenbaren Funktionen zu unterscheiden. In seiner Arbeit „On Computable Numbers, with an Application to the Entscheidungsproblem“, die auf Gödels Unvollständigkeitssatz und auf Hilberts Entscheidungsproblem Bezug nimmt, wird die Maschine zum ersten Mal erwähnt. Sie ist konstruiert, um zu entscheiden, ob eine Berechnung endlich ist, d.h. ob eine Funk-

tion berechenbar ist oder nicht. Diese Entscheidung wird ersetzt durch die Frage, ob die Turing-Maschine bei ihren Berechnungen anhält.

Diese Maschine wird bei allen von Menschen berechenbaren mathematischen Funktionen halten, jedoch kann keine allgemeine Aussage über weitere mathematischen Funktionen getroffen werden.

Eine Turing-Maschine besteht aus einem unendlich langen Speicherband, d.h. es ist mindestens so lang wie benötigt. Das Band ist in unendlich viele Felder geteilt, in welchen einzelne Zeichen gespeichert werden. Die Menge aller möglichen Zeichen heißt Bandalphabet und wird mit Γ bezeichnet; leere Felder mit $\square$, die auch *Blank* genannt werden. Die Eingabe des Programms erfolgt über das Eingabealphabet Σ. Zunächst ist auf dem Band nur die Eingabe gespeichert, die restlichen Felder sind leer.[50]

Neben diesem Speicherband gibt es einen Lese- und Schreibkopf, der von Feld zu Feld läuft und dessen Inhalt dieser verändern kann. Da es sich bei der Turing-Maschine nur um ein theoretisches Modell handelt, ist die Umsetzung als Maschine irrelevant.

Durch diese elementaren Operationen können komplexe Programme zusammengestellt werden. Das zeigt sich darin, dass alle heutigen Computer im Grundsatz Turing-Maschinen sind.[26]

Eine Turing-Maschine befindet sich in Zuständen, die Elemente der Menge Q sind, wobei q_0 der Anfangszustand ist. Diese Zustände ändern sich durch die Überführungsfunktion $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, 0, R\}$, welche einen Rechenschritt darstellt. Das Feld, auf dem der Lese- und Schreibkopf steht, wird durch δ verändert, außerdem wird durch L oder R das als nächstes zu lesende Feld angezeigt: L für links und R für rechts. 0 steht dabei für keine Bewegung und für das Ende des Algorithmus. Außerdem gibt es die Menge der Endzustände $F \subseteq Q$. Insgesamt kann eine Turing-Maschine durch das Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ beschrieben werden.[37] (Vergleiche hierzu auch [11].)

Ein Beispiel soll die Funktionsweise verdeutlichen (vgl. [37]). Sei $Q = \{s_1, s_2 s_3, s_4, s_5, s_6\}$ die Menge aller Zustände, $\Sigma = \{1\}$, $\Gamma = \{0, 1\}$,

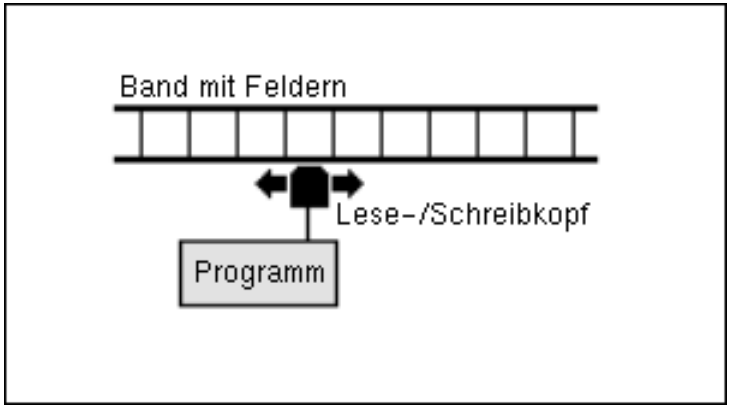

Abbildung 2.2: Die Turing-Maschine [50]

$q_0 = s_1$ und $F = \{s_6\}$ der Endzustand. Die Funktion δ ist folgendermaßen definiert:
$\delta(s_1, 1) = (s_2, 0, R)$, $\delta(s_1, 0) = (s_6, 0, 0)$
$\delta(s_2, 1) = (s_2, 1, R)$, $\delta(s_2, 0) = (s_3, 0, R)$
$\delta(s_3, 1) = (s_3, 1, R)$, $\delta(s_3, 0) = (s_4, 1, L)$
$\delta(s_4, 1) = (s_4, 1, L)$, $\delta(s_4, 0) = (s_5, 0, L)$
$\delta(s_5, 1) = (s_5, 1, L)$, $\delta(s_5, 0) = (s_1, 1, R)$

Das Programm verdoppelt die Anzahl der eingegebenen Einsen und fügt ein leeres Feld, hier als 0 dargestellt, in die Mitte ein, so wird zum Beispiel aus der Eingabe {11} die Ausgabe {11011}. Das geschieht so, dass zunächst das erste Feld mit der 1 auf 0 gesetzt wird, um die 1 zu markieren, die hinten angefügt wird. Dann bewegt sich der Kopf so lange nach rechts, ohne etwas in den Feldern zu verändern, bis eine 0 gefunden wird. Das Feld rechts neben der Null wird auf 1 gesetzt. Dann wird ganz links wieder die 0 auf 1 gesetzt. Danach wird das zweite Feld auf 0 gesetzt, da nun für diese 1 eine weitere hinten angefügt werden soll. Anschließend läuft der Kopf nach rechts, bis es nach der 1 hinter der 0 in der Mitte

die nächste 0 findet, die zur 1 geändert wird. Nun wird das zweite Feld wieder auf 1 zurückgesetzt. Diese Schritte werden so lange wiederholt, bis für jede Eins vor der Mitte eine hinter dieser geschrieben wurde. Im Einzelnen sehen die Schritte so aus:

Das Speicherband sieht zu Beginn des Programms folgendermaßen aus: Über dem fett gedruckten Feld steht der Kopf.

Schritt	Zustand	Band				
1	s_1	**1**	1	0	0	0

Der Lese- und Schreibkopf steht zu Beginn auf dem ersten Feld mit der 1. Da $q_0 = s_1$, ergibt die Überführungsfunktion δ nach Definition im ersten Schritt $(s_2, 0, R)$. D.h. die erste Speicherzelle wird auf 0 gesetzt und der Kopf bewegt sich zum rechten Feld, sodass das Band sich so verändert hat:

Schritt	Zustand	Band				
2	s_2	0	**1**	0	0	0

Im zweiten Feld steht die 1, sodass der Kopf sich wieder nach rechts bewegt. Der Zustand verändert sich in diesem Schritt nicht. Da im dritten Feld allerdings eine 0 steht, ändert sich der Zustand im vierten Schritt zu s_3. Die weiteren Schritte ergeben sich aus der Definition der δ-Funktion.

Schritt	Zustand	Band				
3	s_2	0	1	**0**	0	0
4	s_3	0	1	0	**0**	0
5	s_4	0	1	**0**	1	0
6	s_5	0	**1**	0	1	0
7	s_5	**0**	1	0	1	0
8	s_1	1	**1**	0	1	0
9	s_2	1	0	**0**	1	0
10	s_3	1	0	0	**1**	0
11	s_3	1	0	0	1	**0**
12	s_4	1	0	0	**1**	1
13	s_4	1	0	**0**	1	1

Schritt	Zustand	Band
14	s_5	1 **0** 0 1 1
15	s_1	1 1 **0** 1 1
16	s_6	die Maschine hält an

Die Maschine hält, wenn als Bewegungsrichtung für den Kopf die 0 angegeben wird. Wie in der Tabelle zu sehen, ist die Zahl der Einsen verdoppelt worden und in der Mitte ist ein leeres Feld geblieben.

Von der Turing-Maschine gibt es viele Variationen. Zum Beispiel kann eine Maschine mehrere Speicherbänder besitzen, die je über einen eigenen Lese- und Schreibkopf verfügen. Hierdurch ändert sich lediglich die Überführungsfunktion, aber nicht die Leistung der Maschine.[50] Die universelle Turing-Maschine erhält die Überführungsfunktion als Eingabeparamter, sodass sie nicht mehr auf ein Programm festgelegt ist.

Andererseits gibt es mehrere Turing-Maschinen, die auf ein bestimmtes Programm festgelegt sind. Als Beispiel wird hier der „Fleißige Biber“ vorgestellt. Die spezielle Fragestellung, die mit Hilfe dieser Maschine gelöst werden soll, lautet: Wenn man eine feste Anzahl von Zuständen vorgibt, können damit nur eine bestimmte Anzahl von Einsen ausgegeben werden. Denn die Maschine soll nach endlich vielen Schritten halten. Diese Anzahl der Einsen soll bestimmt werden.[51]

2.4 Intelligenz von Maschinen

2.4.1 Der Turing-Test

Neben dem theoretischen Konstruieren von Computern beschäftigte sich Turing mit der Fragestellung, ob Maschinen eines Tages denken können. Er war einer der ersten, der sich mit diesem Gebiet der künstlichen Intelligenz beschäftigte.[26]

Die Frage, ob Maschinen denken können, sei sehr schwer zu beantworten, da die Begriffe „Maschine“ und „Denken“ nicht genau definiert seien,

weshalb Turing sie umformulierte. Er stellte sich ein Spiel vor, das er *Imitationsspiel* nannte, und das folgendermaßen funktioniert: Es spielen drei Personen A, B und C, wobei A und B unterschiedlichen Geschlechts sind. Die Aufgabe von C ist es, anhand von Fragen herauszufinden, ob A ein Mann und B eine Frau ist oder umgekehrt. Damit dieses Spiel nicht trivial wird, befindet sich C in einem anderen Raum als A und B, und es wird mit Hilfe von Fernschreibern miteinander kommuniziert. Person B wird aufgefordert, immer ehrlich zu antworten, um so dem Fragesteller C zu helfen. Die andere Person bekommt die Aufgabe, durch zweideutige Antworten C hinters Licht zu führen. [10] (S. 9 ff)

Im Zusammenhang mit der Frage „Können Maschinen denken?“ funktioniert das Spiel so, dass die Person A durch eine Maschine ersetzt wird und C die Aufgabe bekommt, herausfinden, welcher der beiden Antwortgeber eine Maschine, welcher ein Mensch ist. Die Maschine versucht dabei, sich so menschlich wie möglich zu verhalten und darf sich bei Rechenaufgaben nicht durch zu schnelle Antworten verraten. Die ursprüngliche Frage wird ersetzt durch die Frage, ob die Maschine den Fragesteller C ebenso hinters Licht führen kann, wie ein Mensch. Falls C Mensch und Maschine nicht klar unterscheiden kann, ist das Spiel gewonnen und der Maschine kann daher künstliche Intelligenz zugesprochen werden. Dieses Spiel wird *Turing-Test* genannt. Ist das Spiel gewonnen, so ist der Test bestanden[38]. Inwieweit diese Umformulierung gerechtfertigt ist, soll im Folgenden diskutiert werden.

Eine Unformulierung der eigentlichen Frage ist nach Turing dringend erforderlich, da die Begriffe „Maschine“ und „Denken“ nicht klar definiert seien. Um sie zu definieren, könne deren Gebrauch beobachtet werden. Dazu wendet Turing allerdings ein, dass dann auch die Frage des Denkens indirekt durch eine Umfrage beantwortet werden würde. Da das aber nicht in seinem Sinne sei, bliebe Turing nur die Ersetzung durch den Turing-Test. Durch die oben genannte neue Formulierung werden die physischen stark von den intellektuellen Fähigkeiten eines Menschen unterschieden.

Als Gegenargumente werden unter anderem genannt, dass die Simulation einer Unterhaltung als Intelligenz angesehen werde. Und im Gegensatz

dazu wird die Frage aufgeworfen, ob jede als intelligent bezeichnete Maschine den Turing-Test bestehe. Kleine Kinder und psychiatrisch kranke Menschen würden den Test wohlmöglich auch nicht bestehen. Eine Maschine allerdings könne als intelligent bezeichnet werden, wenn sie über eine ausreichend große Speicherkapazität verfüge. Denn sie könne viele Gesprächsverläufe speichern und müsse für passende Antworten lediglich den Speicher durchsuchen.[38]

2.4.2 Pro und Contra

Während Turing sich mit der zentralen Fragestellung, ob Maschinen denken können, auseinandersetzte, sammelte er auch die Meinungen anderer, die er in einem seiner Werke „Maschinelle Rechner und Intelligenz“[20](S. 62-72) zusammenstellt. Er selbst schreibt dazu, dass er sich vorstelle, dass in fünfzig Jahren, zum Ende des 20. Jahrhunderts, die Computer die Fähigkeit besitzen, dass ein Fragesteller in 30% der Fälle zu einem falschen Ergebnis komme, d.h. der Test zu 30% bestanden werde. Zu der eigentlichen Frage äußert er sich nicht, da er der Überzeugung ist, dass sie einer Diskussion nicht würdig sei. Denn die Bedeutung der Formulierung werde sich in den folgenden Jahren grundlegend ändern. Im Folgenden werden die Gegenargumente, die Turing in seinem Artikel beschreibt, dargestellt.

Zunächst beschäftigt sich Turing mit dem *theologischen Einwand*. Dass angeblich nur Menschen denken könnten und nicht einmal die Tiere, ganz abgesehen von Maschinen, akzeptiere er nicht, da er sich höchstens auf die Einteilung Lebewesen und Maschinen, also belebt und unbelebt einlassen könne. Die oben genannte Einteilung, die Tiere und Maschinen auf eine Ebene setze, sei außerdem willkürlich, da sie zum Beispiel im Islam anders gesehen werde. Denn dort haben nicht einmal Frauen eine Seele. Als Hauptargument wird genannt, dass nur Gott den Menschen eine Seele und damit die Möglichkeit zum Denken geben könne. Somit könnten die Menschen keine neuen Seelen schaffen, sondern nur „Wohnungen für die Seelen“[20](S. 63), womit es in Gottes Hand läge, den von Menschen erschaffenen Maschinen eine Seele zu geben. Turing

selbst steht aber nicht hinter diesem Argument, da er theologische Begründungen generell für unzulänglich halte.

Danach wird der „*Vogel-Strauß-Einwand*“ betrachtet (im englischen Original: „‚Heads in the Sand' Objection“ [19](S. 58)), der die Konsequenzen mit einbezieht, die entstehen, wenn Maschinen denken könnten. Nach Turing zieht niemand diese gerne in Betracht, da die Menschheit per Definition allem auf Grund des Denkens als überlegen angesehen werde. Dieser Einwand hat in Turings Sicht zu wenig Gehalt, um als Grund für das Nichtdenkenkönnen von Maschinen angesehen werden zu können.

Zum *mathematischen Einwand* greift er auf den Unvollständigkeitssatz von Gödel zurück. Da Gödel sich in diesem auf logische Systeme bezieht, hat Turing den Satz für seine Maschinen umformuliert. So sagt er, dass es Fragen gibt, auf die eine Maschine nicht antworten könne, weder mit einer falschen noch mit einer richtigen Antwort. Das entspräche den Behauptungen in Gödels logischen Systemen, die weder bewiesen noch widerlegt werden können. Er gehe in diesem Fall zur Vereinfachung davon aus, dass nur Fragen gestellt werden, auf die mit Ja oder Nein geantwortet werden könne. Als Beispiel für eine nicht zu beantwortende Frage nennt er die Folgende, wobei ‚soundso' für eine genaue Beschreibung steht. „Stell dir eine Maschine vor, die im Einzelnen soundso aussieht. Wird diese Maschine je auf eine Frage mit 'Ja' antworten?“[20](S. 64). So sieht Turing eine Reihe von Unzulänglichkeiten bei Maschinen, die es beim Menschen nicht gäbe. Auf Grund der beschränkten Leistungsfähigkeit einer Maschine wirft er die Frage auf, ob die Intelligenz bei Menschen nicht auch beschränkt sei, denn auch diese sind fehlbar. Deshalb sieht er darin keinen Beweis, dass Maschinen nicht denken könnten.

Anschließend zitiert er Professor Jefferson (zitiert nach [20] S. 65), der sich zum *Bewusstseins-Argument* äußert. Dieser behauptet, dass, falls eine Maschine ein Gedicht schreibe, dies lediglich eine Zufallskonstellation von Symbolen sei, aber keine Gefühlsäußerung. Eine Maschine sei nicht in der Lage, Gefühle zu äußern, sich zum Beispiel über einen Erfolg zu freuen, oder Trauer zu zeigen, falls eine Komponente der Maschine kaputt gehe. Diese Eigenleistung des Computers wäre nicht möglich, da eine Maschine kein Bewusstsein besäße. Der *solipsistische Standpunkt*

zu diesem Argument ist der extremste. Der besagt, dass man selbst eine Maschine sein müsse, um zu fühlen, ob eine Maschine denke. Daraus resultiere das Problem, wie man anderen Menschen seine Erkenntnisse glaubhaft mache. Dazu müsse man ebenso der andere Mensch sein, um dies zu erfahren. Um herauszufinden, ob jemand denken kann, schlägt Turing vor zu testen, ob Fragen konsistent beantwortet werden können.

Im Anschluss wird die *Argumentation mit verschiedenen Unzulänglichkeiten* erwähnt. Als Unzulänglichkeit wird unter anderem verstanden, dass Maschinen nie Eigenschaften haben werden wie: nett, schön und freundlich sein, Sinn für Humor haben, Erdbeeren mit Schlagsahne mögen oder jemandem den Kopf verdrehen. Maschinen seien zu seiner Zeit allerdings nur für einen bestimmten Zweck konstruiert worden und könnten nicht zu einem anderen verwendet werden. Zum Zweck der Schönheit, des Nettseins seien noch keine Maschinen konstruiert worden, sodass davon ausgegangen werde, dass diese Eigenschaften nicht programmiert werden könnten. Turing bezeichnet dies als *empirische Induktion*. Ob Maschinen wirklich Erdbeeren werden essen können, soll dahin gestellt bleiben, als Unzulänglichkeit wird aber genannt, dass eine Freundschaft zwischen Mensch und Maschine nicht mit einer Freundschaft zwischen zwei Menschen verglichen werden könne.

Als weitere Unzulänglichkeit wird genannt, dass Maschinen keine Fehler machen würden. Im Imitationsspiel würde es also reichen, Rechenaufgaben zu stellen, um die Maschine vom Menschen zu unterscheiden. Nun könne man die Maschine, die C verwirren soll, so programmieren, dass auch sie Rechenfehler mache. Dann hätte die Maschine zu entscheiden, wann und wo sie Fehler einstreue, sodass nach Turing zunächst zwei Arten von Fehlern zu unterscheiden seien. Zum Einen die Funktionsfehler, welche Hard- oder Softwarefehler sind, und zum Anderen Entscheidungsfehler. Falls zu dieser Diskussion nur theoretische Maschinen in Betracht gezogen werden, wie zum Beispiel in der Philosophie, sind Funktionsfehler ausgeschlossen. Als Entscheidungsfehler bezeichnet Turing solche, bei denen Maschinen falsche Schlüsse ziehen. Diese Fehler könnten hingegen von allen Maschinen gemacht werden. Weiter lässt sich fragen, ob der Gegenstand des Denkens einer Maschine die Maschine selbst sein

kann. Das hieße, Maschinen könnten an Hand ihrer eigenen Verhaltensweise ihre Programme verändern, eine Eigenschaft, die mit dem Denken gleichzusetzen ist. Dass Maschinen nicht verschiedene Verhaltensweisen zeigen könnten, ist nach Turing lediglich eine Frage der Speicherkapazität.

Als sechsten Einwand nennt er den von *Lady Lovelace*. Diese behauptet, dass eine Maschine nie etwas wirklich Neues schaffen oder Überraschungen bereiten könnte, da Maschinen nur das tun würden, was man ihnen befielt. Dazu fragte Turing, was etwas Neues sei. Denn es sei alles da und müsse nur noch entdeckt werden. Außerdem berichtet Turing von seinen Erfahrungen, dass er sowohl von Maschinen überrascht wurde, wie auch von Büchern. Da Bücher aber nicht denken könnten, ist die Fähigkeit zu überraschen keine, die Denken voraussetze.

Die *stetigen Maschinen*, die mit einem Nervensystem zu vergleichen sind, unterscheiden sich grundsätzlich von *diskreten Maschinen*, sodass diskrete die stetigen nicht nachahmen können. Wenn im Imitationsspiel von jedem Typ eine Maschine antritt, wird der Fragesteller diese jedoch schwer auseinanderscheiden können.

Anschließend diskutiert er das Argument der *Ungebundenheit des Verhaltens*. Die Befürworter sagen: „Wenn die Menschen alle genau festgelegte Verhaltensregeln hätten, durch die sie ihr Leben ordneten, wären sie nicht besser als Maschinen. Aber solche gibt es nicht, und folglich können sie keine Maschinen sein“[20](S. 71). Als Beispiel wird das Verhalten an einer Verkehrsampel genannt. Wie man sich bei grünem oder roten Licht verhalten solle, ist leicht zu definieren. Aber wie soll sich eine Maschine verhalten, falls beide Lichter leuchten? Turing argumentiert, dass zunächst zwei Begriffe definiert werden müssten. Zum Einen die Verhaltensregeln, wie etwa das Verhalten, wenn eine Ampel rot zeige, und zum Anderen die Verhaltensgesetze. Diese sind Naturgesetze des Körpers, wie dass man schreit, wenn man gekniffen wird. Turings Gegenargument ist nun, dass nicht die Verhaltensregeln, sondern die Verhaltensgesetze das Leben bestimmten. Da die Gesetze wissenschaftlich beobachtet werden könnten, seien diese existent. Also könne auch der Mensch, laut Turing, als Maschine angesehen werden, und selbst eine

diskrete Maschine könne entsprechend programmiert werden.

Zuletzt nennt Turing das für ihn bedeutendste Gegenargument zu der Ansicht, dass Computer denken könnten: das *Argument der außersinnlichen Wahrnehmung*. Dieses Argument umfasst Telepathie, Hellsehen, Vorwissen und Psychokinese. Da für das Vorhandensein der Telepathie „statistisch die Beweise [...] überwältigend“[20](S. 72) sind, seien sie nicht von der Hand zu weisen. Außerdem sei außersinnliche Wahrnehmung beim Denken „von besonderer Relevanz“[20](S. 72). In Bezug auf das Imitationsspiel hätten diese Wahrnehmungen besonderen Einfluss. Würden zum Beispiel ein Mensch mit telepathischen Fähigkeiten und ein Digitalrechner gefragt, welche Farbe die Karte in der Hand des Fragestellers hätte, wäre der Mensch im Vorteil, die richtigen Antwort zu geben und sich so erkennbar zu machen. Falls der Fragesteller über psychokinetische Kräfte verfügt, so könne er den Zufallsgenerator der Digitalmaschine beeinflussen, sodass diese häufig richtig tippt und so von beiden gleich viele richtige Antworten kommen würden. Davon hätte C wiederum keinen Vorteil. Ein Fragesteller, der hellsehen könnte, würde hingegen keine Fragen benötigen, da er so zu einem richtigen Ergebnis kommen würde.

Da Turing alle Gegenargumente entkräftigen kann, sieht er keinen Grund, warum Maschinen in der Zukunft nicht denken können sollten. Wobei er die Frage durch sein Imitationsspiel ersetzt. Turing hat damit sehr wichtige Grundlagen gelegt, was daran zu erkennen ist, dass der Turing-Test in der Informatik heutzutage eine wichtige Rolle spielt. Bis heute ist es aber nicht gelungen, einen Computer so zu programmieren, dass er den Turing-Test besteht. Der Loebner-Preis, der seit 1991 ausgeschrieben ist, soll verliehen werden, sobald es gelingt, den Turing-Test zu bestehen.[38]

Kapitel 3

Komplexität und Grenzen formaler Argumentation: Gregory Chaitins Interpretation von Gödel

Janina Ried

3.1 Einleitung

Der folgende Artikel beschäftigt sich mit dem Unvollständigkeitstheorem von Gödel in Bezug auf die Ansätze des amerikanischen Mathematikers Gregory Chaitin.
Im Vordergrund stehen dabei die Entwicklung des Begriffs der algorithmischen Komplexität durch Chaitin und dem daraus folgenden informationstheoretischen Unvollständigkeitstheorem, sowie die Überlegungen zur Grenze formal logischer Argumentation und ihrer Bedeutung für die mathematische Forschung (siehe dazu auch 2 von Lena Borgman und 6 von Sören Dobberschütz).

Abbildung 3.1: Gregory Chaitin im „Gödel classroom“ an der Universität von Vienna [48]

Dabei wird im ersten Teil auf die Fortführung der Theorien von Gödel durch Chaitin unter Verwendung der Erkenntnisse Turings durch die Entwickung der neuen Methode zur Messung der Programmkomplexität nach der Bitanzahl eingegangen. Desweiteren folgt die Darstellung des Zusammenhangs von Zufall, „Nichtkomprimierbarkeit“ und Unvollständigkeit, der letztlich zum informationstheoretischen Unvollständigkeitstheorem führt. Anschließend daran werden die Auswirkungen auf die Frage nach der Existenz einer Grundtheorie der Mathematik beschrieben und die Forderungen Chaitins für den Umgang mit den Erkenntnissen der Unvollständigkeit in der mathematischen Forschung dargelegt. Dabei wird auch auf die Forderungen nach einem Umdenken der gesamten mathematischen Welt im Sinne einer Annäherung an experimentelle Wissenschaften unter bestimmten Voraussetzungen eingegangen.
Vor der Behandlung dieser Aspekte steht aber zuerst einer kurze biographische Vorstellung des Mathematikers Gregory Chaitin.

3.2 Chaitin - Biographie

Gregory Chaitin wurde 1947 in New York geboren und wuchs in Manhatten auf. Da seine Begabung schon früh erkannt wurde, nahm er als Schüler an vielen Programmen für hochbegabte Kinder teil. Von besonderer Bedeutung für seine (mathematische) Entwicklung war dabei das Science Honors Program der Columbia Universtität während sei-

ner High-School Zeit, dessen Programm die Beschäftigung mit Computern, seinerzeit noch nicht so gewöhnlich wie heute, vorsah. Diese ersten Erfahrungen und die ausgelöste Faszination führten zu Chaitins erster Veröffentlichung („An Improvement on a Theorem of E. F. Moore“).

Chaitin selbst beschreibt Mathematik nur als seine zweite Liebe, galt seine erste doch der Physik und Astronomie. Mathematik diente ihm anfangs nur als Handwerkszeug, um Physik zu verstehen, später wechselte er aber aus verschiedenen Gründen ganz zur Mathematik. Allerdings hat er im Gegensatz zu einigen anderen Mathematikern den Bezug der Mathematik zur „Außenwelt“ nie aus den Augen gelassen (vgl. [2] S.1-3).

Nach der High School wechselte Chaitin auf das City College der Universität von New York. Schon im Sommer 1964 vor diesem Wechsel begann seine Auseinandersetztung mit Gödels Unvollständigkeitstheorem. Angeregt durch die Lektüre der Abhandlungen von Neumanns und Morgensterns gelangen Chaitin im Sommer 1965 bedeutende Arbeiten über Zufall im Zusammenhang mit Programmkomplexität und in Anwendung auf Turingmaschinen. Er wurde sogar vorübergehend von seiner Unterrichtspflicht entbunden , um diese Ergebnisse festzuhalten (u.a. „On the length of programs for computing finite binary sequences by bounded-transfer Turing machines“ (1966)). Nach diesen Veröffentlichungen hatte Chaitin bereits mit 18 Jahren schon insgesamt drei verschiedene Theorien zur Programmkomplexität entwickelt (vgl. [2] S.5).

Im darauffolgenden Jahr zog Chaitins Familie nach Buenos Aires, diesen Umzug sieht Chaitin selbst als Beginn seines Lebens als Erwachsener. Noch im selben Jahr begann er bei dem Unternehmen IBM als Programmierer hauptberuflich zu arbeiten und seine mathematische Forschung nur noch als „Hobby“ zu betreiben. Trotzdem erschienen in den folgenden Jahren viele Arbeiten von ihm, z.B über selbstreproduzierende Automaten. Im Alter von 21 Jahren (1970) entwickelte Chaitin dann seine informationstheoretische Formulierung des Unvollständigkeitstheorems von Gödel.

In den sich anschließenden Jahren erweiterte er diesen Ansatz und veröffentlichte diverse Artikel („Computational complexity and Gödel's

incompleteness theorem“ (1970)). Desweiteren war dies auch der Punkt, an dem er mit der Programmiersprache LISP in Kontakt kam. Diese Sprache benutzte er im folgenden u.a. zur weiteren Untersuchung seiner informationstheoretischen Ansätze zur Unvollständigkeit (vgl. [2] S.6-8). Nach der Veröffentlichung weiterer Ergebnisse über Unvollständigkeit und Zufall (u.a. „Information-theoretic computational complexity“ (1974) wurde Chaitin 1976 ständiges Mitglied des IBM Watson Research Center.
Neben seiner Forschungstätigkeit an diesem Institut in New York lehrt Gregory Chaitin an der University of Buenos Aires und der University of Auckland. Die University of Maine verlieh ihm 1995 die Ehrendoktorwürde (vgl. [52]). Insgesamt hat er bis heute neun Bücher geschreiben, das letzte „Meta Math!“ ist 2005 im Pantheon-Verlag erschienen, und ungezählt viele Artikel veröffentlicht (vgl. [57]).
Die Beschäftigung mit Unvollständigkeit hat sich wie ein roter Faden durch Chaitins Leben gezogen. Im Folgenden werden seine grundsätzlichen Erkenntnisse in diesem Themenfeld vorgestellt.

3.3 Algorithmische Komplexität - *Informationstheoretische Ansätze zur Unvollständigkeit*

Gregory Chaitin schuf die algorithmische Informationstheorie, gedacht als Weiterentwicklung bzw. Intensivierung von Gödels Unvollständigkeitstheorem. Anknüpfend an die Arbeiten Turings und von Neumanns entwickelte Chaitin den neuen Begriff der algorithmischen Komplexität, der es ermöglicht, die Unvollständigkeit, die Gödel nur im speziellen gezeigt hat, zu verallgemeinern. Dabei geht er insbesondere auf den Zufall ein, den er mit der Unvollständigkeit in Verbindung bringt. Ausgehend von der Unvollständigkeit eines Axiomensystems stößt er dabei auf die Grenzen formal logischer Argumentation.

Vor der Darlegung der grundlegenden Erkenntnisse Chaitins werden die Ideen Gödels und Turings kurz dargelegt, um später direkte Bezüge herstellen zu können (s. dazu auch Kapitel 2 von Lena Borgmann).
Gödel widerlegte Hilberts Traum („Hilbert´s dream“ vgl. [22]) einer vollkommen in sich geschlossenen mathematischen Theorie, als er zeigen konnte, dass es wahre (mathematische) Aussagen gibt, die in dem gegebenen Axiomensystem nicht bewiesen werden können. Sein Beweis dazu ist sehr kompliziert in der formalen Sprache der Zahlentheorie mathematischer Logik und im Aufbau ähnlich zu einem Computerprogramm, aber der Grundgedanke basiert auf einer mathematischen Form des Lügnerparadoxon. Gödel untersuchte das Paradoxon „Diese Aussage ist unbeweisbar.“. Aus den zwei Möglichkeiten, entweder, die Aussage ist wahr, d.h. es gibt wahre unbeweisbare Aussagen, oder die Aussage ist falsch, d.h. es gibt falsche Aussagen, die beweisbar sind, folgt für das mathematische System entweder die Unvollständigkeit oder die Inkonsistenz, wobei letztere für jedes mathematische System ausgeschlossen werden muss (vgl. [23]). Die Frage, die Gödel unbeantwortet läßt, ist, ob es sich bei dem Phänomen der Unvollständigkeit um ein seltenes, spezielles oder um ein häufiges, weitverbreitetes Phänomen handelt (vgl. [25]). Dieser Frage geht Chaitin in seinen Arbeiten nach.
Wenige Jahre nach Gödel konstruierte Alan Turing einen hypothetischen Computer, die Turingmaschine, und beschrieb das „Halting Problem“ auf dieser, d.h. die Tatsache, dass es keine generelle Prozedur, kein Programm gibt, das entscheidet, ob ein Programm endet oder unendlich weiterläuft. Der Beweis dazu hat Ähnlichkeit mit dem von Gödel, beide arbeiten mit Paradoxien. Es wird ein Programm der Länge n-bits konstruiert, das unter Verwendung einer solchen allgemeinen Prozedur als Output eine Zahl liefert, die mindestens ein $n + 1$-Bits langes Programm für ihre Konstruktion, d.h. Ausgabe benötigt. In Erweiterung der Theorie Gödels zeigte Turing somit, dass „incompletness is natural and pervasive“ (vgl. [25]). Chaitin verfolgt diesen von Turing eingeschlagen Weg, die „Natürlichkeit der Unvollständigkeit“ aufzuzeigen weiter, dazu verwendet er in seinen Arbeiten häufig die Turigmaschine als formalen Ansatz.

Einer der Hauptunterschiede zu seinen Vorgängern ist Chaitins Idee zur Messung der Programmkomplexität. Im Gegensatz zu seinen Zeitgenossen, die die Zeit, die ein Computer braucht, um ein Programm zu erzeugen, als Maß für die Komplexität ansahen (z.B. von Neumann), ist es Chaitins Ansatz die Anzahl der Bits zu messen, die ein Programm benötigt, um gegebene Informationen zu generieren. Die algorithmische Information oder Komplexität, die eine Datenmenge bzw. ein Programm für deren Generierung enthält, ist die minimale Anzahl an Bits, die das kleinste dieser Programme benötigt (vgl. [22] und [25]).
Dieses Maß verknüpft Chaitin mit dem Zufall, bzw. der nicht Komprimierbarkeit einer Datenmenge. Ein illustrierendes Beispiel hierfür ist das Verhalten eines Gases. Die Analogie zum Grad der algorithmischen Komplexität ist hierbei die Entropie des Systems, also des Gases. Die Atomverteilung in einem Gas zu beschreiben, benötigt ein verhältnismäßig langes Programm, da die Entropie, und damit der Zufall, relativ hoch ist. Dagegen kann man einen Kristall mit seiner regelmäßigen Anordnung und geringen Entropie in einem relativ kurzen Programm beschreiben. Es gilt also je höher das Chaos (die Entropie) einer Datenmenge (z.B. Atomverteilung in einem Gas), desto länger das (kleinste) Programm, desto höher die algorithmische Information (der Datenmenge) bzw. Komplexität (des Programms zur Erzeugung der Datenmenge)(vgl. [22]). Chaitin geht sogar noch einen Schritt weiter und definiert algorithmischen Zufall als die „Nichtkomprimierbarkeit“ („algoritmic randomness“ [23]). Demnach ist eine Datenmenge zufällig, wenn die Beschreibung der Datenmenge durch ein Programm nur durch die Aufzählung jedes einzelnen Datenpunkts möglich ist. Das kann man z.B. mit einem Münzwurf verdeutlichen: Wirft man eine Münze 1000 mal, sind die Ereignisse Kopf und Zahl zufällig, die einzige Möglichkeit die Datenmenge mit einem Programm zu beschreiben ist also zu sagen „das erste Ergebnis war ..., das zweite Ergebnis war ..., ..., das 1000ste Ergebnis war ...“. Legt man die Münze dagegen ordentlich abwechselnd auf die Kopfseite und auf die Zahlseite und auch das 1000 mal, so ist das Programm dazu mit „500mal Abfolge Zahl, Kopf“ deutlich kürzer. Daran sieht man, dass zwei gleichgroße Datenmengen durchaus sehr verschiedene algorithmische Komplexität besitzen können und der Grund hierfür der Zufall ist.

Das Problem der „Nichtkomprimierbarkeit“ im Zusammenhang mit Zufall, bringt Chaitin durch die Definition seiner Zahl Ω auf den Punkt. Ω ist die Wahrscheinlichkeit dafür, dass ein zufälliges Programm hält oder nicht. Nach der genauen Definition ist Ω die Summe aller Wahrscheinlichkeiten für das Eintreten eines anhaltenden Programm. Dazu gibt Chaitin in einem Artikel folgendes Beispiel (vgl. [2] S.8): Angenommen in einem speziellen Computer gibt es nur genau drei Programme, die enden und zwar 110, 11100 und 11110, alle anderen Programme enden nicht. Die Wahrscheinlichkeit, dass das erste Programm eintritt ist $\left(\frac{1}{2}\right)^3$ und dass entweder das zweite oder das dritte eintritt ist $\left(\frac{1}{2}\right)^5$, denn jedes Bit der Programme ist mit der Wahrscheinlichkeit von $\frac{1}{2}$ 1 bzw. 0. Demzufolge ist $\Omega = \left(\frac{1}{2}\right)^3 + \left(\frac{1}{2}\right)^5 + \left(\frac{1}{2}\right)^5$ und im binären Zahlensystem folgt damit für das Ω dieses speziellen Computers $\Omega = 0.001 + 0.00001 + 0.00001 = 0.00110$ (vgl. [25]). Chaitin zeigt nun, dass diese Zahl Ω nicht komprimierbar, d.h. algorithmisch zufällig ist, indem er die Annahme der Existenz eines weniger als n Bits langen Programms für die Erzeugung der ersten n Stellen von Ω mit dem Haltingproblem von Turing zum Widerspruch führt. Die Folgerung, die Chaitin daraus zieht, ist, dass es Zahlen in jedem Computersystem gibt, die nicht komprimiert werden können. Übersetzt in die Sprache der Mathematik bedeutet das, dass es in jedem Axiomensystem Theoreme gibt, die nicht bewiesen werden können, da die „Mächtigkeit der Axiome“ (s. unten) nicht ausreicht (vgl. [25]).

Diese Aussage erreicht Chaitin aber auch auf einem anderen Weg, indem er seine Definiton der algorithmischen Komplexität auf *reale* Programme anwendet. Betrachtet man die Anzahl der Bits des kleinsten Programms, um eine Datenmenge zu generieren, als das Maß für deren Komplexität, so stellt sich automatisch die Frage nach dem kleinsten Programm. Es wird also das Wissen über das kleinste mögliche Programm benötigt, gerade dessen kann man sich aber nicht sicher sein (vgl. [22]). Chaitin begründet, dass es unmöglich ist zu zeigen, dass ein Programm mit einer Größe von mehr als n Bits das Kleinstmögliche ist, wenn man nur Axiome der Größenordnung von n Bits hat, um die Aussage davon abzuleiten. Das bedeutet im Allgemeinen, dass man keine Aussage beweisen kann, deren programmierte Länge größer ist als die Länge der programmierten

Version der Axiome in diesem System (vgl. [22]). Auf den Punkt gebracht hat Chaitin diese Überlegung mit seiner Äußerung:

> „... if one has ten pounds of a set of axioms and a twenty-pound theorem, then that theorem cannot be derived from those axioms." ([23])
> „... wenn man 10 Pfund einer Menge von Axiomen und ein zwanzig-Pfung Theorem hat, dann kann dieses Theorem nicht von diesen Axiomen hergeleitet werden."

Diese Überlegungen werden auch als *informationstheoretisches Unvollständigkeitsaxiom* bezeichnet.
Damit hat Chaitin, ähnlich wie Turing, gezeigt, dass Unvollständigkeit in jedem System existiert, bzw. schärfer formuliert, dass kein System vollständig sein kann, wobei Chaitin insofern auch über Turing hinausgeht, als dass er die Unvollständigkeit nicht nur auf die Unlösbarkeit des Haltingproblems zurückführt, sondern allgemeiner auf das „Gewichts-" Verhältnis von Theorem und Axiomsystem.

Chaitins Erkenntnise vor allem die Verknüpfung des Zufalls mit der Unvollständigkeit lassen das Unvollständigkeitstheorem von Gödel in einem anderen Licht erscheinen. Während Gödels Ideen noch als „pathologische und ungewöhnliche" (s. [23]) Phänomene aufgefasst werden konnten und wurden, ist es Chaitins erklärtes Ziel die Unvollständigkeit als natürliches Phänomen der Mathematik aufzuzeigen. Das gelingt ihm vor allem dadurch, dass er den Zufall als Grund der Unvollständigkeit anführt.
Aber Chaitins Ziele gehen noch über die theoretische Seite der Unvollständigkeit hinaus. In welcher Weise sein Unvollständigkeitstheorem der Existenz einer mathematischen Universaltheorie (TOE theory of everything vgl. [25]) widerspricht und in welcher Weise die daraus resultierenden Grenzen formaler Argumentation nach Chaitins Meinung bei der praktischen Arbeit der Mathematiker Beachtung finden sollen, wird im folgenden Abschnitt behandelt.

3.4 Grenzen formaler Argumentation - *Praktische Bedeutung der Unvollständigkeit*

Vor der Beschäftigung mit der Frage, inwiefern Chaitins Theorem die Grenzen formaler Argumentation aufweist, muss definiert werden, was unbegrenzte formale Argumentation bedeutet.
Nach Chaitin ist unbegrenzte formale Argumentation gleichbedeutend mit bzw. die logische Folgerung aus einer mathematischen Universaltheorie. Darunter versteht er ein abgeschlossenes System von Axiomen und Ableitungsregeln, mit Hilfe derer man aus den Axiomen die gesamte mathematische Wahrheit beweisen kann. Dieses Axiomensystem müsste weiterhin endlich sein bzw. endliche Komplexität besitzen, da es sonst nicht in sich geschlossen sein könnte (vgl. [44]).
In der Existens von Ω sieht er nun den Beweis, dass es keine solche Grundsatzheorie (endlicher Komplexität) der gesamten Mathematik geben kann. Denn, wenn es eine solche Grundsatzheorie gäbe, dann gäbe es auch ein Programm, das Ω konstruieren könnte. Das führt aber in zweierlei Hinsicht zu einem Widerspruch: Gäbe es ein solches Programm, so müsste dieses unendliche Komplexität besitzen, da Ω unendliche Komplexität besitzt, das widerspricht dann aber der Existenz einer Grundsatztheorie mit endlicher Komplexität. Existiert anders herum eine solche Theorie, ist jedes Objekt in der Mathematik durch ein Programm beschreibbar, so wäre auch Ω durch ein (endendes) Programm beschreibbar, was im Widerspruch zu der unendlichen Komplexität von Ω steht (vgl. [44]). Folglich kann es keine universelle Grundsatztheorie in der Mathematik geben und somit ist auch unbegrenzte formale Argumentation nicht möglich, da es eben solche „Strukturbrüche“ wie die Unkomprimierbarkeit von Ω gibt (vgl. [44]).

Die Folgerung, die Chaitin aus der Nichtexistenz einer universellen Theorie der Mathematik zieht, ist die Tatsache, dass es Theoreme in der Mathematik gibt, die ohne Grund bzw. Beweis möglich, d.h. wahr sind. Dieses widerspricht dem Weltbild des reinen Mathematikers vollkommen,

denn bis zum Auftauchen des Unvollständigkeitsbegriffs und teilweise auch noch danach wurde die Meinung vertreten, dass alles Wahre in der Mathematik auch einen Grund hat, also beweisbar ist bzw. sein muss. Dieses neue Weltbild führt Chaitin zu Überlegungen wie Mathematik heute betrieben werden soll. Seine drei grundlegenden Forderungen werden im Folgenden portraitiert.

Ausgehend von den nötigen Veränderungen im mathematischen Weltbild fordert Chaitin indirekt eine Annäherung an die Methoden der Physik. Er spricht davon, dass Mathematik und Physik gar nicht so verschieden sind, wie es beide Seiten gerne von sich glauben. In der traditionellen Sichtweise seien Physik und Mathmatik vom Grundprinzip her unterschiedlich. Die Physik steht in direktem Zusammenhang mit dem Universum, durch Experimente und Beobachtungen werden seine Regeln beschrieben, dagegen ist die Mathematik vom Universum unabhängig, sie basiert nicht auf Beobachtungen der Natur, sondern auf allgemein gültiger Logik. Dieser Unterschied wird durch die Unvollständigkeit relativiert, in diesem Sinne rücken Mathematik und Physik näher zusammen (vgl. [25]). Ein gutes Beispiel hierfür ist,nach Chaitin, der Vergleich des Verhaltens der Atome bei radioaktivem Zerfall und der Verteilung von Primzahlen. Diese beiden sehr unterschiedlichen Gebiete werden durch eine Auffälligkeit verbunden, die Chaitin einst zu seiner Vermutung führten, dass der Zufall die Ursache der Unvollständigkeit ist. Und ebendieser ist es, der die beiden Gebiete verbindet. Bei der Beobachtung des radioaktiven Zerfalls ist es unmöglich vorher zu sagen, wann das nächste Atom zerfallen wird, da dies ein zufälliger Prozess ist, man weiß nur, dass es irgendwann passieren wird und man kann den ungefähren Verlauf der Kurve der Anzahl der Atome nach der Zeit prognostizieren. Ganz ähnlich im zweiten Fall, bei der Verteilung der Primzahen in der Menge aller natürlichen Zahlen kann man nicht genau vorhersagen, wann genau die nächte Primzahl in Erscheinung tritt, man kann aber zumindest vermuten, dass nach einer Primzahl nach einem unbestimmten Intervall wieder eine Primzahl kommt (vgl. [22]). Chaitin sieht dieses Beispiel als einen Beweis für die Rolle des Zufalls in der reinen Mathematik an. Während die Physiker kein Problem damit haben, die Zufälligkeit zu

akzeptieren, fällt es den Mathematikern schwer, in dieser Weise umzudenken, obwohl Gödels und Chaitins Erkenntnisse genau so ein Umdenken anstoßen müssten. Dieses Problem greift Chaitin auf und fordert ein Umdenken der Mathematik. Zwar ist auch er nicht der Meinung, dass Physik und Mathematik dasselbe sind, aber er vertritt die Meinung, dass ein gewisses Umdenken der Mathematik in Richtung der physikalischen Denkweise notwendig ist (vgl. [25]).

In diesem Sinne stellt er Überlegungen zur Übertragbarkeit des experimentellen Prinzips der Naturwissenschaften auf die Mathematik an. Chaitin bemerkt dabei, dass die antiken Hochkulturen auch in der Mathematik oft experimentell gearbeitet haben und dabei gute Ergebnisse erzielt haben. Erst die Griechen kamen auf die Idee, dass in der Mathematik alles von einem theoretischen Standpunkt aus bewiesen werden muss. Natürlich haben die heutigen Mathematiker ganz andere Möglichkeiten als die Mathematiker der Antike, aber dennoch werden die fortgeschrittenen Möglichkeiten zur experimentellen mathematischen Forschung hauptsächlich von den Informatikern genutzt und nicht von den Mathematikern. Chaitin geht sogar so weit zu behaupten, dass viel mathematischer Fortschritt unter dem Deckmantel der Computerwissenschaften verborgen ist (vgl. [23]). In diesem Sinne fordert Chaitin ein vermehrtes Nutzen dieser erweiterten Möglichkeiten der Mathematik, anstatt die strenge Suche nach dem Beweis aller Behauptungen. Die experimentelle Mathematik versteht Chaitin als die Suche nach neuen mathematischen Erkenntnissen durch die Betrachtung vieler Beispiele mit Hilfe des Computers. Dabei kann nach Chaitins Meinung eine große Übereinstimmmung der Theorie mit der Praxis durch Computerbeispiele niemals dieselbe Überzeugungskraft haben, wie ein kurzer Beweis, unter Umständen aber eher überzeugen können als ein sehr langer, komplizierter und umständlicher Beweis (vgl. [25]). Auf die Frage hin, ob große vom Computer erzeugte Datenmengen Beweise nutzlos machen, antwortet Chaitin, dass beide Seiten, das Computerexperiment und der Beweis von Bedeutung sind. Denn ein Beweis kann Fehler enthalten (die bei sehr komplexen unter Umständen nicht auffallen) und ein Computerexperiment kann vor dem Ergebnis, das die Theorie widerlegt abbrechen (vgl. [25]).

In logischer Konseqenz dazu hat sich Chaitin mit der Frage beschäftigt, ab wann und ob überhaupt formal unbewiesene Theoreme als Axiome angenommen werden dürfen. Etwas ketzerischer formuliert geht es also darum, ob es reicht, zwei Monate vergeblich zu versuchen ein Theorem zu beweisen und man es danach als Axiom annehmen kann (vgl. [23]). In dieser Form ist die Antwort sowohl von Gödel als auch von Chaitin ein klares Nein. Aber dennoch sehen beide die Möglichkeit unter gewissen Umständen eine weiteres Axiom zuzulassen (vgl. [25]).
Das Prinzip, neue Axiome zu wählen und zu einem bestimmten Axiomensystem hinzuzufügen, ist in der Mathematik nicht unbekannt. Ein Beispiel, das Chaitin hierzu nennt ist das Parallelaxiom der euklidischen Geometrie. Dieses Axiom ist von großer Bedeutung für die euklidische Geometrie und Jahrhunderte lang versuchten Mathematiker es von den anderen Axiomen abzuleiten, ohne Erfolg. Bis schließlich die nicht-euklidische Geometrie entwickelt wurde, die das besagte Axiom durch andere ersetzt und so zu anderen Ergebnissen kommt (vgl. [25]). Trotzdem hat die Hinzunahme des Parallelaxioms sehr viele Ergebnisse gebracht, es war in diesem Sinne ertragbringend. Dieses Verhältnis zur Hinzunahme von neuen Axiomen fordert auch Gödel selbst. Er spricht dabei von Erfolg im Sinne von Fortschritt für die Mathematik bringend und nennt als Beispiel die Vereinfachung von Beweisen wahrer Theorien, die ohne das neue Axiom wesentlich komplizierter zu führen sind. Gödel sieht also in der Auswirkung auf die Mathematik eine weitere Daseinsberechtigung für Theoreme neben der klassischen Beweisbarkeit (vgl. [25] und [23]).
Chaitin vergleicht die Axiome mit dem praktischen Gegenstück zu seinen Erkenntnissen über die algorithmische Komplexität. Im gleichen Sinn seines Unvollständigkeitstheorems sind Axiome Ausdruck der logischen „Unkomprimierbarkeit“. Zwar fordert er aus diesem Grund mehr Offenheit gegenüber der Annahme neuer Axiome, sagt aber auch deutlich, dass dieses nicht willkürlich geschehen darf. Er geht sogar so weit, die Vorstellung einer in sich geschlossenen Mathematik wie Hilbert sie sich erträumte mit einer Diktatur zu vergleichen, die keinen Raum für neue Ideen lässt (vgl. [25]).

Chaitin sieht demzufolge in der Unvollständigkeit durchaus auch etwas positives, im Sinne einer mathematischen Bereicherung durch das Zulassen verschiedenster neuer, aber sinnvoller Ideen. Damit argumentiert er gegen den Pessimismus, den Gödels Unvollständigkeitstheorem kurz nach seiner Entdeckung verbreitet hat. Nach Chaitins Meinung wiegen die neuen Möglichkeiten der Mathematik den Verlust der Abgeschlossenheit auf. Zwar verliert die Mathematik in gewisser Weise ihre universelle Gültigkeit, bekommt aber im Gegenzug dafür ungeahnte Möglichkeiten der Weiterentwicklung.

Kapitel 4

Mathematisches Denken zwischen Berechenbarkeit und Intuition: Ideen und Argumente von Roger Penrose

Ann-Kristin Petersen

4.1 Einführung

Eine der bislang auf dem Gebiet der Biologie und Medizin ungeklärten Fragen ist die, wie das menschliche Denken funktioniert und was dabei im Gehirn geschieht. Wenn diese Frage beantwortet ist, kann darauf aufbauend vielleicht die Frage nach der Möglichkeit einem Computer das menschliche Denken zu implementieren beantwortet werden.
Im folgenden werde ich mich mit den Begriffen Denken und Berechenbarkeit näher auseinandersetzen, um mich daraufhin im Abschnitt 4.5 mit Hilfe des Gödel-Turing Arguments von Roger Penrose mit der Frage der Berechenbarkeit des mathematischen Denkens zu befassen.

4.2 Denken

> „Denken, die den menschlichen Erkenntnisprozess wesentlich kennzeichnende, aktive, verstandesmäßige und ordnungsstiftende Verarbeitung gegebener Informationen, mit dem Ziel, Begriffe zu bilden, Bedeutungen zu verstehen, Sinnzusammenhänge offen zu legen, Schlussfolgerungen zu ziehen, Entscheidungen zu treffen und Probleme zu lösen.“[31]

Dies ist eine Möglichkeit, wie man „Denken“ definieren kann. Eine andere ist folgende:

> „Denken umfasst als psychischer Vorgang das vorsätzliche Bemühen und den psychischen Prozess, Gegenstände zu finden, zu erfassen, zu erkennen, zu verstehen und zu unterscheiden, sie einzuordnen, zu beurteilen und als Themen zu behandeln. Es gilt als die spezifisch menschliche Fähigkeit zur Erfassung von Wirklichkeit, zur problemlösenden Daseinsbewältigung und zur Erkenntnis von Möglichkeiten und der Vergegenwärtigung von Ereignissen oder Informationen durch ikonische Systeme (Sprache, Schrift, Zeichen, Bilder).“[46]

In der zweiten Definition ist erwähnt, dass Denken als „menschliche Fähigkeit“ gilt. Doch warum ist das so? Wäre es nicht möglich, dass auch Computer denken können?
Was passiert bei einem Menschen im Gehirn, wenn er denkt? Angenommen es gäbe einen Algorithmus [1], nach dessen Schema das Denken verläuft, wäre man dann nicht in der Lage auch einen Computer zum denken zu bringen? Dann wäre Denken aber nicht mehr spezifisch

[1] Algorithmen sind „allgemeine Verfahren zur Lösung aller Aufgaben einer gegebenen Aufgabenklasse. Durch sie sollen Prozesse so beschrieben werden, daß sie danach von einer Maschine nachgebildet oder gesteuert werden können.“ [7] (S. 359) Algorithmen können somit zu einem Computer-Programm umgeschrieben werden.

menschlich. Oder gibt es diesen Algorithmus gar nicht und denken ist eher „intuitiv“? Mit diesen und ähnlichen Fragen hat sich Roger Penrose in seinem Buch „Schatten des Geistes - Wege zu einer neuen Physik des Bewußtseins“ beschäftigt.

4.3 Roger Penrose

Sir Roger Penrose (∗8. August 1931) ist ein englischer Mathematiker und theoretischer Physiker.

Abbildung 4.1: Sir Roger Penrose vor dem Fallturm der Universität Bremen

Seine Eltern sind der Genetiker, Mathematiker und Schachtheoretiker Lionel Sharples Penrose und Margaret Leathes; seine Brüder sind der Schachmeister Jonathan Penrose und der Mathematiker Oliver Penrose.
Roger Penrose ist Rouse-Ball Professor für Mathematik an der Universität Oxford und Mitglied der Royal Society.

Nach ihm sind unter anderem die Penrose-Parkettierungen, ein aperiodisches Muster, welches die Ebene parkettiert (d.h. überschneidungs- und lückenfrei überdeckt) und das Penrose-Dreieck (siehe Abbildung 4.2), ein Dreieck mit drei aufeinanderstehenden rechten Winkeln, benannt.

Im Bereich der Physik hat Roger Penrose unter anderem mit Stephen Hawking den Satz von Hawking-Penrose bewiesen, der besagt, dass es keine Lösungen der Einsteinschen Feldgleichungen ohne Singularitäten (Urknall oder Schwarze Löcher) gibt. Für diese und andere wissenschaftliche Beiträge erhielt Roger Penrose zahlreiche Auszeichnungen, unter

anderem den Adelstitel der Britischen Krone. [45, 59, 13] (Siehe dazu auch das Kapitel 5 von Jesco Humpola)

Neben dem oben erwähnten Buch „Schatten des Geistes -Wege zu einer neuen Physik des Bewußtseins“ hat Roger Penrose unter anderem noch „Computerdenken“ und „Das Große, das Kleine und der menschliche Geist“ als populär-wissenschaftliche Bücher auf dem Gebiet der Philosophie geschrieben. In allen dreien ist künstliche Intelligenz ein Thema. In „Schatten des Geistes“ diskutiert Roger Penrose die Frage der Berechenbarkeit des mathematischen Denkens.

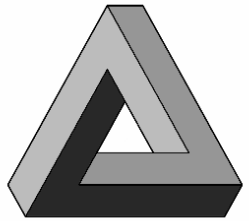

Abbildung 4.2:
Das Penrose-Dreieck

4.4 Berechenbarkeit

Da Mathematik das „Denken in reinster Form“[15] (S. 81) ist, hält Roger Penrose diesen Bereich des Denkens für denjenigen, in dem Berechenbarkeit, wenn es sie gibt, am ehesten erkannt werden kann. Mit Berechenbarkeit ist das gemeint, was eine Turing-Maschine leisten kann. Eine Turing-Maschine ist ein idealisierter Computer mit unendlich viel Speicherkapazität, bei dem keine Rechenfehler und Rechenungenauigkeiten vorkommen.(Siehe dazu auch die Beiträge von Lena Borgmann, Kapitel 2 und Sören Dobberschütz, Kapitel 6) Sie kann aber nicht nur rechnen im engeren Sinne (addieren, subtrahieren, multiplizieren, dividieren), sondern auch allgemeinere Zahlenprobleme lösen, wie z.B.:

Gesucht ist eine natürliche Zahl [2]*, die nicht als Summe von drei Quadratzahlen darstellbar ist.*[13] (S. 136)

[2]Bei diesen Zahlenproblemen ist mit natürlichen Zahlen die Menge $\mathbb{N}_0 = \{0, 1, 2, 3, \cdots\}$ gemeint.

Das Vorgehen ist nun folgendes: Die Turing-Maschine testet zuerst die kleinste natürliche Zahl, die Null, $0 = 0^2 + 0^2 + 0^2$. Damit ist die 0 darstellbar als Summe dreier Quadratzahlen. Dann kommt die Eins, $1 = 0^2 + 0^2 + 1^2$, damit ist auch sie als Summe dreier Quadratzahlen darstellbar. Analog werden die $2 = 0^2 + 1^2 + 1^2$, $3 = 1^2 + 1^2 + 1^2$, $4 = 0^2 + 0^2 + 2^2$, $5 = 0^2 + 1^2 + 2^2$ und $6 = 1^2 + 1^2 + 2^2$ überprüft. Bei der Sieben stellt die Turing-Maschine dann fest, dass sie nicht als Summe dreier Quadratzahlen darstellbar ist:

$$
\begin{aligned}
7 &\neq 0^2 + 0^2 + 0^2,\\
7 &\neq 0^2 + 0^2 + 1^2,\\
7 &\neq 0^2 + 1^2 + 1^2,\\
7 &\neq 1^2 + 1^2 + 1^2,\\
7 &\neq 0^2 + 0^2 + 2^2,\\
7 &\neq 0^2 + 1^2 + 2^2,\\
7 &\neq 1^2 + 1^2 + 2^2,\\
7 &\neq 0^2 + 2^2 + 2^2,\\
7 &\neq 1^2 + 2^2 + 2^2,\\
7 &\neq 2^2 + 2^2 + 2^2.
\end{aligned}
$$

3^2 kann kein Summand bei der Darstellung der Sieben als Summe dreier Quadratzahlen sein da $3^2 = 9 > 7$ ist. Es muss also, wenn es eine solche Darstellung gibt, eine der oben aufgeführten sein. Da jedoch keine von ihnen in Frage kommt, lässt sich die Sieben nicht als Summe dreier Quadratzahlen darstellen. Damit ist sie die Lösung dieser Berechnung.

Es gibt aber auch Berechnungen, die nicht enden, z.B.:

Gesucht ist eine ungerade Zahl, die als Summe zweier gerader Zahlen darstellbar ist.[13] (S. 138)

Eine Turing-Maschine, die dieses Zahlenproblem lösen soll, hört nicht mehr auf zu rechnen, da sie jede ungerade natürliche Zahl untersucht

und versucht diese durch Kombinationen aus geraden Zahlen, die kleiner als die ungerade Zahl sind, darzustellen. Da es aber keine ungerade Zahl gibt, die sich als Summe zweier gerader Zahlen darstellen lässt, findet die Turing-Maschine keine und kann somit kein Ergebnis ausgeben. Sie ist nicht in der Lage zu erkennen, dass diese Berechnung keine Lösung hat.
Bekommt hingegen ein Mensch diese Aufgabe, so wird er sagen: „Dieses Problem hat keine Lösung.“ Es ist keine Schwierigkeit für ihn seine Aussage durch einen Beweis zu bestätigen. Auf welchem Wege ist dieser Mensch zu seiner Antwort gekommen? Gibt es einen Algorithmus im Gehirn eines Menschen um sich davon zu überzeugen, dass gewisse Berechnungen nicht enden?
Um diese Frage beantworten zu können, müssen die Berechnungen noch etwas allgemeiner formuliert werden, nämlich in Abhängigkeit einer natürlichen Zahl n. So wird aus dem ersten Beispiel folgendes:

Gesucht ist eine natürliche Zahl, die nicht als Summe von n Quadratzahlen darstellbar ist. [13] (S. 139)

Wenn die natürliche Zahl n größer oder gleich 4 ist, endet diese Berechnung nicht. Bewiesen hat diese Aussage der Mathematiker Joseph Louis Lagrange ($*1736$, $\dagger 1813$) in einem seiner Theoreme.
Eine Änderung des zweiten Beispiels auf die gleiche Art ergibt:

Gesucht ist eine ungerade Zahl, die als Summe von n geraden Zahlen darstellbar ist.[13] (S. 139)

Für jede natürliche Zahl n ist diese Berechnung endlos.

Werden nun alle in Abhängigkeit einer natürlichen Zahl n formulierbaren Berechnungen, die für jedes n nicht endlich sind, als Computer-Programm $C(n)$ aufgefasst und, um sie aufzulisten, mit einem Ordnungsindex k indiziert, so ergibt sich eine Liste aller nicht endenden Computer-Programme $C(n)$, die von einer natürlichen Zahl n abhängen:

$$C_1(n)\, C_2(n)\, C_3(n)\, \ldots\, C_k(n)\, \ldots [15] \text{ (S. 93)}$$

Hierbei umfasst das Computer-Programm $C_k(n)$ unendlich viele nicht endende Teilprogramme (für jede natürliche Zahl n eines), d.h. das k-

te Computerprogramm $C_k(n)$ umfasst die Teilprogramme $C_k(1)$, $C_k(2)$, $C_k(3)\ldots$. Damit lässt sich obige Auflistung auch auf folgende, ausführlichere, Weise schreiben:

$$\begin{array}{lcccc}
1.\text{Programm} & C_1(1) & C_1(2) & C_1(3) & \ldots \\
2.\text{Programm} & C_2(1) & C_2(2) & C_2(3) & \ldots \\
3.\text{Programm} & C_3(1) & C_3(2) & C_3(3) & \ldots \\
 & \vdots & \vdots & \vdots & \\
k.\text{Programm} & C_k(1) & C_k(2) & C_k(3) & \ldots \\
 & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Der hier dargestellte Begriff der Berechenbarkeit ist über die Möglichkeiten, die eine Turing-Maschine besitzt, definiert. Berechenbarkeit ist also nichts rein menschliches. Stellt sich die Frage wie es mit dem Denken aussieht. Ist menschliches Denken eine Art Berechnung? Gibt es also einen Algorithmus mit Hilfe dessen es möglich ist, dass auch Computer denken können? Im folgenden Abschnitt wird aus der Annahme eines fehlerfreien Algorithmus mit Hilfe des oben beschriebenen Begriffs der Berechenbarkeit ein Widerspruch herbeigeführt.

4.5 Gödel-Turing-Argumentation von Roger Penrose

Angenommen es gibt einen Algorithmus A, der beim Überprüfen eines Computer-Programms, wie es oben beschrieben wurde, abläuft, d.h. wenn ein Mathematiker beweisen will, dass ein Computer-Programm aus obiger Auflistung nicht endet, z.B. dass man keine ungerade Zahl findet, die sich als Summe zweier gerader Zahlen darstellen lässt, dann läuft in seinem Gehirn ein Algorithmus ab, nach dessen Schema der Beweis geführt wird.

Soll nun bewiesen werden, dass das p-te Computer-Programm bezüglich

der n-ten natürlichen Zahl nicht endet, so wird der Algorithmus A angewendet auf das Paar (p, n) ablaufen, kurz $A(p, n)$.

> *Sei also mit $A(p, n)$ das Überprüfen des p-ten Computer-Programms bezüglich der natürlichen Zahl n durch den Algorithmus A gemeint, wobei ein Enden dieses Algorithmus bedeutet, dass der Beweis gefunden wurde, dass das Computer-Programm $C_p(n)$ nicht endet.*

Das bedeutet aber auch, wenn der Algorithmus $A(p, n)$ nicht endet, kann auf diesem Wege keine Aussage darüber getroffen werden, ob das Computer-Programm $C_p(n)$ endet.

In den bisherigen Betrachtungen des Algorithmus A wurde jedes Computer-Programm C_k allgemein auf die Zahl n angewendet. Wird nun $k = n$ gesetzt, d.h. der Ordnungsindex und die natürliche Zahl, auf die das Programm angewendet wird, stimmen überein, dann durchläuft der Algorithmus A alle Computer-Programme auf der Diagonalen, d.h.

$$
\begin{array}{lllll}
C_1(1) & & & & \\
 & C_2(2) & & & \\
 & & C_3(3) & & \\
 & & & \ddots & \\
 & & & & C_k(k) \\
 & & & & \qquad\ddots
\end{array}
$$

werden überprüft. Wird der Algorithmus A auf die Programme der Diagonalen eingeschränkt. D.h. es wird $A(n, n)$ für alle natürlichen Zahlen n betrachtet, so ist er nur noch von der einen natürlichen Zahl n abhängig und nicht mehr von den beiden natürlichen Zahlen p und n.

Der Schritt von einer Auflistung wie oben auf ihre Diagonale überzugehen, kommt in mathematischen Beweisen häufiger vor und wird als „Cantorsches Diagonalverfahren“ bezeichnet. Im Anhang A wird dieses Verfahren näher erläutert.

Dadurch, dass auf die Diagonale der Auflistung übergegangen wurde, ist der Algorithmus A nur noch von einer Variablen abhängig. Es wird nur noch eine natürliche Zahl n benötigt um genau anzugeben welches Computer-Programm durch den Algorithmus überprüft werden soll. Da aber in der ursprünglichen Auflistung alle Computer-Programme enthalten sind, die nur von einer Variablen abhängen, muss der Algorithmus in dieser Liste stehen.

Sei nun der Algorithmus A das k-te Programm dieser Liste. Das bedeutet dann, dass $A(n, n) = C_k(n)$ ist. Wobei der hier dazugekommene Ordnungsindex k eine feste natürliche Zahl ist. Dieses Programm kann nun auf alle natürlichen Zahlen angewendet werden, also auch auf die natürliche Zahl k. Im folgenden wird das Überprüfen des k-ten Teilprogramms des k-ten Programms durch den Algorithmus A betrachtet, d.h. $A(k, k)$ bzw. $C_k(k)$, da $A(k, k) = C_k(k)$.

Wenn nun $A(k, k)$ endet, dann bedeutet dies nach Definition des Algorithmus A, dass das Computer-Programm $C_k(k)$ nicht endet (vgl. Seite 58). Nun ist aber $A(k, k) = C_k(k)$. Da $C_k(k)$ nicht endet, kann auch $A(k, k)$ nicht enden. Aus der Tatsache, dass $A(k, k)$ nicht endet, kann aber kein Rückschluss auf die Endlichkeit von $C_k(k)$ getroffen werden (dieser Zusammenhang folgt auch aus der Definition des Algorithmus A (Seite 58)).[15, 13]

Somit wurde „eine Berechnung $C_k(k)$ gefunden, von der wir wissen, dass sie nicht anhält; aber das vorgegebene Rechenverfahren A ist nicht leistungsfähig genug, um diese Tatsache zu beweisen. Dies ist der Gödel-(Turing-) Satz in der Form, auf die ich [Roger Penrose] mich stütze.“[15] (S. 95) (Siehe dazu auch die Beiträge von Jana Bohnstengel, Kapitel 1 und Lena Borgmann, Kapitel 2)

Gegen diesen Gödel-(Turing-) Satz von Roger Penrose gibt es diverse Einwände, z.B. dass in mathematischen Überlegungen viele verschiedene Herleitungen verwendet werden, es also nicht nur einen Algorithmus A geben dürfte, sondern mehrere notwenig wären. Dieser Einwand wurde von Roger Penrose dadurch ausgeräumt, dass eine endliche Liste von Algorithmen zu einem einzelnen Algorithmus zusammengefasst werden kann.[15] (S. 97)

Wie der Name Gödel-(Turing-) Satz schon zum Ausdruck bringt, stützt Roger Penrose sich auf Gedanken von Kurt Gödel (∗1906, †1978) und Alan Mathison Turing (∗1912, †1954). Neben der Folgerung, dass „menschliche Mathematiker [...] zum Nachweis mathematischer Wahrheit keinen nachweislich korrekten Algorithmus“ verwenden, zu dem sowohl Kurt Gödel als auch Alan Turing kamen, zogen beide darüber hinaus noch unterschiedliche Schlüsse.[15] (S. 95). Trotz Gödels Glauben mathematische Einsicht nicht auf Berechnungen reduzieren zu können, schloss er dieses dennoch nicht vollständig aus. So sagte er:

> „Andererseits bleibt es auf der Basis des bislang Bewiesenen möglich, daß es eine Theorem-Beweismaschine geben könnte [...], welche tatsächlich der mathematischen Intuition gleichwertig *ist*. Das kann aber nicht bewiesen werden - ebensowenig wie *bewiesen* werden kann, daß diese Maschine in der Zahlentheorie nur *korrekte* Sätze liefert.“[15] (S. 160 f)

Nach Alan Turing ist der Gödel-(Turing-) Satz verträglich mit der Vorstellung, dass „Mathematiker im Grunde Computer sind, wenn nur die algorithmischen Verfahren, nach denen sie vorgehen, um mathematische Wahrheiten zu ermitteln, im Prinzip *nicht fehlerfrei* sind“[13] (S. 143), wie man an folgendem Zitat erkennen kann:

> „Anders gesagt, eine Maschine kann dann, wenn erwartet wird, daß sie unfehlbar ist, nicht auch intelligent sein. Es gibt mehrere Sätze, die fast genau diese Aussage machen. Aber diese Sätze sagen nichts darüber, wieviel Intelligenz sich zeigen könnte, wenn eine Maschine nicht vorgibt, unfehlbar zu sein.“[13] (S. 143)

Damit ist gezeigt, dass es diesen angenommenen fehlerfreien Algorithmus nicht gibt. Es ist aber noch nicht ausgeschlossen, dass es einen fehlerbehafteten oder einen nicht-erkennbaren Algorithmus geben könnte. Diese Annahmen werden im folgenden Abschnitt weiter diskutiert.

4.6 Veränderte Annahmen

Sei also im folgenden die Annahme gemacht, dass es einen nicht fehlerfreien Algorithmus gibt. Das bedeutet, dass es keine Sicherheit gibt, dass die Beweise, die bisher geführt wurden, auch wirklich stimmen und nicht einen Widerspruch enthalten, der nicht erkennbar ist, weil der Algorithmus, mittels dem die Beweise geführt wurden, nicht fehlerfrei ist. Wenn dem so ist, kann man sich nicht einmal mehr sicher sein, dass die Überlegungen zu Wirkungsweisen der Welt wirklich so stimmen. Da Mathematik in vielen Erklärungen der Naturwissenschaften miteinbezogen ist, müssten alle diese in Frage gestellt werden.

Mit nicht fehlerfrei ist hier nicht das Vorhandensein von erkennbaren und korrigierbaren Fehlern gemeint, sondern, dass die Folgerungen als ganzes sich widersprechen, obwohl jede Einzelheit stimmt. Diese Widersprüche sind aber nicht unbedingt sofort erkennbar.
Ein Beispiel ist das Russellsche Paradoxon. Dieses hat Bertrand Russel (∗1872, †1970) in einer Arbeit von Gottlob Frege (∗1848, †1925) entdeckt, durch einen vorhandenen Widerspruch bei dem Umgang mit unendlichen Mengen. (Siehe dazu auch das Kapitel 1 von Jana Bohnstengel) Hätte es diesen Widerspruch nicht gegeben, so wäre das Paradoxon nicht entdeckt worden.
Ist es folglich möglich, dass in den bisher bewiesenen Sätzen der Mathematik auch Paradoxa stecken, die bloß noch nicht erkannt wurden, denn laut der gemachten Annahme ist der Algorithmus nicht fehlerfrei. Dann wären bis zum Entdecken eines Paradoxons viele Aussagen lediglich durch „Vertrauen in Bewährtes“ bekräftigt. Doch würde sich dann jemals eine Aussage endgültig beweisen lassen? Beim Entdecken eines Paradoxons würden sich die mathematischen Schlussregeln wieder ändern, womit sich dann auch der Algorithmus ändern würde. Alle Schlüsse könnten nur noch unter Vorbehalt betrachtet werden.[15] (S. 173 ff)

Auch wenn es keine eindeutigen Argumente gegen die Möglichkeit eines nicht fehlerfreien Algorithmus gibt, ist die Existenz eines solchen für Roger Penrose nicht plausibel.

In seinem Buch „Schatten des Geistes“ diskutiert Roger Penrose noch weitere mögliche Formen des Algorithmus; so könnte dieser z.B. nicht-erkennbar sein. Dabei ist mit nicht-erkennbar gemeint, dass die Details dieses Algorithmus über das hinausgehen, was sich in der Praxis erfassen lässt. Das bedeutet aber auch, dass es im Prinzip möglich ist, die Details anzugeben, es jedoch in der Praxis zu lange dauern würde.
Betrachtet man natürliche Zahlen n, deren Binärdarstellung aus mehr als 203 Ziffern besteht, so wäre man, wenn man mit Beginn des Urknalls angefangen hätte die natürlichen Zahlen der Reihe nach zu nennen, wobei man für das Nennen einer natürlichen Zahl die Planck-Zeit von etwa 0.5×10^{-43} Sekunden benötigt, noch nicht bei diesen natürlichen Zahlen n angekommen. Wenn nun der Algorithmus eine Anzahl von Operationen in dieser Größenordnung besitzt, so liegt es jenseits der menschlichen Möglichkeiten die Details des Algorithmus zu nennen. Wenn man darüberhinaus versuchen würde diesen Algorithmus in einen Computer zu implementieren, so wäre dies in der Praxis nicht möglich. Man müsste also einen anderen Weg finden, falls es einen nicht-erkennbaren Algorithmus gibt, ihn zu implementieren. Der einzige bleibende Weg ist der, dass der Algorithmus Schritt für Schritt aufgebaut wird und die Computer durch einbringen ihrer Lernerfahrungen beim Aufbau einer neuen Generation von Computern mithelfen. Damit würe die neueste Generation von Computern keine allein vom Menschen geschaffene mehr. Es würde so eine Art von Darwinscher Evolution entstehen, welche auch eine Erklärung für das Entstehen eines nicht-erkennbaren Algorithmus wäre. Die Möglichkeit auf diesem Wege den nicht-erkennbaren Algorithmus zu implementieren ist jedoch mit Zweifeln behaftet, weshalb laut Roger Penrose diese Annahme nicht zu dem Ziel führt Computer zu einem den Menschen ähnlichen Niveau des Denkens zu verhelfen. [15] (S. 179 ff)

Bleibt also nur die Annahmen erneut zu verändern. In seinem Buch „Schatten des Geistes“ hat Roger Penrose noch weitere veränderte Annahmen diskutiert. Dabei tauchen stets Zweifel auf, so dass er zu dem Schluss kommt, dass Denken eine rein menschliche Eigenschaft bleibt.

4.7 Schluss

Der Themenkomplex des Denkens ist ein zur Zeit in der Biologie und angrenzenden Disziplinen noch nicht vollständig erforschtes Gebiet. Endgültige Aussagen über die damit verbundenen Möglichkeiten einem Computer das menschliche Denken zu implementieren ist somit schwierig. Auf einer ganz anderen Ebene hat Roger Penrose sich mit diesem Thema befasst. Er hat auf eine mathematische Weise argumentiert, warum es seiner Meinung nach keine künstliche Intelligenz geben kann, zumindest in der Hinsicht, dass Computer das Niveau des menschlichen Denkens erlangen können.

4.8 Anhang A: Das Cantorsche Diagonalverfahren

Das Cantorsche Diagonalverfahren ist ein in der Mathematik verwendetes Beweisverfahren, dessen Kern darin besteht, sich bei einer bestimmten Auflistung auf die Diagonale zu konzentrieren, mit Hilfe derer ein neues, potentiell in dieser Auflistung vorhandenes Element zu konstruieren, welches aber als Folge der Konstruktionsvorschrift nicht in ihr enthalten ist.

Angenommen man hat drei Symbole zur Verfügung: $\star$, $\circ$, $\diamond$. Aus diesen drei Symbolen werden nun unendlich lange Ketten gebildet, z.B. wäre

$$\star \quad \circ \quad \star \quad \diamond \quad \diamond \quad \diamond \quad \cdots$$

ein Anfang einer Kette und

$$\star \quad \star \quad \star \quad \circ \quad \circ \quad \star \quad \cdots$$

ein anderer. Das Ziel ist es nun zu zeigen, dass es, unter der Annahme eine vollständige Auflistung aller unendlich vielen verschiedenen und unendlich langen Ketten zu haben, noch mindestens eine weitere, bisher noch nicht konstruierte Kette gibt. Sei also folgendes der Anfang einer vollständigen Auflistung:

$$\begin{array}{cccccccc}
\star & \diamond & \diamond & \star & \circ & \diamond & \cdots \\
\circ & \circ & \star & \circ & \star & \diamond & \cdots \\
\circ & \star & \star & \diamond & \circ & \circ & \cdots \\
\diamond & \diamond & \star & \circ & \star & \diamond & \cdots \\
\star & \star & \star & \circ & \diamond & \diamond & \cdots \\
\circ & \circ & \star & \diamond & \star & \circ & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Als nächster Schritt wird die Diagonale dieser vollständigen Auflistung betrachtet, d.h.

$$\star \quad \circ \quad \star \quad \circ \quad \diamond \quad \circ \quad \cdots$$

Daraufhin wird eine Kette konstruiert, die sich an jeder Stelle von der Diagonalen unterscheidet, d.h. wenn in der Diagonal-Kette ein $\star$ steht, so kann in der zu konsturierenden Kette ein $\circ$ oder ein $\diamond$ stehen. Die konstruierte Kette könnte somit folgende Gestalt haben (es gibt unendlich viele Möglichkeiten diese Kette zu konstruieren, da an jeder Position zwischen zwei Elementen ausgewählt werden kann):

$$\circ \quad \diamond \quad \circ \quad \star \quad \circ \quad \diamond \quad \cdots$$

Diese neue unendlich lange Kette kann aber auf Grund der Konstruktionsvorschrift nicht in der vollständigen Liste enthalten sein, denn sie unterscheidet sich an der ersten Stelle von der ersten Kette, stimmt somit nicht mit ihr überein, an der zweiten Stelle von der zweiten Kette, stimmt somit auch mit dieser nicht überein u.s.w. Die neue Kette unterscheidet sich bezüglich jeder der Ketten dieser vollständigen Liste an mindestens einer Position. Damit ist sie nicht in der vollständigen Liste enthalten.

Die hier vorgestellte formelfreie Darstellung des Diagonalverfahrens ist nicht die übliche, sie ist aber flexibel und enthält die übliche Darstellung wie z.B. den originalen Beweis von Cantor, dass die Menge aller reellen Zahlen nicht abzählbar ist.

Im Haupttext ist es der Algorithmus A, der nicht in der vollständigen Liste der Computer-Programme, die von einer natürlichen Zahl n abhängen, steht, da er immer, wenn er endet, beweist, dass das Programm auf der Diagonalen nicht endet, also ist er nicht gleich einem Programm dieser Liste, was dann zu dem Schluss führt, dass der Algorithmus A kein Computer-Programm, das von einer natürlichen Zahl n abhängt, sein kann, da die Liste vollständig ist. Somit ist Denken nicht berechenbar.

Kapitel 5

Unmögliche Konstruktionen: Das Kohomologie-Argument von Roger Penrose

Jesco Humpola

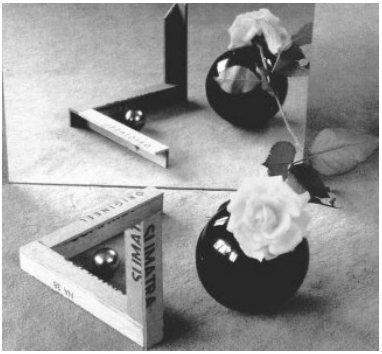

Abbildung 5.1: Penrose-tribar in der Wirklichkeit [58]

5.1 Einleitung

Abbildung 5.1 zeigt zwei verschiedene Ansichten einer Metallkonstruktion. Im Spiegel ist sie als natürliches Objekt zu erkennen, im Vordergrund zeigt sie ein Dreieck, welches widersprüchlich erscheint. Es ist noch einmal in Abbildung 5.2 zu sehen und wurde erstmals 1934 von dem schwedischen Künstler Reutersvärd erfunden. So wie dieses blieben die meisten seiner Werke der Öffentlichkeit allerdings unbekannt. 1958 zeichnete Roger Penrose, Professor am Mathematical Instiute Oxford, U. K., diese Figur erneut [35], die nach ihm „Penrose-*tribar*“ benannt wurde. Er verfasste eine mathematische Analyse [29], die dieses Kapitel des Gödel-Projekts erläutern soll.

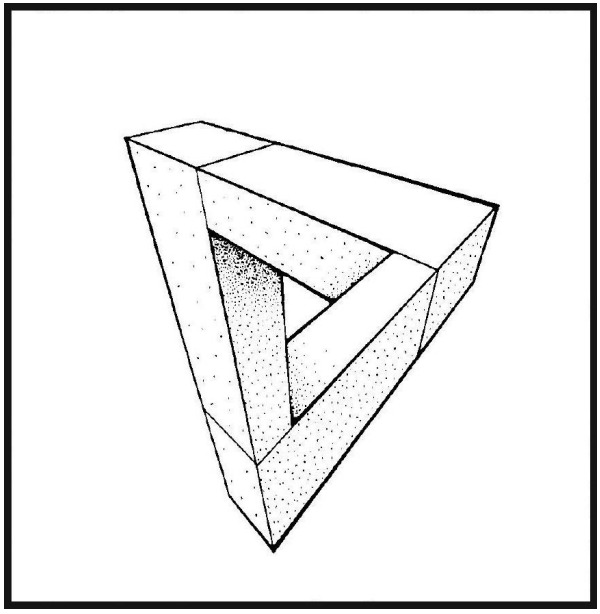

Abbildung 5.2: Penrose-„*tribar*“

Warum meinen wir, dass PENROSES *„tribar“* unmöglich ist? Das Auge des Betrachters sieht in Abbildung 5.2 lediglich ein zweidimensionales Bild, versucht aber, es dreidimensional zu interpretieren. Der Mensch will also automatisch mehr sehen, als die Abbildung zulässt.
Weiterhin ist festzustellen, dass es nur einen einzigen Raumwinkel gibt, bei dem das Metallstück aus Abbildung 5.1 kurios erscheint. Aus diesem Blickwinkel heraus wurde das Foto gemacht. Hier erscheinen die drei 90°-Winkel, aus denen das *„tribar“* besteht, gleich groß, sind jedoch unterschiedlich ausgerichtet. Sie werden im Folgenden als Objekte O_i bezeichnet.

5.2 Mathematische Beschreibung des *„tribar“*

In den Abbildungen 5.3 und 5.4 ist PENROSES Methode veranschaulicht, mit der er eine Verbindung zwischen der Geometrie des *„tribar“* und der Mathematik herstellt.

5.2.1 Gebiet und Überdeckung

PENROSE hebt das *„tribar“* wie in Abb. 5.3 zu sehen, hervor, indem er das **Gebiet Q**, in dem es enthalten ist, weiß markiert. Anschließend **überdeckt** er dieses mit den drei Teilbereichen $\mathbf{Q_1}$, $\mathbf{Q_2}$ und $\mathbf{Q_3}$, die jeweils ein einziges Objekt, nämlich einen Winkel des „tribars“, zeigen, vgl. Abb. 5.4. Es sind somit diese beiden Bedingungen erfüllt:

$$Q = \bigcup_{i=1}^{3} Q_i,\ Q_i \text{ offen}$$
$$\forall i, j \in \{1, 2, 3\} : Q_i \cap Q_j \neq \varnothing$$

Q_i soll an dieser Stelle das Objekt O_i, also den i-ten Winkel abbilden. Q_i ist quasi ein Foto von O_i.

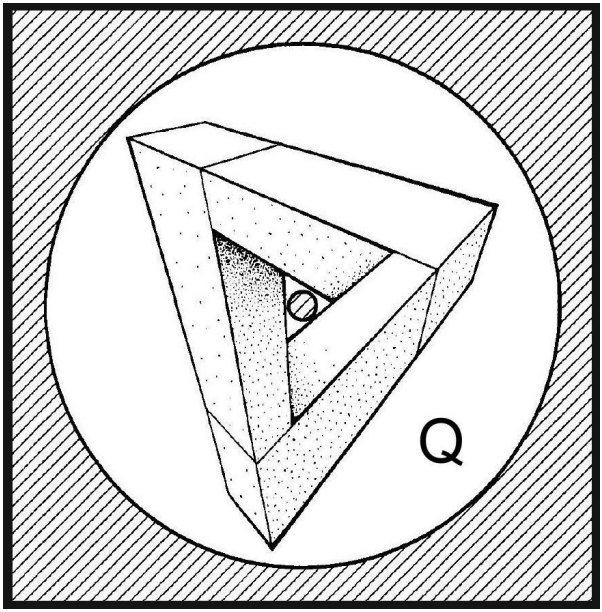

Abbildung 5.3: Gebiet Q wird weiß eingezeichnet

5.2.2 Die Verhältnisfunktion d_{ij}

Auf den Durchschnitten $Q_i \cap Q_j$ sollen jetzt Funktionen definiert werden. Betrachtet man in Abbildung 5.4 nur Q_1 und Q_2, indem man Q_3 abdeckt, so „stimmt“ das Bild. Analog können auch Q_1 bzw. Q_2 abgedeckt werden. Sind aber alle Q_i zu sehen, so „stimmt“ die Figur nicht mehr. Deshalb zeichnet PENROSE zusätzliche Punkte gemäß Abbildung 5.4.
$A \in Q_1 \cap Q_2$ sei ein Punkt. Betrachtet man nur Q_1, so ist festzustellen, dass er im Vordergrund liegt, wo er mit A_{12} bezeichnet wird. Bei Betrachtung von Q_2 befindet sich der Punkt mehr im Hintergrund und wird mit A_{21} bezeichnet.

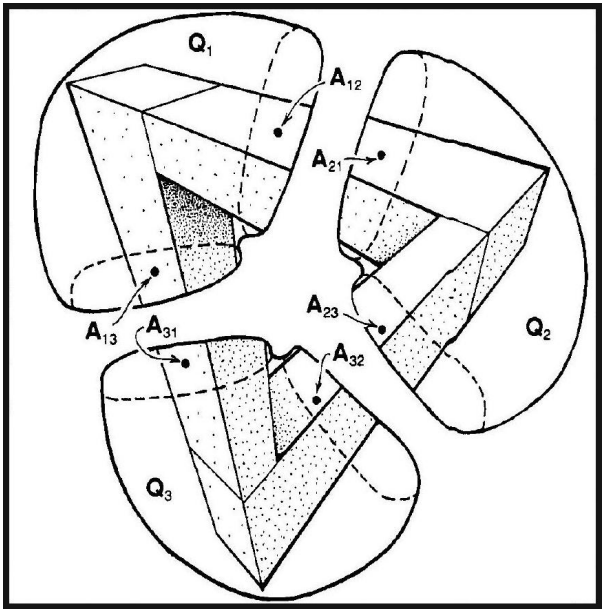

Abbildung 5.4: Überdeckungen werden gezeichnet

Allgemein entspricht $\mathbf{A_{ij}}$ also einem **Punkt $\mathbf{A \in Q_i \cap Q_j}$ aus Sicht von $\mathbf{Q_i}$** - das Auge fokussiert also erst das Objekt O_i und nimmt anschließend den Punkt A zur Kenntnis. Auf diese Weise kann es die Lage von Punkten auf den Winkeln markieren.
PENROSE definiert nun für $i, j = 1, 2, 3$, $i \neq j$ folgende Funktionen:

$$\begin{aligned} d_{ij} : & Q_i \cap Q_j \to \mathbb{R}^{>0} \\ & A \quad \mapsto \frac{\text{Abstand [Auge - Punkt } A_{ij}]}{\text{Abstand [Auge - Punkt } A_{ji}]} \end{aligned}$$

Er stellt fest, dass $d_{ij}(A)$ für jedes $A \in Q_i \cap Q_j$ den gleichen Wert

annimmt, weshalb sich die kanonische Definition ergibt:

$$d_{ij} := \frac{\text{Abstand [Auge - Punkt } A_{ij}]}{\text{Abstand [Auge - Punkt } A_{ji}]},\ A \in Q_i \cap Q_j \tag{5.1}$$

5.2.3 d_{ij} am Beispiel von Abbildung 5.1

Die Metallkonstruktion aus Abbildung 5.1 besteht offenbar aus zwei rechten Winkeln, die miteinander verschweißt wurden. Somit gilt, wenn A_{12} und A_{21} einen Punkt auf der Schweißnaht bezeichnen: $d_{12} = 1 = d_{21}$. Man hat somit gleichzeitig die Teilgebiete Q_1 und Q_2 festgelegt, welche jeweils einen Winkel enthalten. Der dritte Winkel, der in Q_3 enthalten ist, entsteht erst durch die spezielle Betrachtung, vgl. Abschnitt 5.1. Hier liegt der Grund für die Unmöglichkeit des *„tribar“* und somit muss gelten: $d_{13} \neq 1$ oder $d_{23} \neq 1$.

5.2.4 Eigenschaften von d_{ij}

Offensichtlich gilt gemäß (5.1)

$$d_{ij} = \frac{1}{d_{ji}} \tag{5.2}$$

da nur Indizes vertauscht werden.
Könnte man das *„tribar“* aus einer beliebigen Raumrichtung betrachten und würde somit seine ganze Gestalt erkennen, in Abbildung 5.1 ist dies erst durch den Spiegel möglich, so ergäbe sich eine weitere Eigenschaft der Funktion d_{ij}. Rein theoretisch kann man nämlich die Winkel bzw. Objekte O_i durch „moving in and out“, wie in Abbildung 5.5 dargestellt, bei gleichzeitigem Vergrößern bzw. Verkleinern, verschieben. Da der Betrachter seine Position beibehält, bleibt das Bild in Q_i, das das Auge wahrnimmt, gleich und man hätte künstlich produziert, was Penrose *„ambiguity*[1] *of distance“* nennt. Offenbar kann der Betrachter nicht feststellen, welchen Abstand O_i von seinem Auge hat.

[1] ambiguity, engl.: Mehrdeutigkeit

Für die Verhältnisfunktion folgt, wenn O_i bewegt wird:

$$(d_{ij}, d_{ik}) \mapsto (\lambda \cdot d_{ij}, \lambda \cdot d_{ik}),\ \lambda \in \mathbb{R}^{>0} \tag{5.3}$$

Das „moving out“ entspricht $\lambda > 1$ und „moving in“ $\lambda < 1$. Weiterhin gilt, wenn O_j bewegt wird:

$$d_{ij} \cdot d_{jk} \mapsto \frac{1}{\lambda} \cdot d_{ij} \cdot \lambda \cdot d_{jk} = d_{ij} \cdot d_{jk}$$

Analog für O_i und O_k. Aus dieser Tatsache folgert man, dass unter der Freiheit des „moving in and out“ (Abb. 5.5) folgendes mit $c = const$ gilt:

$$d_{ij} \cdot d_{jk} \cdot d_{ki} = c$$

Wichtig ist, dass der Betrachter seine Position, aus der er das *„tribar“* betrachtet, nicht ändert.

5.2.5 Wert der Konstanten c

Die Frage nach dem Wert der Konstanten c hat Penrose leider nicht beantwortet.
Um dies zu tun, ist ein Blick in die Physik der Optik nötig, nämlich die Wahrnehmung von Objekten durch das menschliche Auge. Der Effekt aus Abbildung 5.1 ist nur möglich, wenn gilt:

$$d_{ij} \cdot d_{jk} = d_{ik} \tag{5.4}$$

Denn d_{ij} steht offensichtlich für einen Wert, den das Auge umskalieren muss, wenn es beim *Abtasten* des *„tribar“* von O_i zu O_j wechselt. Anschließend wechselt es von O_j zu O_k und muss mit dem Faktor d_{jk} skalieren. Um nun wieder von O_k zu O_i zu kommen, muss das Auge, da in der Physik Skalierungsfaktoren multipliziert werden, um den Faktor $d_{ij} \cdot d_{jk}$ zurückskalieren, was Skalieren mit dem inversen Wert bedeutet. Also erhält man:

$$d_{ki} = \frac{1}{d_{ij} \cdot d_{jk}} \Leftrightarrow d_{ij} \cdot d_{jk} \cdot d_{ki} = 1$$

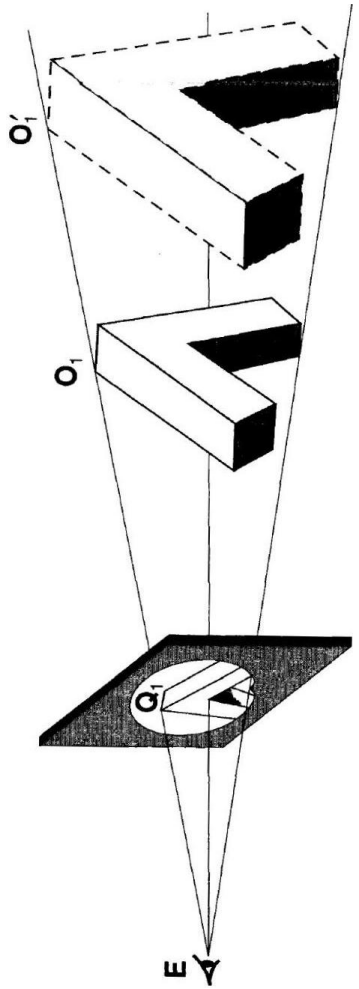

Abbildung 5.5: „moving in and out“ für O_1

5.2.6 Das Kennzeichen der „possible figure“

Nach PENROSE besteht der entscheidende Unterschied, ob das betrachtete Bild eine „impossible figure“ (Abbildung 5.1) oder eine „possible figure“, z. B. ein Warndreieck, darstellt, in dieser Aussage:

$$\text{Bild möglich} \Leftrightarrow \exists \lambda_i, \lambda_j \in \mathbb{R}^{>0} : d_{ij} = \frac{\lambda_i}{\lambda_j} \tag{5.5}$$

Diese Tatsache kann leicht verifiziert werden:
Wegen (5.2) betrachtet man den Vektor

$$\begin{pmatrix} d_{12} \\ d_{23} \\ d_{31} \end{pmatrix} \tag{5.6}$$

welcher bei einem Warndreieck dem Einheitsvektor entspricht:

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Die Winkel wären in diesem Fall 60°-Winkel und würden alle in einer Ebene liegen. Das Warndreieck ist hier als Platzhalter für eine „possible figure“ gedacht. Es kann verändert werden, indem die einzelnen 60°-Winkel wie in Abbildung 5.5 zu sehen, so manipuliert werden, dass der Betrachter immer noch das gleiche Warndreieck erkennt, der Vektor aus (5.6) aber anders aussieht. Also werden nacheinander O_1, O_2 und O_3 gemäß Abbildung 5.5 verschoben. Wegen (5.3) folgt:

$$\begin{pmatrix} d_{12} \\ d_{23} \\ d_{31} \end{pmatrix} \xrightarrow{O_1} \begin{pmatrix} \lambda_1 \cdot d_{12} \\ d_{23} \\ \frac{1}{\lambda_1} \cdot d_{31} \end{pmatrix} \xrightarrow{O_2} \begin{pmatrix} \frac{\lambda_1}{\lambda_2} \cdot d_{12} \\ \lambda_2 \cdot d_{23} \\ \frac{1}{\lambda_1} \cdot d_{31} \end{pmatrix} \xrightarrow{O_3} \begin{pmatrix} \frac{\lambda_1}{\lambda_2} \cdot d_{12} \\ \frac{\lambda_2}{\lambda_3} \cdot d_{23} \\ \frac{\lambda_3}{\lambda_1} \cdot d_{31} \end{pmatrix}$$

Somit ändert sich der Einheitsvektor durch das „moving“ folgendermaßen:

$$\begin{pmatrix}1\\1\\1\end{pmatrix} \overset{\text{„moving“}}{\longrightarrow} \begin{pmatrix}\frac{\lambda_1}{\lambda_2}\\ \frac{\lambda_2}{\lambda_3}\\ \frac{\lambda_3}{\lambda_1}\end{pmatrix} =: \vec{v}$$

Der Einheitsvektor repräsentiert eine „possible figure“. Da der Betrachter, wie oben erwähnt, das „moving“ nicht bemerkt, weil sein Auge ein konstantes Bild wahrnimmt, steht auch $\vec{v}$ für eine **mögliche Figur**.

5.2.7 Anleitung zur Analyse

Zusammenfassend ist zu sagen: wenn man feststellen möchte, ob es sich bei einem Objekt um eine „possible figure“ handelt, so muss wegen (5.2) und (5.5) überprüft werden, ob folgendes gilt:

$$\begin{pmatrix}d_{12}\\d_{23}\\d_{31}\\d_{13}\\d_{32}\\d_{21}\end{pmatrix} \in B := \left\{ \begin{pmatrix}\frac{\lambda_1}{\lambda_2}\\ \frac{\lambda_2}{\lambda_3}\\ \frac{\lambda_3}{\lambda_1}\\ \frac{\lambda_1}{\lambda_3}\\ \frac{\lambda_3}{\lambda_2}\\ \frac{\lambda_2}{\lambda_1}\end{pmatrix} \text{ mit } \lambda_i \in \mathbb{R}^{>0} \right\}$$

5.3 Das Analogon der Mathematik: *Cousin-problem*

Was inspirierte Penrose, seine *„tribar“*-Figur auf diese Weise zu analysieren?

5.3.1 Garbenkohomologie

Die Garbenkohomologie befasst sich mit *Kozykeln* und *Koränder*n. Diese beiden Mengen enthalten Vektoren, deren Elemente Abbildungen von dem Gebiet Ω in eine kommutative Gruppe $\mathbb{G}$ sind.
Wenn Z die Menge der Kozykel und B die Menge der Koränder bezeichnet, dann ist die *erste kohomologische Gruppe* $H^1(\Omega, \mathbb{G})$ wie folgt definiert:

$$H^1(\Omega, \mathbb{G}) := Z(\Omega, \mathbb{G})/B(\Omega, \mathbb{G})$$

Penrose identifiziert die Menge der d_{ij} als Kozykel:

$$Z := \left\{ (d_{12}, d_{23}, d_{31}, d_{13}, d_{32}, d_{21})^t \right\}$$

Gilt (5.5), so nennt er $B \ni (d_{ij})_{i,j=1,2,3;i\neq j}$ einen Korand. Die ganze Menge B erhält er, indem er das „moving“ aus Abschnitt 5.2.6 zulässt. Sie hat die Form:

$$B := \left\{ \left(\frac{\lambda_1}{\lambda_2}, \frac{\lambda_2}{\lambda_3}, \frac{\lambda_3}{\lambda_1}, \frac{\lambda_1}{\lambda_3}, \frac{\lambda_3}{\lambda_2}, \frac{\lambda_2}{\lambda_1} \right)^t \text{ mit } \lambda_i \in \mathbb{R}^{>0} \right\}$$

Somit setzt er gleichzeitig $\Omega = Q$ und $\mathbb{G} = \mathbb{R}^{>0}$. Außerdem gibt die Garbenkohomologie den aufgestellten Bedingungen ihre Namen:

$$\begin{aligned} \text{1-Kokette:} \quad & d_{ij} = \frac{1}{d_{ji}} \\ \text{Kozyklus:} \quad & \frac{d_{ij} \cdot d_{jk}}{d_{ik}} = 1 \\ \text{Korandoperator:} \quad & f(d_{ij}) := d_{ij} \cdot \frac{d_{jk}}{d_{ik}} \end{aligned}$$

5.3.2 Mathematische Lösung

Penrose hat nun eine mathematische Antwort gefunden, ob ein Objekt – in seinem Fall ist es das *„tribar“* – „possible“ oder „impossible“ ist:

Zu diesem Objekt wird die Menge Z gemäß (5.1) bestimmt. Ist es „möglich“, gilt $Z \subset B$ und man erhält:

$$H^1(Q, \mathbb{R}^{>0}) = \{1\}$$

Andernfalls ist die Gruppe H^1 verschieden von der trivialen Gruppe $\{1\}$.

5.3.3 Cousin-Problem

Gegeben sei ein Gebiet Ω. Unter einem **Cousin-Problem** auf Ω verstehen wir eine offene Überdeckung $\{U_i\}_i$ von Ω und eine Familie gewisser Funktionen $f_{ij} : U_i \cap U_j \to \mathbb{C}$ für alle i und j mit $U_i \cap U_j \neq \varnothing$, so dass für alle i, j und k, bei denen das Sinn macht, die beiden Bedingungen

$$\begin{aligned} f_{ij}(z) &= -f_{ji}(z) \quad && \text{für} \quad z \in U_i \cap U_j \\ f_{ij}(z) + f_{jk}(z) &= f_{ik}(z) \quad && \text{für} \quad z \in U_i \cap U_j \cap U_k \end{aligned} \tag{5.7}$$

gelten. Wir sagen, dass das Cousin-problem lösbar ist, wenn es eine Familie von Funktionen $f_i : U_i \to C$ gibt, so dass für alle i und j die Beziehung

$$f_{ij}(z) = f_i(z) - f_j(z) \text{ für } z \in U_i \cap U_j \tag{5.8}$$

erfüllt ist. [49]

5.3.4 Analyse der Analogie

Das kommt uns bekannt vor. Ein Vergleich mit der Analyse des *„tribar“* liefert:

$$\begin{aligned} \Omega &= Q \\ U_i &= Q_i \\ f_{ij} &\triangleq d_{ij} \\ f_i &\triangleq \lambda_i \\ \mathbb{C} &\triangleq \mathbb{R}^{>0} \end{aligned}$$

Ersetzt man die Addition und Subtraktion durch Multiplikation und Division, und ändert die Inversenbildung entsprechend, so folgt aus (5.7):

$$f_{ij}(z) = \frac{1}{f_{ji}(z)} \text{ für } z \in U_i \cap U_j$$
$$f_{ij}(z) \cdot f_{jk}(z) = f_{ik}(z) \text{ für } z \in U_i \cap U_j \cap U_k$$

Dies entspricht (5.2) und (5.4). Aus (5.8) ergibt sich:

$$f_{ij}(z) = \frac{f_i(z)}{f_j(z)} \text{ für } z \in U_i \cap U_j$$

Dies entspricht (5.5). Das Cousin-Problem nun, lässt sich genau dann lösen, wenn $H^1(\Omega, \mathbb{C}) = \{1\}$ ist.

Roger Penrose nennt das Cousin-Problem nicht, kannte es aber vermutlich. Er definierte entsprechende Funktionen (vgl. Abschnitt 5.2) und kam zu dem verblüffend übersichtlichen Ergebnis:

„possible figure“ $\Leftrightarrow H^1(Q, \mathbb{R}^{>0}) = \{1\}$

Kapitel 6

Logische Tiefe von Programmen: Die Analyse von Charles Bennett

Sören Dobberschütz

6.1 Einleitung

6.1.1 Die Tiefe der größten Primzahl

Jeder Mathematiker kennt Sätze, die ihm intuitiv als „flach“ erscheinen, während andere eine große Komplexität besitzen.

Betrachten wir beispielsweise die folgende Aussage:

> *Es gibt unendlich viele Primzahlen.*

Auch wenn der **Beweis** sehr bekannt ist, sei er an dieser Stelle nochmals kurz angeführt:

Angenommen, es gibt nur endlich viele Primzahlen; betrachte dann die Menge

$$P := \{p_1, p_2, \ldots, p_n\}$$

aller Primzahlen.

Für die natürliche Zahl

$$p := (p_1 \cdot p_2 \cdot \ldots \cdot p_n) + 1$$

gilt dann, dass p durch p_1 bis p_n nur mit Rest 1 geteilt werden kann (aufgrund der Primfaktorzerlegung gilt dies auch für jede andere Zahl), also selber eine Primzahl ist, die nicht in P enthalten ist. ↯

Eine andere Möglichkeit wäre zu zeigen, dass zu jeder natürlichen Zahl n eine Primzahl p mit $p > n$ existiert.

Betrachte dazu

$$p := n! + 1$$

Diese Zahl p ist durch die Zahlen $2, \ldots, n$ jeweils nur mit Rest 1 teilbar. p ist also (außer durch 1) ohne Rest nur durch Zahlen teilbar, die größer als n sind. Folglich ist entweder p selbst eine Primzahl, oder einer seiner Primteiler ist größer als n. □

Auch wenn dieser Satz eine überaus wichtige Aussage darstellt und weitreichende Konsequenzen für die Mathematik und deren Anwendungen nach sich zieht, so ist er doch in seinem Inhalt und seinem Beweis recht „einfach“; im Gegensatz zu folgendem Beispiel:

> **Satz über die implizite Funktion:** *Seien E_1, E_2 sowie F Banachräume. Sei $D_f \subset E_1 \times E_2$ offen, $k \in \mathbb{N}$, $f \in \mathcal{C}^k(D_f, F)$, $(x_0, y_0) \in D_f$ und $c_0 := f(x_0, y_0)$*
>
> *Setze*
>
> $$Df(x,y)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \underbrace{Df(x,y)\begin{pmatrix} x_1 \\ 0 \end{pmatrix}}_{=:D_1 f(x,y)(x_1)} + \underbrace{Df(x,y)\begin{pmatrix} 0 \\ x_2 \end{pmatrix}}_{=:D_2 f(x,y)(x_2)}$$

Gilt dann

$$D_2 f(x_0, y_0) \text{ ist invertierbar,}$$

so existieren offene Umgebungen $U \subset D_f$ *von* (x_0, y_0) *und* $U_1 \subset E_1$ *von* x_0 *sowie ein* $g \in \mathcal{C}^k(U_1, E_2)$ *so dass*

$$(x, y) \in U \text{ und } f(x, y) = c_o \iff x \in U_1 \text{ und } y = g(x)$$

Dieser Satz erscheint (auch abgesehen von der Länge) ungleich komplexer und tiefer als der Satz über die Anzahl der Primzahlen. Was kann der Grund für dieses Gefühl sein?
Während der Satz über die Anzahl der Primzahlen sich mit Hilfe von grundlegenden Eigenschaften natürlicher Zahlen beweisen lässt, wie z.B. der Primfaktorzerlegung, der Multiplikation, der Division mit Rest und einer einfachen Annahme, so erfordert der Satz über die implizite Funktion weitere Sätze und Hilfsmittel wie den Banachschen Fixpunktsatz, die Neumannsche Reihe sowie Mittelwertsätze und andere Ergebnisse der Differentialrechnung. Diese sind viel verborgener in der Welt der Mathematik als beispielsweise die Grundrechenarten und erfordern wiederum andere Sätze zu ihrem Beweis.

Wie kann man nun versuchen, die „logische Tiefe“ eines Objektes zu beschreiben oder gar zu „messen“? Im Folgenden wird ein Ansatz vorgestellt, der die Tiefe mit Hilfe von Turing-Maschinen zu erfassen versucht (vgl. [1]).

Statt mathematischer Sätze werden dabei Zahlenfolgen verwendet, die eine Maschine ähnlich einem Computer interpretiert und damit Ausgaben erzeugt. Und statt des Weges von einfachen Axiomen hin zu komplexen Sätzen wird der Weg von Programmen zu bestimmten Ausgaben verfolgt. Am Beispiel der Primzahlen erkennt man, dass ein mathematischer Satz möglicherweise mehrere Beweise besitzt; und so werden wir auch im Folgenden manchmal verschiedene Programme betrachten, die dieselbe Ausgabe erzeugen.

6.1.2 Tiefe vs. Informationsgehalt

Die logische Tiefe eines Objekts darf nicht als Informationsgehalt verstanden werden, sondern vielmehr als Maß[1] für den Wert der Informationen [1]:
Die Verteilungsfunktion der Normalverteilung ist in Tabellen vertafelt; dabei handelt es sich um die Werte der Funktion

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2}t^2} \, \mathrm{d}t$$

für verschiedene x aus den reellen Zahlen.
Solch eine Tabelle besitzt zweifelsfrei einen hohen Informationsgehalt; wie steht es aber mit dem Wert dieser Information? Es handelt sich bei genauerer Betrachtung der Tabelle nur um ein Hilfsmittel, entstanden aus Bequemlichkeit. Obige Funktion trägt alle Informationen der Tabelle in sich – man benötigt aber weitere Hilfsmittel (wie Verfahren zur Bestimmung des Integrals), um die Tabelle wieder aus der Funktion hervorzubringen.
Somit ist der Wert der Tabelle geringer als der von $\Phi(x)$.

Wie man an diesem Beispiel erkennt, liegt der Wert einer Information in Zusammenhängen verborgen, die nicht leicht einzusehen sind und ein Verknüpfen von weiteren Informationen erfordert, um sie wieder „freizulegen“.

Um im Folgenden die logische Tiefe angemessen definieren zu können, sind zunächst einige Begriffsbildungen erforderlich, die im nächsten Abschnitt vorgestellt werden.

[1] In diesem Text wird mit „Maß“ nicht der übliche mathematische Sachverhalt aus der Maß- und Integrationstheorie bezeichnet, sondern vielmehr ein Konstrukt zur Quantifizierung bestimmter Sachverhalte.

6.2 Grundlegende Begriffe

Wir möchten, dass die Begriffe, die wir im Folgenden definieren, möglichst unabhängig von der „Form“ der Definition sind.
Beispielsweise kann man bei der Formulierung mathematischer Sätze wie der in Abschnitt 6.1 zwischen einer sehr strengen, formalen Formulierung mittels abstrakter Zeichen oder einer wortreichen Formulierung wählen. Wenn der Inhalt gleich ist, ist es auch wünschenswert, dass alle hier eingeführten Begriffe anwendbar sind.
Zu diesem Zweck wählen wir eine ganz bestimmte Darstellung, in die alle Sachverhalte transformiert werden können – nämlich die Darstellung mit Hilfe einer Turing-Maschine:

6.2.1 Die verwendete Turing-Maschine

Eine genauere Beschreibung einer Turing-Maschine erfolgte in Kapitel 2 von Lena Borgmann.

Die im Folgenden verwendete Maschine besitzt folgende Komponenten:

- Ein „Schreib-Lese-Band“; ein Speicherband, auf das Daten geschrieben und von dem Daten gelesen werden können. Auf diesem Band erfolgt auch die Ausgabe, die das Programm erzeugt.
- Ein „Lese-Band“; ein Speicherband, von dem nur gelesen werden kann und auf dem sich das Programm befindet, das ausgeführt wird.

Alle Daten liegen in binärer Form vor, d.h. als eine Folge von 0 und 1. Solch eine Folge wird auch als „string“ bezeichnet, und ihre einzelnen Komponenten (die 0 und 1) nennt man „bit“. Die Länge eines strings x kürzen wir mit $|x|$ ab; die Verkettung der strings y und z mit yz.

Um beschreiben zu können, was bei einer Berechnung passiert, führen wir folgende Notation ein:
Mit U bezeichnen wir eine bestimmte Turing-Maschine; dazu sei p das

Programm auf dem „Lese-Band“ und w der Inhalt des „Schreib-Lese-Bandes“.

$$U(p, w) = x$$

soll nun anzeigen, dass eine Turing-Maschine in obiger Konfiguration nach Ausführung des Programms (d.h. nach Verwendung des *gesamten* strings p) mit der Ausgabe x anhält. Liegen keine Daten w auf dem „Schreib-Lese-Band“ vor, so schreiben wir abkürzend $U(p) = x$.

Die Ausgabe eines Programmes sagt natürlich noch nichts darüber aus, wie lange die Maschine für sie benötigte. Daher wollen wir auch die „Zeit“ messen, die für die Ausführung nötig war und bezeichnen diese mit

$$T(p, w) \qquad \text{bzw.} \qquad T(p)$$

(die Zeit wird dabei nicht in Sekunden oder Minuten gemessen, sondern in Takteinheiten, die die Turing-Maschine rechnet – wobei sich der eine Wert immer aus dem anderen errechnen lässt).

Natürlich kann es auch Programme $p^{\#}$ geben, die nicht nach endlicher Zeit beendet werden. In diesem Fall soll $U(p^{\#})$ bzw. $U(p^{\#}, w)$ undefiniert bleiben. Gleiches gelte für den Fall, dass nicht der gesamte Programmstring gelesen wurde.

Welche Möglichkeiten kann man sich vorstellen, schneller zu einer bestimmten Ausgabe zu gelangen? Man könnte ein anderes Programm verwenden, das auf einer anderen Maschine U' ausgeführt wird.
Hier gehen wir jedoch davon aus, dass jede Turing-Maschine jede andere simulieren kann, indem dem eigentlichen Programm ein „Simulationsstring“ s vorangestellt wird. Simuliert nun U' ein Programm p für U, so ergibt sich also

$$U(p) = U'(sp)$$

Dabei nehmen wir an, dass die Programmausführung mit Simulation nur einen festen Wert K länger dauert, der unabhängig von der Länge des eigentlichen Programms ist (also $T'(sp) = T(p) + K$ für alle Programme p).

Manchmal wäre es gut, die Ausgabe eines Programms als Eingabe in einem anderen zu nutzen; dafür soll es einen string r geben, der die Hintereinanderausführung von Programmen ermöglicht. Mit Hilfe unserer Bezeichnungen erhalten wir

$$U(p, w) = x \text{ und } U(q, x) = y \quad \Longrightarrow \quad U(rpq, w) = y$$

6.2.2 Die Menge aller haltenden Programme

Wir können nun die Menge $\mathcal{U}$ aller Programme, die nach endlicher Zeit anhalten, betrachten:

$$\mathcal{U} := \{p \mid U(p) \text{ ist definiert}\}$$

Diese Menge erfüllt die Eigenschaft einer sogenannten *Präfix-Menge*, d.h. dass kein Element die Fortsetzung eines anderen ist.
Im Fall von $\mathcal{U}$ kann kein Element ein anderes fortsetzen, da für die Programme p und q die Turing-Maschine beim Versuch der Ausführung von pq schon nach dem „p-Teil“ anhält. Beispielsweise können somit 00 und 001 nicht gleichzeitig Elemente in $\mathcal{U}$ sein.

Für jede Präfix-Menge S gilt die sogenannte *Kraft-Ungleichung*

$$\sum_{p \in S} 2^{-|p|} \leq 1$$

(siehe [12]).

6.2.3 Minimalprogramme und algorithmische Zufälligkeit

Wie schon erwähnt, kann es mehrere Programme geben, die dieselbe Ausgabe x erzeugen. Das kürzeste davon wird mit x^* bezeichnet und heißt *Minimalprogramm von* x; also

$$x^* := \arg\min\{|p| : U(p) = x\}$$

Dies kann man auch definieren, falls zusätzliche Daten w auf dem Schreib-Lese-Band vorliegen; in diesem Fall heißt analog

$$(x/w)^* := \arg\min\{|p| : U(p, w) = x\}$$

Minimalprogramm von x relativ zu w. (Dabei bezeichnet $\arg\min\{|p| : U(p) = x\}$ das Programm p – sofern es existiert –, das als Ausgabe x erzeugt und unter allen solchen Programmen die kürzeste Länge besitzt.)

Die Anwesenheit der zusätzlichen Informationen w kann ein Programm p zur Ausgabe von x unter Umständen verkürzen; andererseits kann diese Information das Programm nicht beliebig verlängern, da wir annehmen, dass w vor der Ausführung von p durch einen string konstanter Länge gelöscht werden kann.

Wir nennen nun einen string x *kompressibel um s bit*, wenn sein Minimalprogramm um mindestens s bit kürzer ist als er selbst, d.h.

$$|x^*| \leq |x| - s$$

Analog nennen wir einen string *kompressibel um s bit relativ zu w*, wenn

$$|(x/w)^*| \leq |x| - s$$

Inwieweit ist nun das Minimalprogramm x^* kompressibel? Dieses kann nur um eine feste Konstante (abhängig von der Turing-Maschine) kürzer sein als x^* – denn angenommen, $(x^*)^*$ wäre sehr viel kürzer als x^*, so wäre ein Programm der Form „führe das Ergebnis von $(x^*)^*$ als Programm aus“ das Minimalprogramm zu x.

Ausgehend von diesen Überlegungen nennt man einen endlichen string *algorithmisch zufällig*, wenn er fast inkompressibel ist (d.h. höchstens um eine nur von der Turing-Maschine abhängige Konstante kompressibel).[2] Diese Definition kann man auf unendlich lange strings erweitern, indem man in diesem Fall fordert, dass alle „Anfangsstücke“ (also die ersten n bit für alle $n \in \mathbb{N}$) algorithmisch zufällig sind.

[2] Zu jedem $n \in \mathbb{N}$ gibt es mindestens einen inkompressiblen string der Länge n: Es gibt 2^n strings der Länge n, aber nur $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ mögliche „kürzere“ Beschreibungen. Es muss also mindestens einen string x der Länge n geben, der nicht kompressibel ist.

6.2.4 Algorithmische Wahrscheinlichkeit

Wir betrachten nun folgendes Experiment:

Wir lassen eine Turing-Maschine eine Berechnung mit einer zufälligen Eingabe durchführen. Wie groß ist die Wahrscheinlichkeit, dass sie mit einer bestimmten Ausgabe x hält? Dazu kann man alle strings betrachten, die x als Ausgabe erzeugen. Die Wahrscheinlichkeit, dass sich an einer bestimmten Stelle im string eine 0 befindet, beträgt $\frac{1}{2}$ (analog für 1); somit ergibt sich für den gesamten string x mit der Länge $|x|$ die Wahrscheinlichkeit $(\frac{1}{2})^{|x|} = 2^{-|x|}$. Davon ausgehend nennt man

$$P(x) = \sum_{\{p:\ U(p)=x\}} 2^{-|p|}$$

die *algorithmische Wahrscheinlichkeit von x* (vgl. dazu auch Abschnitt 3.3 im Kapitel 3 von Janina Ried).

Das Wissen darüber, dass ein Programm p die Ausgabe x erzeugt, liefert noch keine Aussage darüber, welche Zeit dafür benötigt wird.
Daher definiert man analog die *zeit-beschränkte algorithmische Wahrscheinlichkeit von x* als die Wahrscheinlichkeit, für eine zufällige Eingabe innerhalb der Zeit t die Ausgabe x zu erzeugen:

$$P_t(x) = \sum_{\substack{\{p:\ U(p)=x,\\ T(p)\leq t\}}} 2^{-|p|}$$

(Analog betrachte man obige Konstruktionen für die algorithmische Wahrscheinlichkeit von x relativ zu w, $P(x/w)$ bzw. $P_t(x/w)$.)

Mit diesen Begriffen können wir nun versuchen, uns an die Tiefe eines strings anzunähern:

6.3 Logische Tiefe

6.3.1 Wie kann man logische Tiefe definieren?

Wir können nun versuchen, die logische Tiefe eines strings zu definieren. (Einen string, der nicht tief ist, bezeichnen wir im Folgenden als „seicht“.)

Wann könnte ein string tief sein? Eine Idee wäre, dass er um so tiefer ist, je länger die Turing-Maschine zur Berechnung braucht, etwa in der Form:

> Die logische Tiefe eines strings x ist die Zeit, die sein Minimalprogramm x^* zur Ausführung benötigt.

Dabei ergibt sich ein mögliches Problem: Unter Umständen existiert ein anderes Programm, dass x viel schneller berechnet, selbst aber nur um einige wenige bit länger ist als x^*. Dadurch, dass im Minimalprogramm alle Redundanzen „eliminiert“ wurden, benötigt dieses mehr Zeit zur Ausführung. Wenn jedoch ein Programm logisch tief ist, so wird es wenige dieser Redundanzen geben, und auch nicht-minimale Programme werden lange zur Berechnung benötigen. Daher kann man die erste Idee verfeinern, indem man zusätzlich ein „Signifikanzniveau“ einführt und zu „fast-minimalen“ Programmen übergeht:

> Die logische Tiefe zum Signifikanzniveau s eines strings x ist die minimale Zeit, die ein Programm zur Berechnung von x benötigt, das nicht länger als s bit als das Minimalprogramm ist.

In einem weiteren Schritt wollen wir nun die Definition der logischen Tiefe so ändern, dass diese unabhängig vom Minimalprogramm wird. Dazu definieren wir nun letztendlich:

> **Logische Tiefe:** *Die logische Tiefe $D_s(x)$ eines strings x zum Signifikanzniveau s ist definiert als die kürzeste Zeit die*

benötigt wird, x durch ein s-inkompressibles Programm zu erzeugen:

$$D_s(x) := \min \{T(p) : U(p) = x \ \wedge \ (|p| - |p^*|) < s\}$$

Durch $D_s(x)$ wird die Redundanz in jedem Programm, das x ausgibt, gemessen. Wie oben beschrieben ist dies ein Indikator für die logische Tiefe.

Analog kann man die logische Tiefe von x relativ zu w definieren als

$$D_s(x/w) := \min \{T(p, w) : U(p, w) = x \ \wedge \ (|p| - |(p/w)^*|) < s\}$$

Ausgehend von diesen Definitionen wird in [1] gezeigt, dass die Tiefe eines strings nicht von der verwendeten Turing-Maschine abhängt, und dass sich tiefe Objekte nicht leicht aus seichten erzeugen lassen – weder durch einen vorgegebenen Prozess noch durch zufällige Operationen.

Betrachten wir nun einige Beispiele (vgl. [1]):

6.3.2 Beispiele

Zunächst ein Beispiel für ein **seichtes Objekt**:

Wir betrachten den string

$$\underbrace{0000\ldots00}_{n \text{ Stück}} =: 0^n$$

Intuitiv würde man diesen string als seicht klassifizieren. Das einzige mögliche Problem dabei ist die Zahl n: Ist n selbst tief, so ist auch 0^n tief (da dann das Abbruchkriterium „Breche das Schreiben von '0' nach n Stück ab“ schwierig zu erzeugen ist).
Gibt man der Turing-Maschine jedoch das n vor und betrachtet die logische Tiefe relativ zu n, so ist dieser string in der Tat seicht.

Des weiteren ist auch eine zufällige Zahlenfolge seicht (ein mögliches Programm kopiert mehr oder minder die Folge und hält nach deren Länge an).

Erstaunlicherweise ist auch die in Abschnitt 3.3 beschriebene Haltewahrscheinlichkeit Ω seicht (vgl. [1]). Anschaulich ist die Information in Ω derart komprimiert enthalten, dass Ω selbst zufällig erscheint.

Tiefe Objekte sind, wie oben erwähnt, schwierig aus seichten zu erzeugen. Mit Hilfe von Cantors Diagonalargument ist dies jedoch durchaus möglich (vgl. dazu [1]).

Ein weiteres Beispiel wäre ein Programm der Form „Finde alle n bit lange strings x mit

$$P_t(x) \geq 2^{-n},$$

(wobei t sehr groß vorgegeben wird) und gib den ersten string aus, der nicht in dieser Menge ist“ (vgl. dazu ebenfalls [1]).

6.3.3 Ein Maß für Tiefe

Etwas unbefriedigend an den obigen Definitionen ist das Signifikanzniveau s. Ist es möglich, auch unabhängig von s ein „Maß“ für die logische Tiefe eines Objekts anzugeben?

Dazu könnte man vom Mittel der Ausführungszeiten aller Programme ausgehen, die x erzeugen. Dies liefert jedoch kein brauchbares Ergebnis, da Programme dominieren, die bei der Ausführung viel Zeit „verschwenden“ und deutlich länger zur Berechnung von x benötigen als sein Minimalprogramm.
Jedoch kann man die langsamen Programme „bestrafen“, damit die schnelleren Programme das Maß dominieren. Dies führt auf den Ausdruck

$$D(x) = \frac{\sum\limits_{\{p:\ U(p)=x\}} 2^{-|p|}}{\sum\limits_{\{p:\ U(p)=x\}} \frac{2^{-|p|}}{T(p)}}$$

als *mittlere Tiefe eines strings* x.

Anhang

Abbildung 6.1: Charles Bennett [54]

Die in diesem Abschnitt vorgestellten Ideen wurden um 1988 von Charles H. Bennett entwickelt und in [1] beschrieben.

Bennett wurde 1943 in den USA geboren. Zunächst studierte er Chemie und wandte sich dann in seiner Promotion der Simulation von chemischen Reaktionen zu.
1972 ging er zu IBM Research und untersuchte das Verhältnis von Physik und Informationstheorie. Im Zuge dieser Tätigkeit entwickelte er auch den Begriff der „logischen Tiefe“.
Mittlerweile beschäftigt er sich mit Quantencomputern sowie mit Quanteninformationen und deren Übermittlung.

Bennett ist verheiratet und hat drei Kinder. [55]

Kapitel 7

Können Computer über sich nachdenken? Ein Modell für Selbstbewusstsein

Ulrich Krause

7.1 Denken und Reflexion

Bei den folgenden Ausführungen handelt es sich um eine erweiterte Fassung des letzten Teils meines Vortrags „Gödel und die Grenzen des Wissens“ vom 28. April 2006 im Haus der Wissenschaft in Bremen.

Im Denken kann die Außenwelt reflektiert werden – und auch die Innenwelt, überraschenderweise. Gedanken können in weiteren Gedanken reflektiert werde. Wie die Beiträge dieses Bandes belegen findet sich dieses selbstbezügliche Moment sowohl in den Paradoxa von Epimenides, Russel und anderen als auch in den Untersuchungen von Gödel, Turing, Chaitin, Penrose und Bennett.

Der Prozess der Reflexion ist insbesondere zentral für mathematisches Denken. Für das historisch gewachsene *Gebäude der Mathematik* war es ein wesentlicher Prozess, dass auf den Schultern vorangegangener Generationen Reflexionen auf immer höherer Stufe stattfanden. Die sprichwörtliche *Abstraktheit* der modernen Mathematik ist gewissermaßen ein notwendiger Ausdruck der entwickelten Reflexionstiefe. Beispielsweise spielt bei dem Übergang von einem bestimmten Axiomensystem zu einem neuen Axiomensystem die Reflexion des alten Systems eine Rolle, indem etwa Axiome variiert und neue Zusammenhänge zwischen ihnen assoziiert werden. Penrose scheint etwas Ähnliches im Auge zu haben, wenn er schreibt:

> „Once we have perceived how to encapsulate a body of sound mathematical insights into a computational procedure, then this very perception provides us with a *new* mathematical insight, as trustworthy as those insights that we have incorporated into our computational procedure, but which is not itself incorporated."
>
> ([14, S. 40]).

Dieser Prozess der wiederholten Reflexion generiert schrittweise neue Einsichten und ist per se nicht abgeschlossen, sondern offen. Das wird drastisch exemplifiziert durch *Cantors Diagonalargument* (siehe z. B. Kapitel 4 von Ann-Kristin Petersen), wo für jede vorgegebene Liste von Symbolfolgen sich durch „Reflexion" der Diagonalen eine neue Symbolfolge finden lässt, die nicht in der Liste enthalten ist. Trotz dieser Offenheit des Reflexionsprozesses unterscheidet man üblicherweise nicht zwischen einem „Menschen", einem „Menschen, der über sich nachdenkt", einem „Menschen, der sein Nachdenken reflektiert" usw. Im Gegenteil, man betrachtet einen Menschen mit all seinen möglichen Reflexionen irgendwie als eine Einheit, seine Reflexionen als Teil des Bewusstseins seiner selbst, seines *Selbstbewusstseins* oder auch seines „ICH". Diese Vorstellung einer Einheit ist alles andere als selbstverständlich, wenn man die Offenheit des Reflexionsprozesses ernst nimmt. Das Problem, das hier

auftaucht, lässt sich auch pointiert so formulieren: Wenn ich *über mich* nachdenke, so ist dies *mein* Gedanke. Man kann sich das vorliegende Problem auch dadurch veranschaulichen, dass man versucht gedanklich *einen* Computer zu konstruieren, der sich sukzessive selbst reflektiert.

Dieses Problem der Einheit des Reflexionsprozesses will ich im folgenden analysieren. Dazu ist zunächst eine klare Formulierung des Begriffs der Reflexion erforderlich. Darauf aufbauend werde ich ein analytisches Modell für „Selbstbewusstsein“ entwickeln. Dieser Rahmen erlaubt es dann die angesprochene Einheit des Reflexionsprozesses vermöge einer finitären Neuformulierung des Diagonalarguments zu begründen.

7.2 Was sind Reflexionen?

Eine (gedankliche) Reflexion bezieht sich auf einen gewissen Gegenstandsbereich und stellt eine spezifische Erweiterung dieses Bereiches dar, die wir im folgenden genauer formulieren wollen. Z.B. ist es gewöhnlich ein unbewusster Akt, wenn ich Nahrung zu mir nehme, ein Akt, den ich etwa in der Form „Ich esse“ bewusst reflektieren kann. Auch diese Reflexion kann mir bewusst werden und zu erneuter Reflexion führen. Es ist daher sinnvoll, von Stufen der Reflexion zu sprechen, wobei der bloße Akt des Essens als 0. Stufe figuriert, „Ich esse“ als 1. Stufe, die Reflexion darauf als 2. Stufe usw. Diesen Prozess fortgesetzter Reflexion wollen wir später näher untersuchen. Zunächst aber stellt sich die Frage, wie sich das charakteristische Moment eines einmaligen Prozesses der Reflexion klar formulieren lässt. Dazu bezeichne M den Gegenstandsbereich, d.h. die Menge der Objekte, die wir als mögliche Gegenstände einer Reflexion ansehen wollen. Durch eine Reflexion richtet sich die Aufmerksamkeit auf einige Objekte, die als „interessant“, „wichtig“ o.ä. betrachtet werden, im Unterschied zu anderen Objekten für die das nicht gilt. Diesen Prozess einer Reflexion, genauer gesagt, einer *primitiven Reflexion*, können wir formal durch eine Funktion beschreiben, die Objekten, denen besondere Aufmerksamkeit gilt, den Wert 1 zuschreibt und allen anderen Objekten der Menge M den Wert 0. Auf diese Weise wird im Gegenstandsbereich

M eine Teilmenge T von Objekten ausgezeichnet, nämlich die Menge aller Objekte, die den Wert 1 erhalten. Die genannte Funktion heißt die charakteristische Funktion der Teilmenge T.

Wir wollen uns hier auf diese primitiven Reflexionen beschränken. Für nichtprimitive Reflexionen würde man statt der Aufmerksamkeitswerte 0 und 1 eine größere Menge von gestuften Aufmerksamkeitswerten zugrunde legen. Im folgenden bezeichne $R(M)$ die Menge aller primitiven Reflexionen des Gegenstandsbereiches M. Aus dem Cantorschen Diagonalargument ergibt sich, dass die Reflexionen eines Gegenstandsbereiches diesen übersteigen. Das lässt sich auch folgendermaßen formulieren. Identifizieren wir ein Objekt von M mit der Funktion, die genau für dieses Objekt den Wert 1 und sonst den Wert 0 hat, also mit der charakteristischen Funktion dieses Objektes, so können wir in der Sprache der Mengen sagen, dass M eine echte Teilmenge von $R(M)$ sein muss.

Obwohl beim Begriff der primitiven Reflexion hier in erster Linie an eine gedankliche menschliche Reflexion gedacht ist, lässt sich dieser Begriff auch durchaus auf einfache Organismen oder auf Computer bzw. Roboter beziehen. Für einen einfachen Organismus könnte eine primitive Reflexion darin bestehen, das Umfeld in Dinge die fressbar sind und solche, die es nicht sind, einzuteilen. Für einen beweglichen Roboter könnte eine primitive Reflexion darin bestehen, sein Umfeld einzuteilen in die Dinge, die vor ihm liegen und diejenigen, die hinter ihm liegen. Primitive Reflexionen können durchaus berechenbar sein, für Organismen etwa durch im Gehirn oder den Genen verankerte Algorithmen und für Roboter etwa durch programmierte Algorithmen. Für höhere Organismen sind höhere Stufen der Reflexion möglich, Reflexionen von Reflexionen etwa, wo die primitiven Reflexionen selbst zum Gegenstandsbereich werden.

Das die „computational procedure“ transzendierende Moment liegt bereits in dessen Reflexion („this very perception“), der wir ebenso vertrauen wie den Einsichten „that we have incorporated into our computational procedure.“ Natürlich kann diese Reflexion auch wieder Gegenstand einer „computational procedure“ sein, an die sich dann wieder eine Reflexion anschließen mag, usw.

Für das folgende halten wir also fest, dass eine (primitive) Reflexion eine Aufmerksamkeits- bzw. Bedeutungszuweisung ist, die den jeweiligen Gegenstandsbereich übersteigt und die sich schrittweise fortsetzen lässt, ohne ein vorbestimmtes Ende zu haben.

7.3 Ein analytisches Modell für „Selbstbewusstsein“

Bezeichne also M einen beliebigen Gegenstandsbereich und $R(M)$ die *Menge aller primitiven Reflexionen von M.*

Da sich Reflexionen auch auf Reflexionen beziehen können, können wir die Menge aller primitiven Reflexionen von $R(M)$ betrachten, d.h. die Menge $R\Big(R(M)\Big)$. Der Prozess der Reflexion lässt sich fortsetzen, und wir erhalten als *Menge aller primitiven Reflexionen n-ter Stufe von M* die Menge $R\Big(R(\cdots R(M)\cdots\Big)$, wobei die Operation R der Reflexion n-mal ausgeführt wird. Diese Menge bezeichnen wir kurz mit $R^n(M)$. Den Gegenstandsbereich selbst fassen wir als Reflexionen 0-ter Stufe auf, $M = R^0(M)$. Durch sukzessive Identifikation und Cantors Diagonalargument erhalten wir eine aufsteigende Kette wiederholter Reflexionen, $M \subsetneqq R(M) \subsetneqq R^2(M) \subsetneqq R^3(M) \subsetneqq \cdots$. Diese Kette besitzt einen Grenzwert $\mathcal{M}^*$, nämlich die Menge der Reflexionen aller Stufen, also $\mathcal{M}^* = \bigcup_{n=0}^{\infty} R^n(M)$. Nun kann auch der Bereich $\mathcal{M}^*$ wieder Gegenstand von Reflexionen sein, und es stellt sich die Frage, ob sich diese Reflexionen in $\mathcal{M}^*$ wiederfinden lassen oder nicht. In dem hier gewählten Rahmen ist dies die Frage nach der Einheit des reflektierenden Subjekts. Die Antwort ist negativ, denn nach Cantors Diagonalargument ist $\mathcal{M}^* \subsetneqq R(\mathcal{M}^*)$, und zwar in dem strengen Sinn, dass sich keine eineindeutige Entsprechung (eine sogenannte Bijektion) zwischen den Elementen von $\mathcal{M}^*$ und denen von $R(\mathcal{M}^*)$ finden lässt. Wir engen nun den Begriff der Reflexionen weiter ein und betrachten für einen Gegenstandsbereich M nur solche primitiven Reflexionen von M, die nur endlich vielen Ob-

jekten besondere Aufmerksamkeit widmen. Eine solche *finitäre Reflexion* ist also eine primitive Reflexion $f\colon M \longrightarrow \{0,\ 1\}$, die nur für endlich viele Objekte von M den Wert 1 annimmt. Es bezeichne $F(M)$ die *Menge aller finitären Reflexionen des Gegenstandsbereiches* M. Für einen endlichen Bereich M ist $F(M) = R(M)$ und, da $R(M)$ wieder endlich ist, ergibt sich sukzessive, dass $F^n(M) = R^n(M)$ ist für alle Stufen n. Für einen unendlichen Gegenstandsbereich ergibt sich jedoch ein Unterschied, was von Bedeutung ist, da $\mathcal{M}^*$ naturgemäß eine unendliche Menge ist. Wie im Falle von R können wir auch den Prozess F iterieren und den Grenzwert aller Reflexionsstufen $\mathcal{M}_* = \bigcup_{n=0}^{\infty} F^n(M)$ betrachten.

Von einer Einheit des reflektierenden Subjekts bzw. von *Selbstbewusstsein* wird man dann sprechen können, wenn eine Selbstreflektion dieser Einheit wieder zu ihr gehört, nach dem Motto: Wenn ich *über mich* nachdenke, dann ist dies *mein* Gedanke. In dem hier vorgestellten Modell heißt dies, dass jede erneute Reflexion der Menge aller Reflexionen beliebiger Stufe bereits in letzterer Menge enthalten ist. Dies ist für primitive Reflexionen *im allgemeinen nicht richtig*, da, wie gesagt, aufgrund des Cantorschen Diagonalarguments $R(\mathcal{M}^*) \subsetneqq M^*$ gilt. Für *finitäre primitive Reflexionen* jedoch ist dies richtig, was im folgenden begründet werden soll.

7.4 Die Möglichkeit von Selbstbewusstsein

Sei also M ein beliebiger Gegenstandsbereich, $F(M)$ die Menge aller finitären Reflexionen des Gegenstandsbereichs M und

$\mathcal{M}_* = \bigcup_{n=0}^{\infty} F^n(M)$ die Menge aller finitären Reflexionen beliebiger Stufe n von M. Wir wollen zeigen, dass finitäre primitive Reflexionen von $\mathcal{M}_*$ nicht über $\mathcal{M}_*$ hinausführen. Aufgrund der bereits vorgenommenen Identifikationen reicht es zu zeigen, dass es eine eindeutige Zuordnung (Bijektion) der Elemente von $F(\mathcal{M}_*)$ zu denen von $\mathcal{M}_*$ gibt. Das ist die

Aussage der folgenden Transformation des Cantorschen Diagonalarguments.

Argument der finitären Reflexion.

Für eine finitäre Reflexion $f\colon \mathcal{M}_* \longrightarrow \{0,1\}$ mit Träger $S = [f = 1] = \{s \in \mathcal{M}_* \mid f(s) = 1\}$ gibt es ein minimales $k \geq 0$ derart, dass $S \subseteq F^k(M)$ ist und durch

$$\tilde{f}(s) = \begin{cases} 1, & s \in S \\ 0, & s \notin S, \quad s \in F^k(M) \end{cases}$$

eine Reflexion $\tilde{f} \in F^{k+1}(M) \subseteq \mathcal{M}_*$ definiert wird.

Die Zuordnung $f \longmapsto \tilde{f}$ ist eine Bijektion von $F(\mathcal{M}_*)$ auf $\mathcal{M}_*$.

Begründung. Da S endlich ist, so ist $S \subseteq F^k(M)$ mit minimalem k und $\tilde{f}$ ist eine Reflexion von $F^k(M)$. Vermöge Identifikation ist $\tilde{f}(s) = 0$ für $s \notin F^k(M)$.

(a) Wir zeigen zunächst, dass $f \longmapsto \tilde{f}$ injektiv ist, d.h., aus $\tilde{f} = \tilde{g}$ folgt $f = g$. Nach Definition von $\tilde{f}$ und $\tilde{g}$ gibt es k und l, ohne Einschränkung $l \leq k$ und daher $F^l(M) \subseteq F^k(M)$, so dass gilt

$\tilde{f}(s) = f(s)$ und $\tilde{g}(s) = g(s)$ für $s \in F^k(M)$.

Daher ist $f(s) = g(s)$ für $s \in F^k(M)$. Für $s \in \mathcal{M}_*$, aber $s \notin F^k(M)$ ist auch $s \notin F^l(M)$ und daher muss $f(s) = 0$ und $g(s) = 0$ sein. Also gilt $f(s) = g(s)$ für alle $s \in \mathcal{M}_*$.

(b) Wir zeigen, dass $f \longmapsto \tilde{f}$ surjektiv ist, d.h., zu $h \in \mathcal{M}_*$ gibt es ein $f \in F(\mathcal{M}_*)$ mit $\tilde{f} = h$.

Da $h \in \mathcal{M}_*$, so ist $h \in F^n(M)$ für ein $n \geq 0$. Da h finitär ist, so gibt es ein minimales p mit $T = [h = 1] \subseteq F^p(M)$. Es ist $p \leq n - 1$. Definiere $f\colon \mathcal{M}_* \longrightarrow \{0,1\}$ durch

$$f(t) = \begin{cases} 1, & t \in T \\ 0, & t \notin T. \end{cases}$$

Es ist $f \in F(\mathcal{M}_*), [f = 1] = T$ und

$$\tilde{f}(t) = \begin{cases} 1, & t \in T \\ 0, & t \notin T, t \in F^p(M). \end{cases}$$

Daher ist $\tilde{f} \in F^{(p+1)}(M)$ und wegen $p \leq n-1$ gilt $\tilde{f} \in F^n(M)$. Also sind $\tilde{f}$ und h in $F^n(M)$ mit gleichem Träger und stimmen daher überein.

7.5 Abschließende Bemerkungen

Der hier vorgestellte analytische Zugang zum Konzept des Selbstbewusstseins lässt noch viele Aspekte, die zu thematisieren wären, außer Acht, z.B. könnte man statt primitiver Reflexionen auch an differenziertere Formen von Reflexion denken wie mehrdimensionale oder gestufte Reflexionen. Die zentrale Absicht bei diesem Zugang ist es, eine klare und konsistente theoretische Formulierung eines Konzepts des Selbstbewusstseins zu geben, um das sich in der Philosophie viele unklare, widersprüchliche, zirkuläre oder paradoxe Gedanken ranken. So schreibt z.B. M. Frank:

> „Aus Gründen, mit denen wir inzwischen vertraut sind, verwirrt sich die Deutung des Gegenstandes von Selbstbewusstsein als Selbst, als Bewusstsein oder als Ich entweder in Zirkeln oder in infiniten Regressen."
>
> ([6, S. 252]).

In der hier vorgenommenen Konstruktion wird der Zirkel der Selbstbezüglichkeit in eine Sequenz fortgesetzter Reflexionen aufgelöst, deren Gesamtheit dann auf eine spezifische und genau formulierbare Weise eine Selbstreflexion erlaubt. Hinsichtlich der in der Geschichte der Philosophie oft thematisierten Inkongruenz von Innen- und Außenperspektive wird in der Konstruktion von $\mathcal{M}_*$ durch die Entsprechung $f \longmapsto \tilde{f}$ eine

Verbindung zwischen der Innenperspektive $\mathcal{M}_*$ und der Außenperspektive $F(\mathcal{M}_*)$ hergestellt.

Das Argument der finitären Reflexion zeigt, dass ein Subjekt – sei es ein Organismus, ein Computer, ein Roboter – sich selbst zu reflektieren vermag, falls es in bezug auf einen gewissen Gegenstandsbereich M eine Struktur wie $\mathcal{M}_*$ aufweist.

Literaturverzeichnis

Bücher

[1] Charles H. Bennett: Logical Depth and Physical Complexity. In: Rolf Herken: The universal Turing machine. A half-century survey. Oxford u.a.: Oxford University Press 1988, S. 227-257

[2] Gregory J. Chaitin: Information-Theoretic Incompleteness. Singapore: World Scientific 1992

[3] Gregory J. Chaitin: On the intelligibility of the universe and the notions of simplicity, complexity and irreducibility. Berlin: Akademie Verlag 2004

[4] Gregory J. Chaitin: The Limits of Mathematics. Singapore: Springer-Verlag Singapore Pte. Ltd. 1998

[5] John W. Dawson, Jr.: Logical Dilemmas. The Life and Work of Kurt Gödel. Wellesley, Massachusetts: A K Peters 1997

[6] Manfred Frank: Selbstbewußtsein und Selbsterkenntnis. Stuttgart: Reclam 1991

[7] Siegfried Gottwald et al: Meyers Kleine Enzyklopädie Mathematik. 14. Auflage. Mannheim: Meyers Lexikonverlag 1995

[8] Gianbruno Guerrerio: Kurt Gödel: logische Paradoxien und mathematische Wahrheit. Heidelberg: Spektrum-der-Wissenschaft-Verlagsgeselleschaft 2002

[9] Susan Haack: Philosophy of Logics.
Cambridge/ London/ New York/ Melbourne: Cambridge University Press 1978

[10] Andrew Hodges: Alan Turing and the Turing Machine. In: Rolf Herken: The universal Turing machine. A half-century survey. Oxford u.a.: Oxford University Press 1988, S. 3-15

[11] Stephen Kleene: Turing's Analysis of Computability, and Major Applications of It. In: Rolf Herken: The universal Turing machine. A half-century survey. Oxford u.a.: Oxford University Press 1988, S. 17-54

[12] Ming Li, Paul Vitányi: An Introduction to Kolmogorov Complexity and Its Applications. New York u.a.: Springer 1997

[13] Roger Penrose: Das Große, das Kleine und der menschliche Geist. Heidelberg: Spektrum Akademischer Verlag 2002
Original: Roger Penrose: The Large, the Small and the Human Mind. Cambridge: Cambridge University Press 1997

[14] Roger Penrose: Is Conscious Awareness Consistent with Space–Time Descriptions? In: E. Rudoph, I.O. Stamatescu (Eds.) Philosophy, Mathematics and Modern Physics. A Dialogue. Berlin: Springer 1994, S. 34-47

[15] Roger Penrose: Schatten des Geistes. Wege zu einer neuen Physik des Bewußtseins. Heidelberg: Spektrum Akademischer Verlag 1995

[16] Stephen Read: Philosophie der Logik. Eine Einführung. Reinbek: rowohlts enzyklopädie 1997

[17] David Ruelle: Zufall und Chaos. Berlin, Heidelberg: Springer-Verlag 1994

[18] R.M. Sainsbury: Paradoxien. Erweiterte Ausgabe. Stuttgart: Philipp Reclam jun. GmbH & Co., 1995

[19] Alan Turing: Computing Machinery and Intelligence. In: Douglas R. Hofstadter, Daniel C. Dennett (Hrsg.): The Mind's I. Fantasies and Reflections on Self and Soul. New York: Basic Books 1981, S. 53-68

[20] Alan Turing: Maschinelle Rechner und Intelligenz. In: Douglas R. Hofstadter, Daniel C. Dennett (Hrsg.): Einsicht ins Ich. Fantasien und Reflexionen über Selbst und Seele. Stuttgart: Klett 1981, S. 59-72

[21] Kurt Gödel: Collected Works, Volume I: Publications 1929 - 1936. New York: Oxford University Press 1986

Zeitschriften-Artikel

[22] Gregory J. Chaitin: Computers, paradoxes and the foundations of mathematics. In: American Scientist, 90. Jg, Nr. 2 (2002), S. 164-171

[23] Gregory J. Chaitin: Gödel´s Theorem and Information. In: International Journal of Theoretical Physics, 21. Jg, Nr. 12 (1982), S. 941-954

[24] Gregory J. Chaitin: Grenzen der Berechenbarkeit. In: Spektrum der Wissenschaft. 27. Jg., Nr. 2 (2004), S. 86-93

[25] Gregory J. Chaitin: The limits of reason. In: Scientific American, 294. Jg., Nr. 3 (2006), S. 74-81

[26] Jack Copeland, Diane Proudfoot: Alan Turing's forgotten Ideas in Computer Science. In: Scientific American, 280. Jg., Nr. 4 (1999), S. 98-103.

[27] Martin Davis: The Incompleteness Theorem. In: Notices of the American Mathematical Society. A Tribute to Kurt Gödel. 53 Jg., Nr.4 (2006), S. 414-418

[28] Lionel S. Penrose und Roger Penrose: Impossible objects: a special type of visual illusion. In: British Journal of Psychology, 49. Jg., Nr. 1 Februar (1958), S. 31-33

[29] Roger Penrose: On the cohomology of impossible figures. In: Structural Topology, 17. Jg. (1991), S. 11-16

[30] Ralf Schindler: Kurt Gödel (1906-1978). In: Mitteilungen der Deutschen Mathematiker-Vereinigung, 14. Jg., Nr. 1 (2006), S. 42-45

Internet-Quellen

[31] `http://de.encarta.msn.com/encyclopedia_761596028/Denken.html`
(Stand 06.07.2006)

[32] `http://de.wikipedia.org/wiki/Alan\_Turing`
(Stand 29.05.2006)

[33] `http://de.wikipedia.org/wiki/Alfred_Tarski`
(Stand 17.05.2006)

[34] `http://de.wikipedia.org/wiki/Kurt_Gödel`
(Stand 17.05.2006)

[35] `http://de.wikipedia.org/wiki/Penrose-Dreieck`
(Stand 02.06.2006)

[36] `http://de.wikipedia.org/wiki/Saul_Kripke`
(Stand 17.05.2006)

[37] `http://de.wikipedia.org/wiki/Turingmaschine`
(Stand 11.06.2006)

[38] http://de.wikipedia.org/wiki/Turingtest
(Stand 11.06.2006)

[39] http://encyclopedie-de.snyke.com/articles/godelscher_unvollstandigkeitssatz.html
(Stand 17.05.2006)

[40] http://ling.uni-konstanz.de/pages/publ/PDF/ap027.pdf
(Stand 22.05.2006)

[41] http://on1.zkm.de/zkm/discuss/msgReader\$2863?mode=day (Stand 29.5.2006)

[42] http://plato.stanford.edu/archives/sum2002/entries/turing/
(Stand 29.05.2006)

[43] http://plato.stanford.edu/entries/tarski-truth/
(Stand 17.05.2006)

[44] http://plus.maths.org/issue37/features/omega/
(Stand 29.5.2006)

[45] http://www.britannica.com/eb/article-9059107?&query=penrose
(Stand 08.07.2006)

[46] http://www.brockhaus.de/infothek/infothek_detail.php?nr=12143
(Stand 21.05.2006)

[47] http://www.cs.auckland.ac.nz/CDMTCS/chaitin/
(Stand 29.5.2006)

[48] http://www.gosai.com/science/mathematics-controversy.html
(Stand 30.7.2006)

[49] http://www.math.uni-wuppertal.de/~fischer/ft/02/ft4_6.pdf
(Stand 02.06.2006)

[50] http://www.mathe.tu-freiberg.de/~matos/dateien/kt_06/tm.pdf
(Stand 17.07.2006)

[51] http://www.matheprisma.uni-wuppertal.de/Module/Turing/
(Stand 17.07.2006)

[52] http://www.nuclear-engineering.at/news/events/science-talk/docs/chaitin_cv.html
(Stand 29.5.2006)

[53] http://www.philosophenlexikon.de/tarski.htm
(Stand 17.05.2006)

[54] http://www.research.cornell.edu/KIC/events/Bennett2005/index.html
(Stand 27.07.2006)

[55] http://www.research.ibm.com/people/b/bennetc/chbbio.html
(Stand 28.07.06)

[56] http://www.rutherfordjournal.org/images/proud01.jpg
(Stand 18.06.2006)

[57] http://www.umcs.maine.edu/~chaitin/complete.html
(Stand 29.5.2006)

[58] http://www.wundersamessammelsurium.de/Optisches/0T_IstEsMoeglich/
(Stand 02.06.2006)

[59] http://www-history.mcs.st-andrews.ac.uk/Biographies/Penrose.html
(Stand 08.07.2006)

Vorträge

[60] Ulrich Krause: Kurt Gödel und die Grenzen des Wissens. Vortrag zum 100. Geburtstag von Kurt Gödel am 28. April 2006 im Haus der Wissenschaft, Bremen