Dirk W. **HOFFMANN**

THEORETISCHE INFORMATIK

5., aktualisierte Auflage

Im Internet:
Lösungen zu den Übungsaufgaben

HANSER

Hoffmann

Theoretische Informatik

Dirk W. Hoffmann

Theoretische Informatik

5., aktualisierte Auflage

HANSER

Autor:

Prof. Dr. Dirk W. Hoffmann
Fakultät für Informatik und Wirtschaftsinformatik
Hochschule Karlsruhe – Technik und Wirtschaft

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Internet: www.hanser-fachbuch.de

Lektorat: Dipl.-Ing. Natalia Silakova-Herzberg
Herstellung: Frauke Schafft
Satz: Dirk W. Hoffmann
Titelmotiv: © stock.adobe.com/fran_kie
Covergestaltung: Max Kostopoulos
Coverkonzept: Marc Müller-Bremer, www.rebranding.de, München
Druck und Bindung: Eberl & Koesel GmbH, Altusried-Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN 978-3-446-47029-3
E-Book-ISBN 978-3-446-47256-3

Vorwort

Für die meisten Menschen ist die Informatik fest mit der Entstehungsgeschichte des Computers verbunden; einer Technik, die von außen betrachtet keinen Grenzen zu unterliegen scheint. Wir erleben seit Jahren eine schier ungebremste Entwicklung und sind längst daran gewöhnt, dass der Computer von heute schon morgen überholt ist. Dass sich hinter der Computertechnik eine tiefgründige Wissenschaft verbirgt, die all die großen Erfolge erst möglich macht, bleibt vielen Menschen verborgen. Die Rede ist von der theoretischen Informatik.

In der Grundlagenausbildung hat die theoretische Informatik ihren festen Platz eingenommen. Viele Studierende begegnen ihr mit gemischten Gefühlen und von manchen wird sie gar als bedrohlich empfunden. Mitverantwortlich für diese Misere sind die historischen Wurzeln der theoretischen Informatik. Entstanden aus der Mathematik, wird sie häufig in einer Präzision dargestellt, die in der Informatik ihresgleichen sucht. Manch ein Leser verirrt sich schnell in einem Gewirr aus Definitionen, Sätzen und Beweisen, das die Sicht auf die eigentlichen Konzepte und Methoden unfreiwillig verdeckt. Dass die theoretische Informatik weder schwer noch trocken sein muss, versuche ich mit diesem Buch zu beweisen.

Die folgenden Kapitel werden von zwei Leitmotiven getragen. Zum einen möchte ich die grundlegenden Konzepte, Methoden und Ergebnisse der theoretischen Informatik vermitteln, ohne diese durch einen zu hohen Abstraktionsgrad zu vernebeln. Hierzu werden die Problemstellungen durchweg anhand von Beispielen motiviert und die Grundideen der komplizierteren Beweise an konkreten Probleminstanzen nachvollzogen. Zum anderen habe ich versucht, den Lehrstoff in vielerlei Hinsicht mit Leben zu füllen. An zahlreichen Stellen werden Sie Anmerkungen und Querbezüge vorfinden, die sich mit der historischen Entwicklung dieser einzigartigen Wissenschaftsdisziplin beschäftigen.

Bei allen Versuchen, einen verständlichen Zugang zu der nicht immer einfachen Materie zu schaffen, war es mir ein Anliegen, keinen Verlust an Tiefe zu erleiden. Das Buch ist für den Bachelor-Studiengang konzipiert und deckt die typischen Lehrinhalte ab, die im Grundstudium an den hiesigen Hochschulen und Universitäten unterrichtet werden.

Karlsruhe, im Februar 2022 Dirk W. Hoffmann

Symbolwegweiser

Definition

Satz, Lemma, Korollar

Leichte Übungsaufgabe

Mittelschwere Übungsaufgabe

Schwere Übungsaufgabe

Lösungen zu den Übungsaufgaben

In wenigen Schritten erhalten Sie die Lösungen zu den Übungsaufgaben:

1. Gehen Sie auf die Seite www.dirkwhoffmann.de/TH
2. Geben Sie den neben der Aufgabe abgedruckten Webcode ein
3. Die Musterlösung wird als PDF-Dokument angezeigt

Inhaltsverzeichnis

1 Einführung

„Wir müssen wissen. Wir werden wissen.“

David Hilbert

1.1 Was ist theoretische Informatik?

Kaum eine andere Technologie hat unsere Welt so rasant und nachhaltig verändert wie der Computer. Unzählige Bereiche des täglichen Lebens werden inzwischen von Bits und Bytes dominiert – selbst solche, die noch vor einigen Jahren als elektronikfreie Zone galten. Die Auswirkungen dieser Entwicklung sind bis in unser gesellschaftliches und kulturelles Leben zu spüren und machen selbst vor der deutschen Sprache keinen Halt. Vielleicht haben auch Sie heute schon *gemailt*, *gesimst* oder *gegoogelt* (Abbildung 1.1). Die Digitalisierung unserer Welt ist in vollem Gange und eine Abschwächung der eingeschlagenen Entwicklung zumindest mittelfristig nicht abzusehen.

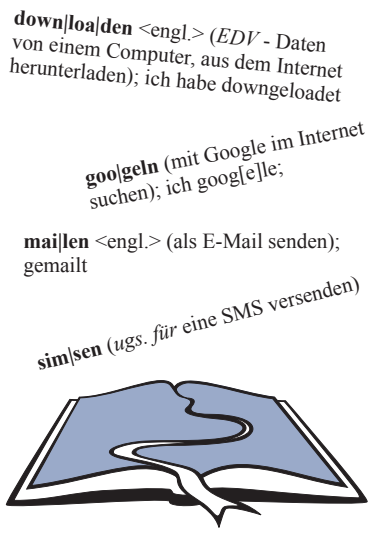


Abbildung 1.1: Die zunehmende Technisierung des Alltagslebens macht auch vor der deutschen Sprache keinen Halt. Im Jahr 2004 schaffte es das neudeutsche Verb *googeln* in den Duden [106]. Dort finden sich auch die Worte *mailen*, *simsen* und *downloaden* wieder.

Die in der Retrospektive einzigartige Evolution der Computertechnik ist eng mit der Entwicklung der Informatik verbunden. Als naturwissenschaftliche Fundierung der Computertechnik untersucht sie die Methoden und Techniken, die eine digitale Welt wie die unsere erst möglich machen. In der gleichen Geschwindigkeit, in der Computer die Welt eroberten, konnte sich die Informatik von einer Nischendisziplin der Mathematik und Elektrotechnik zu einer eigenständigen Grundlagenwissenschaft entwickeln. War sie zu Anfang auf wenige Kernbereiche beschränkt, so präsentiert sich die Informatik mittlerweile als eine breit gefächerte Wissenschaftsdisziplin. Heute existieren Schnittstellen in die verschiedensten Bereiche wie die Biologie, die Medizin und sogar die bildenden Künste.

In Abbildung 1.2 sind die vier Säulen dargestellt, von denen die Informatik gegenwärtig getragen wird. Eine davon ist die theoretische Informatik. Sie beschäftigt sich mit den abstrakten Konzepten und Methoden, die sich hinter den Fassaden moderner Computersysteme verbergen. Die theoretische Informatik ist vor der technischen Informatik die

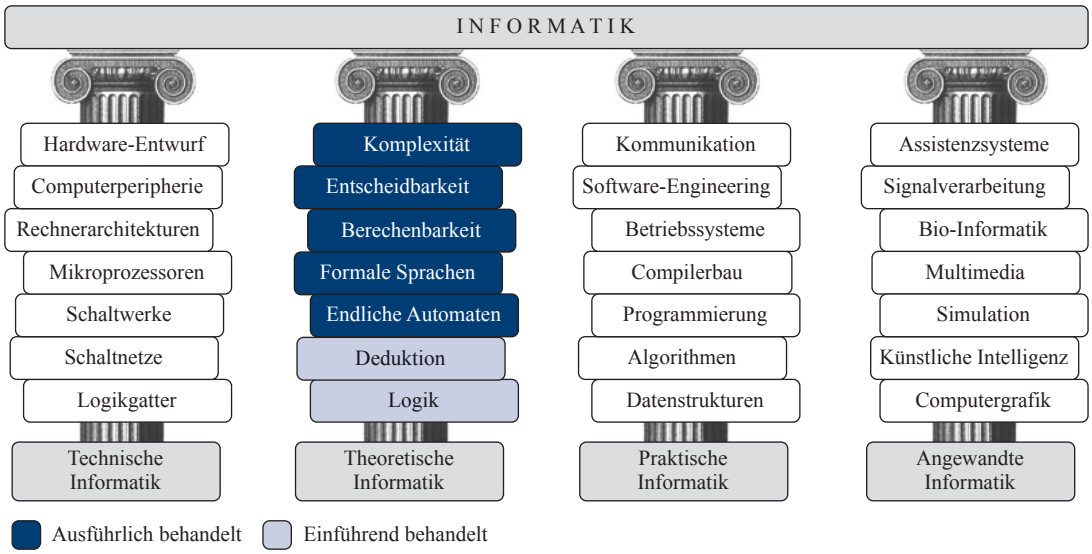


Abbildung 1.2: Die vier Säulen der Informatik

älteste Kernsäule und hat ihren direkten Ursprung in der Mathematik. Trotz ihres relativen Alters hat dieser Zweig der Informatik nichts von seiner ursprünglichen Bedeutung verloren. Er bildet das konzeptionelle Fundament, auf dem die anderen Bereiche der Informatik solide ruhen und aus dessen Wissensfundus sie schöpfen.

Betrachten wir die inhaltlichen Themen der modernen theoretischen Informatik genauer, so lassen sich diese in die folgenden Teilgebiete untergliedern (vgl. Abbildung 1.3):

- Logik und Deduktion (Kapitel 3)

  Die Logik beschäftigt sich mit grundlegenden Fragestellungen mathematischer Theorien. Im Mittelpunkt steht die Untersuchung *formaler Systeme* (*Kalküle*), in denen Aussagen aus einer kleinen Menge vorgegebener Axiome durch die Anwendung fest definierter Schlussregeln abgeleitet werden. Die Logik spielt nicht nur in der theoretischen Informatik, sondern auch in der technischen Informatik und der Software-Entwicklung eine Rolle. Mit der Aussagenlogik gibt sie uns ein Instrumentarium an die Hand, mit dem wir jede erdenkliche Hardware-Schaltung formal beschreiben und analysieren können. Ferner lässt sich mit der Prädikatenlogik und den Logiken höherer Stufe das Verhalten komplexer Hardware- und Software-Systeme exakt spezifizieren und in Teilen verifizieren.

- Formale Sprachen (Kapitel 4)

  Die Theorie der formalen Sprachen beschäftigt sich mit der Analyse, der Klassifikation und der generativen Erzeugung von Wortmengen. Künstliche Sprachen sind nach festen Regeln aufgebaut, die zusammen mit dem verwendeten Symbolvorrat eine formale Grammatik bilden. Die zugrunde liegende Theorie gibt uns die Methoden und Techniken an die Hand, die für den systematischen Umgang mit modernen Programmiersprachen und dem damit zusammenhängenden Compilerbau unabdingbar sind. Viele Erkenntnisse aus diesem Bereich haben ihre Wurzeln in der Linguistik und stoßen dementsprechend auch außerhalb der Informatik auf reges Interesse.

- Automatentheorie (Kapitel 5)

  Hinter dem Begriff des endlichen Automaten verbirgt sich ein abstraktes Maschinenmodell, das sich zur Modellierung, zur Analyse und zur Synthese zustandsbasierter Systeme eignet. Auf der obersten Ebene untergliedern sich endliche Automaten in Akzeptoren und Transduktoren. Erstere zeigen einen engen Bezug zu den formalen Sprachen, Letztere spielen im Bereich des Hardware-Entwurfs eine dominierende Rolle. Sie sind das mathematische Modell, mit dem sich das zeitliche Verhalten synchron getakteter Digitalschaltungen exakt beschreiben und analysieren lässt.

- Berechenbarkeitstheorie (Kapitel 6)

  Die Berechenbarkeitstheorie beschäftigt sich mit grundlegenden Untersuchungen über die algorithmische Lösbarkeit von Problemen. Die Bedeutung dieses Teilgebiets der theoretischen Informatik ist zweigeteilt. Zum einen wird das gesamte Gebiet der Algorithmentechnik durch die Definition formaler Berechnungsmodelle auf einen formalen Unterbau gestellt. Zum anderen ermöglicht uns die systematische Vorgehensweise, die Grenzen der prinzipiellen Berechenbarkeit auszuloten.

- Komplexitätstheorie (Kapitel 7)

  Während die Berechenbarkeitstheorie Fragen nach der puren Existenz von Algorithmen beantwortet, versucht die Komplexitätstheorie die Eigenschaften einer Lösungsstrategie quantitativ zu erfassen. Algorithmen werden anhand ihres Speicherplatzbedarfs und Zeitverbrauchs in verschiedene Komplexitätsklassen eingeteilt, die Rückschlüsse auf deren praktische Verwertbarkeit zulassen. Die Ergebnisse dieser Theorie beeinflussen den gesamten Bereich der modernen Software- und Hardware-Entwicklung.

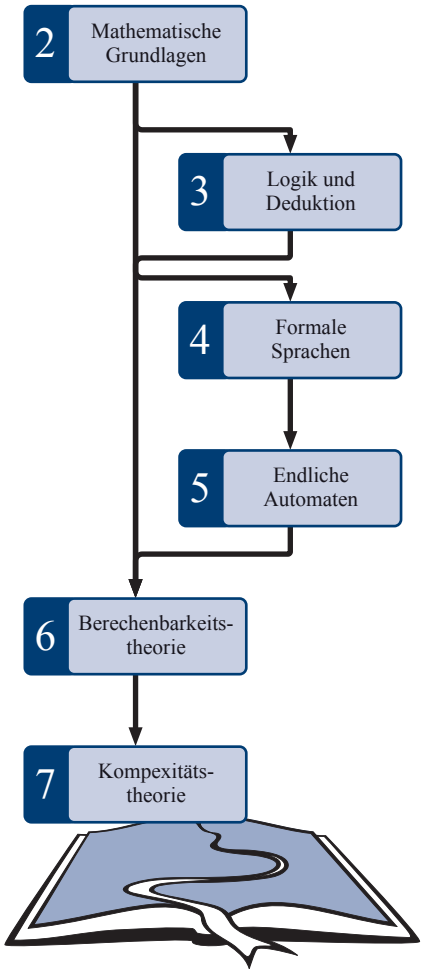


Abbildung 1.3: Kapitelübersicht. Die Pfeile deuten an, wie die einzelnen Kapitel inhaltlich zusammenhängen.

1. Axiom:
„Zu zwei Punkten gibt es genau eine Gerade, auf der sie liegen.“

2. Axiom:
„Jede gerade Strecke zwischen zwei Punkten lässt sich eindeutig verlängern.“

3. Axiom:
„Zu einem Punkt und einer Strecke kann man genau einen Kreis konstruieren.“

4. Axiom:
„Alle rechten Winkel sind gleich.“

5. Axiom:
„Zu einer Geraden und einem Punkt außerhalb der Geraden gibt es genau eine Gerade, die durch den Punkt geht und parallel zur ersten Geraden ist.“

Euklid von Alexandria
(ca. 365 v. Chr. – ca. 300 v. Chr.)

Abbildung 1.4: Die euklidischen Axiome

1.2 Zurück zu den Anfängen

Bevor wir uns ausführlich mit den Begriffen und Methoden der theoretischen Informatik beschäftigen, wollen wir in einem historischen Streifzug herausarbeiten, in welchem Umfeld ihre Teilgebiete entstanden sind und in welchem Zusammenhang sie heute zueinander stehen.

1.2.1 Die Mathematik in der Krise

Die theoretische Informatik hat ihre Wurzeln in der Mathematik. Ihre Geschichte beginnt mit der *Grundlagenkrise*, die Anfang des zwanzigsten Jahrhunderts einen Tiefpunkt in der mehrere tausend Jahre alten Historie der Mathematik markierte. Um die Geschehnisse zu verstehen, die die älteste aller Wissenschaften in ihren Grundfesten erschütterten, wollen wir unseren Blick zunächst auf das achtzehnte Jahrhundert richten. Zu dieser Zeit war die Mathematik schon weit entwickelt, jedoch noch lange nicht die abstrakte Wissenschaft, wie wir sie heute kennen. Fest in der realen Welt verankert, wurde sie vor allem durch Problemstellungen der physikalischen Beobachtung vorangetrieben. Zahlen waren nichts weiter als Messgrößen für reale Objekte und weit von den immateriellen Gedankengebilden der modernen Zahlentheorie entfernt. So wenig wie die Mathematik als eigenständige Wissenschaft existierte, so wenig gab es den reinen Mathematiker.

Im neunzehnten Jahrhundert änderte sich die Sichtweise allmählich in Richtung einer abstrakteren Mathematik. Zahlen und Symbole wurden von ihrer physikalischen Interpretation losgelöst betrachtet und entwickelten sich langsam, aber beharrlich zu immer abstrakter werdenden Größen. Mit der geänderten Sichtweise war es nun möglich, eine Gleichung der Form

$$c^2 = a^2 + b^2$$

völlig unabhängig von ihrer pythagoreischen Bedeutung zu betrachten. In ihrer abstraktesten Interpretation lässt sie sich als mathematisches Spiel begreifen, das uns erlaubt, die linke Seite durch die rechte zu ersetzen. Die Variablen a, b und c degradieren in diesem Fall zu inhaltsleeren Größen, die in keinerlei Bezug mehr zu den Seitenlängen eines rechtwinkligen Dreiecks stehen.

Dass es richtig war, das mathematische Gedankengerüst von seiner physikalischen Interpretation zu trennen, wurde durch die Physik selbst untermauert. So machte die zu Beginn des zwanzigsten Jahrhunderts auf-

„Wenn es sich darum handelt, die Grundlagen einer Wissenschaft zu untersuchen, so hat man ein System von Axiomen aufzustellen, welche eine genaue und vollständige Beschreibung derjenigen Beziehungen enthalten, die zwischen den elementaren Begriffen jener Wissenschaft stattfinden. Die aufgestellten Axiome sind zugleich die Definitionen jener elementaren Begriffe und jede Aussage innerhalb des Bereiches der Wissenschaft, deren Grundlagen wir prüfen, gilt uns nur dann als richtig, falls sie sich mittelst einer endlichen Anzahl logischer Schlüsse aus den aufgestellten Axiomen ableiten lässt. Bei näherer Betrachtung entsteht die Frage, ob etwa gewisse Aussagen einzelner Axiome sich untereinander bedingen und ob nicht somit die Axiome noch gemeinsame Bestandteile enthalten, die man beseitigen muss, wenn man zu einem System von Axiomen gelangen will, die völlig voneinander unabhängig sind.
Vor allem aber möchte ich unter den zahlreichen Fragen, welche hinsichtlich der Axiome gestellt werden können, dies als das wichtigste Problem bezeichnen, zu beweisen, dass dieselben untereinander widerspruchslos sind, d.h., dass man aufgrund derselben mittelst einer endlichen Anzahl von logischen Schlüssen niemals zu Resultaten gelangen kann, die miteinander in Widerspruch stehen.“

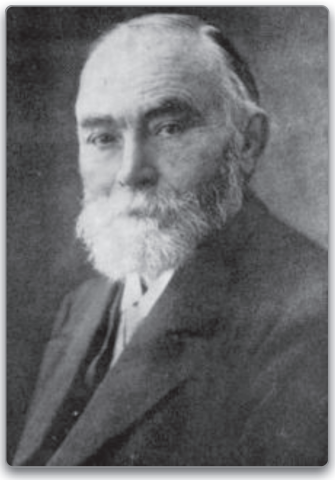
David Hilbert
(1862 – 1943)

Abbildung 1.5: Auszug aus Hilberts historischer Jahrhundertrede auf dem internationalen Kongress der Mathematiker in Paris

keimende Quantenmechanik deutlich, dass die damals wie heute merkwürdig anmutenden Effekte der Elementarteilchenphysik nur mithilfe abstrakter Modelle präzise erfasst werden können. Viele mathematische Konstrukte wie z. B. der *Hilbertraum* oder die *abstrakte Gruppe* konnten nachträglich zur Beschreibung der Natur eingesetzt werden, obwohl diese nichts mit unserer makroskopischen Anschauung gemeinsam haben.

Die zunehmende Abstraktion ließ Raum für Fragen zu, die sich in einer physikalisch gedeuteten Mathematik nicht stellen. Interpretieren wir z. B. die *euklidischen Axiome* (Abbildung 1.4) ausschließlich im Sinne der klassischen Geometrie, so erscheinen sie als reine Selbstverständlichkeit. Sie decken sich mit den Erfahrungen, die wir in der makroskopischen Welt tagtäglich machen und kaum jemand würde auf die Idee kommen, an ihnen zu zweifeln. Entsprechend lange galten die Axiome als unantastbar.

Die Situation ändert sich, sobald wir die Mathematik als ein abstraktes Wechselspiel von Symbolen und Regeln betreiben. Lösen wir uns von der intuitiven Interpretation der euklidischen Axiome, so stellt sich



Gottlob Frege
(1848 – 1925)

Abbildung 1.6: Gottlob Frege. Der im mecklenburgischen Wismar geborene Mathematiker zählt zu den Mitbegründern der mathematischen Logik und der analytischen Philosophie. Im Jahr 1879 eröffnete Frege mit seiner berühmten *Begriffsschrift* einen axiomatischen Zugang zur Logik [35]. Er führt darin die grundlegenden Konzepte und Begriffe ein, die wir auch heute noch in der Prädikatenlogik (Abschnitt 3.2) und den Logiken höherer Stufe (Abschnitt 3.3.2) verwenden. Sein Begriffsgerüst war deutlich weiter entwickelt als die Syllogismen des Aristoteles – der bis dato präzisesten Form des logischen Schließens. Die meiste Zeit seines Lebens war Frege ein überzeugter Verfechter des *Logizismus*. Er vertrat die Auffassung, dass die Mathematik ein Teil der Logik sei. In diesem Sinne müssen sich alle Wahrheiten auf eine Menge von Axiomen zurückführen lassen, die nach Freges Worten *„eines Beweises weder fähig noch bedürftig“* seien. Er stand damit in einer Gegenposition zu anderen Mathematikern seiner Zeit, von denen viele die Logik als isoliertes Teilgebiet der Mathematik begriffen.

die Frage, ob diese ein vollständiges und widerspruchsfreies Gebilde ergeben. Im Jahr 1899 gelang es David Hilbert, diese Frage positiv zu beantworten. Er postulierte ein Axiomensystem, aus dem sich alle Sätze der euklidischen Geometrie ableiten lassen, ohne die verwendeten Symbole mit einer speziellen Interpretation zu versehen [46].

Inspiriert von den Anfangserfolgen stand die Mathematik um die Jahrhundertwende vollends im Zeichen der axiomatischen Methode. Das Führen eines Beweises wurde als der Prozess verstanden, Sätze durch die Anwendung wohldefinierter Schlussregeln aus einer kleinen Menge vorgegebener, a priori als wahr definierter Axiome abzuleiten. Eine spezielle Interpretation der Symbole war hierzu nicht erforderlich und im Grunde genommen auch gar nicht angestrebt. Die Mathematik wurde zu einem Spiel, das nach starren Regeln funktionierte, und das Führen eines Beweises zu einem mechanischen Prozess werden ließ.

Der deutsche Mathematiker David Hilbert war kein Unbekannter. Bereits zu Lebzeiten wurde er als Ikone gefeiert und beeinflusste wie kein anderer die Mathematik des beginnenden zwanzigsten Jahrhunderts. Im Jahr 1900 hielt Hilbert auf dem internationalen Kongress der Mathematiker in Paris eine wegweisende Rede, an der sich die weitere Stoßrichtung der gesamten Mathematik über Jahre hinweg orientieren sollte (vgl. Abbildung 1.5). In seiner Ansprache trug er 23 ungelöste Probleme vor, die für die Mathematik von immenser Wichtigkeit, aber bis dato ungelöst waren.

Bereits an zweiter Stelle forderte Hilbert die Weltgemeinschaft dazu auf, einen Beweis für die Widerspruchsfreiheit der arithmetischen Axiome zu liefern. Das Problem, das Hilbert hier ansprach, war von immenser Wichtigkeit für die gesamte Mathematik, schließlich adressieren die arithmetischen Axiome den vitalen Kern, auf dem alle Teilbereiche dieser Wissenschaft aufbauen. Solange es nicht gelingt, die Widerspruchsfreiheit formal zu beweisen, kann nicht mit Sicherheit ausgeschlossen werden, dass sich z. B. neben dem Theorem $1+1=2$ auch das Theorem $1+1\neq 2$ aus den Axiomen ableiten lässt. Das verästelte Gebäude der Mathematik würde auf einen Schlag in Trümmern vor uns liegen.

Bereits wenige Jahre nach Hilberts Rede sollte die Wissenschaftsgemeinde erleben, wie real eine solche Gefahr wirklich war. Der deutsche Mathematiker Gottlob Frege (Abbildung 1.6) spürte sie am eigenen Leib, als er 1902 ein formales Axiomensystem für ein Teilgebiet der Mathematik aufstellte, das auf den ersten Blick so intuitiv und einfach erscheint wie kaum ein anderes. Die Rede ist von der *Mengenlehre*. Der zweite Band seiner *Grundgesetze der Arithmetik* schließt mit dem folgenden Nachwort [33, 34]:

„Einem wissenschaftlichen Schriftsteller kann kaum etwas Unerwünschteres begegnen, als dass ihm nach Vollendung einer Arbeit eine der Grundlagen seines Baues erschüttert wird."

Doch wodurch wurde Freges Arbeit so grundlegend erschüttert, dass er sein gesamtes Werk gefährdet sah? Die Antwort ist in einem Brief von Bertrand Russell zu finden, den er im Jahr 1902 an Frege schickte – just zu der Zeit, als dieser sein mathematisches Werk vollendete. Aufbauend auf den Begriffen der naiven Mengenlehre definierte Russell die Menge aller Mengen, die sich nicht selbst als Element enthalten:

$$M := \{M' \mid M' \notin M'\}$$

Die Definition von M ist mit der damals verwendeten Mengendefinition von Georg Cantor vereinbar, führt bei genauerer Betrachtung jedoch unweigerlich zu einem Widerspruch. Da für jedes Element a und jede Menge M entweder $a \in M$ oder $a \notin M$ gilt, muss auch M entweder in sich selbst enthalten sein oder nicht. Die Definition von M offenbart uns jedoch das folgende erstaunliche Ergebnis:

$$M \in M \Rightarrow M \notin M, \quad M \notin M \Rightarrow M \in M$$

Der als *Russell'sche Antinomie* bekannte Widerspruch entlarvte den Cantor'schen Mengenbegriff als in sich widersprüchlich und lies Freges Gedankengerüst wie ein Kartenhaus in sich zusammenstürzen. Die Geschehnisse unterstrichen nachhaltig, wie wichtig eine widerspruchsfreie Fundierung der Mathematik tatsächlich war.

Zu den ersten, die sich der neugeborenen Herausforderung stellten, gehörten die britischen Mathematiker Bertrand Russell und Alfred North Whitehead. Sie starteten den Versuch, ein widerspruchsfreies Fundament zu errichten, auf dem die Mathematik für alle Zeiten einen sicheren Halt finden sollte. Nach zehn Jahren intensiver Arbeit war das Ergebnis greifbar: Die *Principia Mathematica* waren fertiggestellt (vgl. Abbildung 1.7). In einem dreibändigen Werk unternahmen Russell und Whitehead den Versuch, weite Bereiche der Mathematik mit den Mitteln der elementaren Logik formal herzuleiten. Ein großer Teil des Werks ist der *Typentheorie* gewidmet; einer widerspruchsfreien Konstruktion des Mengenbegriffs, mit dem die Art von Selbstbezug vermieden wird, die wenige Jahre zuvor die Mathematik in ihre größte Krise stürzte. Heute gilt die Typentheorie der Principia als überholt. An ihre Stelle tritt der formale axiomatische Aufbau der Mengenlehre durch Ernst Zermelo und Abraham Fraenkel, der die Russell'sche Antinomie ebenfalls beseitigt [32, 111].

Die mathematische Widerspruchsfreiheit ist eine unabdingbare Eigenschaft des mathematischen Schließens. Fehlt sie, so verkommt jedes formale System zu einem wertlosen Gedankengebilde. Warum dies so ist, wollen wir im Folgenden kurz begründen. Nehmen wir an, es gebe eine Aussage R, für die sich sowohl R als auch ihre Negation $\neg R$ innerhalb des Kalküls ableiten lassen. Die Situation erscheint wenig bedrohlich, wenn es sich um eine Aussage handelt, die uns nicht weiter interessiert. Eventuell ist R eine Aussage der Russell'schen Art, die uns ohnehin suspekt erscheint. Können wir den Kalkül vielleicht trotzdem sinnvoll einsetzen, wenn wir Aussagen dieser Form schlicht außen vor lassen?

Dass sich widersprüchliche Aussagen in einem Kalkül nicht isoliert betrachten lassen, liegt an den Schlussregeln der klassischen Logik. In Kapitel 3 werden Sie erlernen, wie sich das Theorem

$$\neg P \to (P \to Q)$$

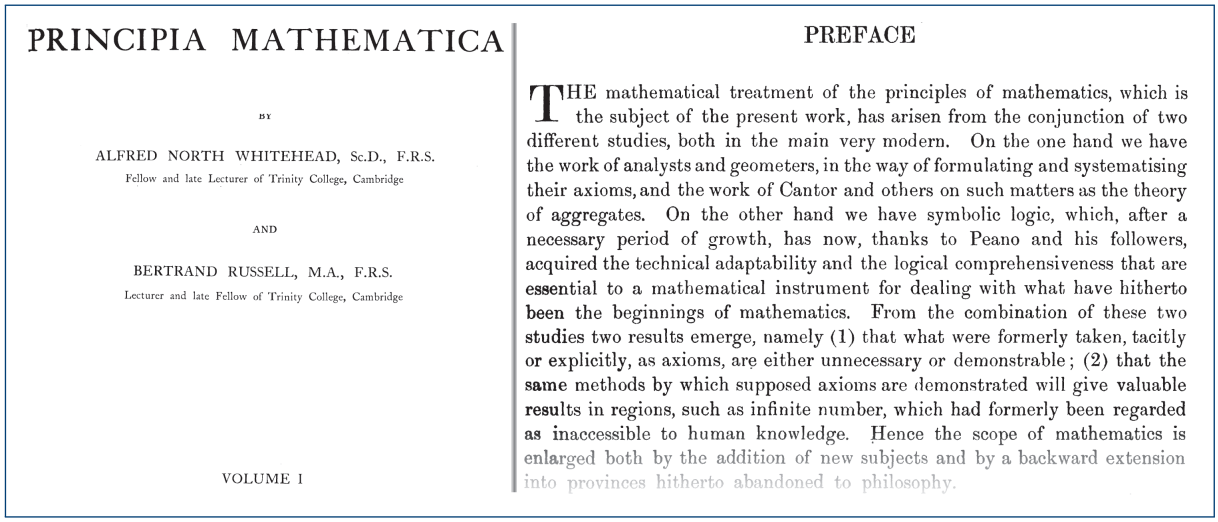
aus den elementaren Axiomen der Logik herleiten lässt. In Worten besagt es: Ist P falsch, so folgt aus der Wahrheit von P die Wahrheit von Q. Substituieren wir R für P, so erhalten wir das Theorem

$$\neg R \to (R \to Q).$$

Da $\neg R$ eine wahre Aussage ist, lässt sich mithilfe der *Abtrennungsregel* (*Modus ponens*) das Theorem

$$R \to Q$$

herleiten. Nach Voraussetzung ist R ebenfalls wahr, so dass eine erneute Anwendung der Abtrennungsregel das Theorem Q hervorbringt. Da die Wahl von Q keinen Einschränkungen unterliegt, können wir eine beliebige Aussage für Q substituieren. Kurzum: In einem widersprüchlichen Kalkül lassen sich ausnahmslos alle Aussagen als wahr beweisen.



PRINCIPIA MATHEMATICA

BY

ALFRED NORTH WHITEHEAD, Sc.D., F.R.S.
Fellow and late Lecturer of Trinity College, Cambridge

AND

BERTRAND RUSSELL, M.A., F.R.S.
Lecturer and late Fellow of Trinity College, Cambridge

VOLUME I

PREFACE

THE mathematical treatment of the principles of mathematics, which is the subject of the present work, has arisen from the conjunction of two different studies, both in the main very modern. On the one hand we have the work of analysts and geometers, in the way of formulating and systematising their axioms, and the work of Cantor and others on such matters as the theory of aggregates. On the other hand we have symbolic logic, which, after a necessary period of growth, has now, thanks to Peano and his followers, acquired the technical adaptability and the logical comprehensiveness that are essential to a mathematical instrument for dealing with what have hitherto been the beginnings of mathematics. From the combination of these two studies two results emerge, namely (1) that what were formerly taken, tacitly or explicitly, as axioms, are either unnecessary or demonstrable; (2) that the same methods by which supposed axioms are demonstrated will give valuable results in regions, such as infinite number, which had formerly been regarded as inaccessible to human knowledge. Hence the scope of mathematics is enlarged both by the addition of new subjects and by a backward extension into provinces hitherto abandoned to philosophy.

Abbildung 1.7: Die *Principia Mathematica*, erstmals erschienen in den Jahren 1910 bis 1913, ist eines der berühmtesten mathematischen Werke unserer Geschichte. Auf über 1800 Seiten, verteilt auf 3 Bände, unternahmen die Autoren den Versuch, alle mathematischen Erkenntnisse aus einer kleinen Menge von Axiomen systematisch herzuleiten.

Die Principia war in puncto Präzision jedem anderen Werk ihrer Zeit weit voraus. Sie fasste einen mathematischen Beweis als eine Folge von Regelanwendungen auf, durch die eine Aussage in endlich vielen Schritten aus einer festgelegten Menge von Axiomen abgeleitet wurde.

1.2.2 Metamathematik

Durch die zunehmende Beschäftigung mit den verschiedensten formalen Systemen entstand im Laufe der Zeit eine Metamathematik, die sich nicht mit der Ableitung von Sätzen *innerhalb* eines Kalküls beschäftigt, sondern mit Sätzen, die Aussagen *über* den Kalkül treffen. Drei Fragestellungen rückten in das Zentrum des Interesses:

- Vollständigkeit

  Ein formales System heißt *vollständig*, wenn jede wahre Aussage, die in der Notation des Kalküls formuliert werden kann, innerhalb desselben beweisbar ist. Mit anderen Worten: Für jede wahre Aussage P muss es eine endliche Kette von Regelanwendungen geben, die P aus den Axiomen deduziert. Beachten Sie, dass uns ein vollständiger Kalkül nicht preisgeben muss, wie eine solche Kette zu finden ist. Die Vollständigkeit garantiert lediglich deren Existenz.

- Widerspruchsfreiheit

  Ein formales System heißt *widerspruchsfrei*, wenn für eine Aussage P niemals gleichzeitig P und die Negation von P (geschrieben als $\neg P$) abgeleitet werden kann. Auf die immense Bedeutung der Widerspruchsfreiheit eines Kalküls sind wir weiter oben bereits eingegangen. Erfüllt ein formales System diese Eigenschaft nicht, so könnte es kaum wertloser sein. Es würde uns gestatten, jede beliebige Aussage zu beweisen.

- Entscheidbarkeit

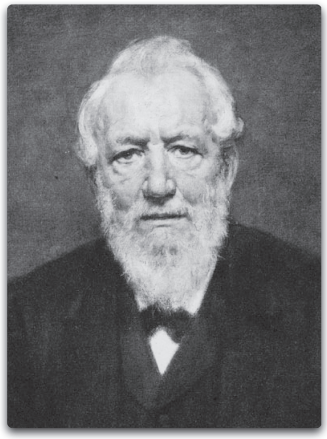
  Ein formales System heißt *entscheidbar*, wenn ein systematisches Verfahren existiert, mit dem für jede Aussage entschieden werden kann, ob sie innerhalb des Kalküls beweisbar ist. Hinter der Eigenschaft der Entscheidbarkeit verbirgt sich nichts Geringeres als der Wunsch nach einer mechanisierten Mathematik. Wäre z. B. die Zahlentheorie vollständig und entscheidbar, so ließe sich für jede wahre zahlentheoretische Aussage auf maschinellem Wege ein Beweis konstruieren. Der Traum eines jeden Mathematikers würde wahr.

Hilbert war überzeugt, dass eine vollständige, widerspruchsfreie und entscheidbare Axiomatisierung der Mathematik möglich sei. Im Jahr 1929 wurden seine Hoffnungen durch die Arbeiten des jungen Mathematikers Kurt Gödel zusätzlich genährt, als dieser in seiner Promotionsschrift die Vollständigkeit der Prädikatenlogik erster Stufe bewies [36]. Es war also möglich, einen Kalkül zu konstruieren, in dem sich jede wahre prädikatenlogische Formel in endlich vielen Schritten aus den Axiomen ableiten lässt. In diesen Tagen schien es nur eine Frage der Zeit zu sein, bis aus Hilberts Vermutungen Gewissheit werden würde.

1930 war das Jahr, in dem die Entwicklung eine abrupte Kehrtwende nehmen sollte. Am 8. September bekräftigte Hilbert vor der Versammlung Deutscher Naturforscher und Ärzte in seiner Heimatstadt Königsberg seine tiefe Überzeugung, dass es in der Wissenschaft keine unlösbaren Probleme gibt. Ein Auszug aus seiner Rede wurde in Form einer Radioansprache ausgestrahlt. Sie schließt mit den berühmten Worten:

> *„Wir dürfen nicht denen glauben, die heute mit philosophischer Miene und überlegenem Tone den Kulturuntergang prophezeien und sich in dem Ignorabimus gefallen. Für uns gibt es kein Ignorabimus, und meiner Meinung nach auch für die Naturwissenschaft überhaupt nicht. Statt des törichten Ignorabimus heiße im Gegenteil unsere Losung: Wir müssen wissen, wir werden wissen.“*

„Ignoramus et ignorabimus.“
(Wir wissen es nicht und wir werden es niemals wissen)



Emil Heinrich Du Bois-Reymond
(1818 – 1896)

„Für uns gibt es kein Ignorabimus“. Mit diesem Satz bekräftigte David Hilbert seine Haltung, dass es in den Naturwissenschaften keine unbeweisbaren Wahrheiten gibt. Der Begriff *Ignorabimus* wurde durch den deutschen Gelehrten Emil Heinrich Du Bois-Reymond geprägt. Durch seine Leipziger Rede vor der Versammlung Deutscher Naturforscher und Ärzte löste er im Jahr 1872 einen Richtungsstreit aus, der auf Jahre hinweg zu kontroversen Diskussionen in der Wissenschaftsgemeinde führen sollte. Er vertrat die Meinung, dass Begriffe wie das Bewusstsein niemals mit naturwissenschaftlichen Methoden erklärbar sein werden. Kurzum: Die Wissenschaft besitzt unüberwindbare Grenzen. *„Ich werde jetzt, wie ich glaube, in sehr zwingender Weise dartun, dass nicht allein bei dem heutigen Stand unserer Kenntnis das Bewusstsein aus seinen materiellen Bedingungen nicht erklärbar ist, was wohl jeder zugibt, sondern dass es auch der Natur der Dinge nach aus diesen Bedingungen nicht erklärbar sein wird.“* [8].

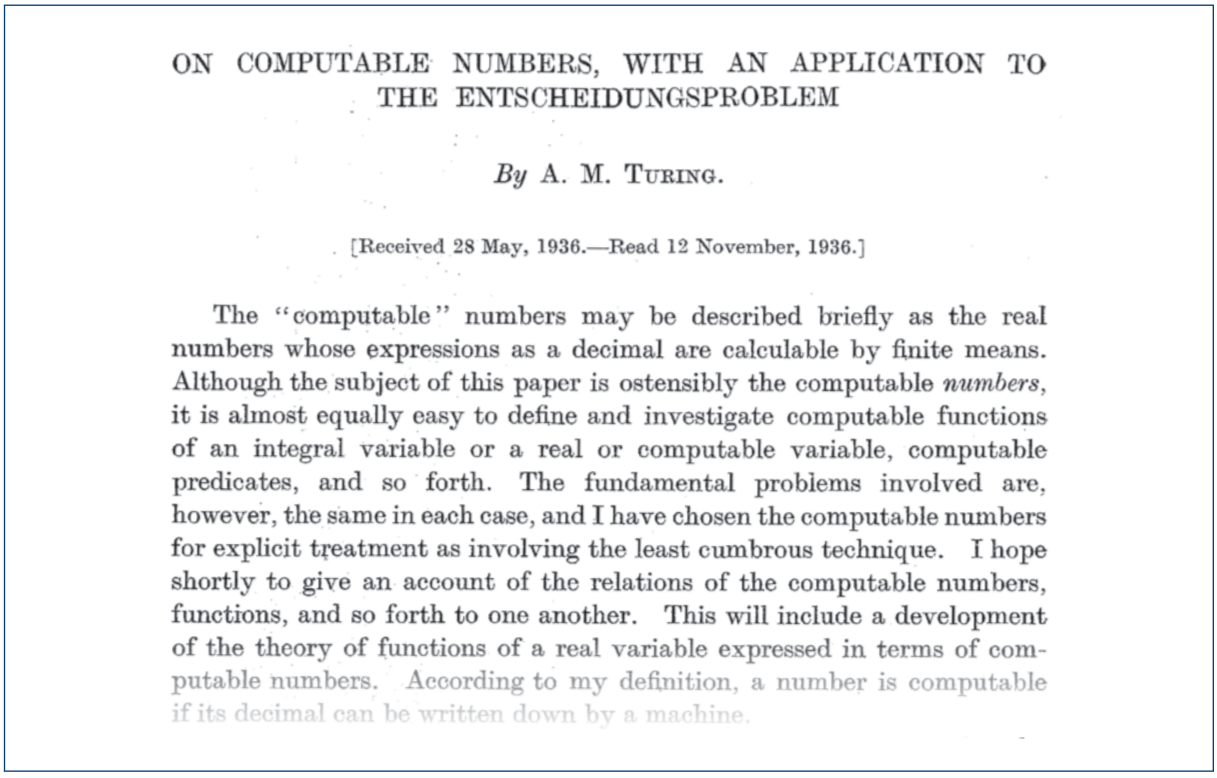
Der Unvollständigkeitsbeweis ist nicht nur aufgrund seiner inhaltlichen Tragweite von Bedeutung. Auch die trickreiche Beweisführung, mit der Gödel sein Resultat erzielte, zeugt von der Tiefgründigkeit des Ergebnisses. Gödel konnte zeigen, dass mathematische Schlussregeln, die Aussagen *über* Zahlen machen, selbst als Zahl verstanden werden konnten. Damit war es möglich, die Ebene der Zahlentheorie mit ihren Metaebenen zu vermischen. Aussagen sind nichts anderes als Zahlen, die selbst Aussagen über Zahlen tätigen. Auf diese Weise gelang es Gödel, Metaaussagen wie *„Aussage XYZ ist beweisbar“* innerhalb des Systems zu codieren.
Um die Unvollständigkeit zu beweisen, wandte Gödel einen Trick an. Er konstruierte Aussagen, die auf sich selbst Bezug nehmen und so eine Metaaussage über sich selbst beinhalten. Auf diese Weise gelang es ihm, eine Formel zu konstruieren, die der Metaaussage *„Diese Formel ist nicht beweisbar“* entspricht. Ist die Formel wahr, so lässt sie sich nicht beweisen und das zugrunde liegende Axiomensystem ist unvollständig. Ist sie falsch, so würde ein Beweis für eine falsche Aussage existieren und das Axiomensystem wäre nicht widerspruchsfrei. Mit anderen Worten: Erfüllt ein Axiomensystem die Eigenschaft der Widerspruchsfreiheit, so ist es zwangsläufig unvollständig.
Der Unvollständigkeitssatz zeigte zudem, dass die Widerspruchsfreiheit eines hinreichend aussagekräftigen formalen Systems nicht *innerhalb* des Systems selbst bewiesen werden kann. Gödel nutzte aus, dass in einem widersprüchlichen System alle Aussagen wahr sind, d. h., ein Kalkül ist genau dann widerspruchsfrei, wenn es eine einzige Aussage gibt, die nicht bewiesen werden kann. Gödel konnte jedoch zeigen, dass eine Aussage der Form *„es existiert eine unbeweisbare Aussage“* ebenfalls nicht innerhalb des Systems bewiesen werden kann.

Die Rede ist im Originalton erhalten und ein unschätzbares Dokument der Zeitgeschichte. Sie zeigt nachdrücklich, wie überzeugt Hilbert von der Durchführbarkeit seines ehrgeizigen Programms wirklich war.

Zum Zeitpunkt seiner Rede wusste Hilbert noch nichts von den Ereignissen, die sich am Vortag an anderer Stelle in Königsberg abspielten. Es war die große Stunde eines vierundzwanzigjährigen Mathematikers, der mit der Präsentation seines Unvollständigkeitssatzes die Mathematik aus den Angeln hob. Derselbe Kurt Gödel, der kurze Zeit zuvor die Vollständigkeit der Prädikatenlogik bewies, konnte zeigen, dass die Arithmetik aus fundamentalen Überlegungen heraus unvollständig sein musste. Sein Ergebnis war so allgemein, dass es auf jedes axiomatische System angewendet werden konnte, das ausdrucksstark genug ist, um die Zahlentheorie zu formalisieren. Damit war nicht nur gezeigt, dass der logische Apparat der Principia Mathematica unvollständig war, sondern auch, dass jeder Versuch, die Principia oder ein ähnliches System zu vervollständigen, von Grund auf zum Scheitern verurteilt ist. Gödel versetzte dem Hilbert'schen Programm einen schweren Schlag, von dem es sich nie erholen sollte.

Gödels Arbeit verwies die Mathematik zweifelsohne in ihre Grenzen, ließ jedoch Hilberts dritte Vermutung außen vor. Auch wenn wir nicht in der Lage sind, einen widerspruchsfreien und zugleich vollständigen Kalkül für die Theorie der Zahlen zu konstruieren, so könnte die Frage nach der Entscheidbarkeit eines Kalküls dennoch positiv beantwortet werden. Der Unvollständigkeitsbeweis schließt nicht aus, dass ein systematisches Verfahren existiert, das für jede Aussage bestimmt, ob es innerhalb des Systems beweisbar ist oder nicht.

Die Hoffnung, dass zumindest diese letzte Frage positiv beantwortet werden könnte, wurde im Jahr 1936 vollends zerstört, als der britische Mathematiker Alan Turing seine grundlegende Arbeit *On computable numbers, with an application to the Entscheidungsproblem* der Öffentlichkeit präsentierte (Abbildung 1.8). Turings Arbeit ist für die theoretische Informatik aus zweierlei Gründen von Bedeutung. Zum einen gelang es ihm als einem der Ersten, die Grenzen der Berechenbarkeit formal zu erfassen und das Entscheidungsproblem negativ zu beantworten; die Jagd nach dem mathematischen Perpetuum Mobile war zu Ende. Zum anderen konstruierte Turing für seinen Beweis ein abstraktes Maschinenmodell, das dem Funktionsprinzip moderner Computer bereits sehr nahe kam. Aus heutiger Sicht bildet das gedankliche Gebilde der *Turing-Maschine* die Nahtstelle zwischen der abstrakten Mathematik des frühen zwanzigsten Jahrhunderts und der Welt der realen Rechenmaschinen. In gewissem Sinne ersann Turing den *missing link*, der die Mathematik in Form des Computers zum Leben erweckte.



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

Abbildung 1.8: Im Jahr 1936 postulierte Alan Turing ein abstraktes Maschinenmodell, das die theoretische Informatik bis heute prägt [98]. Mit der *Turing-Maschine* legte er das formale Fundament, auf dem weite Teile der modernen Berechenbarkeitstheorien beruhen.

Turings theoretische Ergebnisse bilden den Kern der modernen Berechenbarkeitstheorie, die in der Informatik die gleiche Bedeutung besitzt wie die Relativitätstheorie in der Physik. Sie weist uns fundamentale Grenzen auf, die wir dem technologischen Fortschritt zum Trotz niemals werden überwinden können.

In der Tat existieren etliche reale Problemstellungen, die mit mechanisierten Verfahren nicht zu lösen sind. Ein Beispiel ist das *Halteproblem*, das wir in Kapitel 6 in all seinen Facetten untersuchen werden. Es besagt grob, dass kein mechanisiertes Verfahren existiert, mit dem wir immer korrekt entscheiden können, ob ein gegebenes Programm unter einer bestimmten Eingabe anhalten oder unendlich lange laufen wird. Die Unentscheidbarkeit des Halteproblems ist der Grund, dass selbst die modernsten Software-Compiler nicht zuverlässig entscheiden können, ob ein Programm eine Endlosschleife enthält oder nicht. Die Suche nach

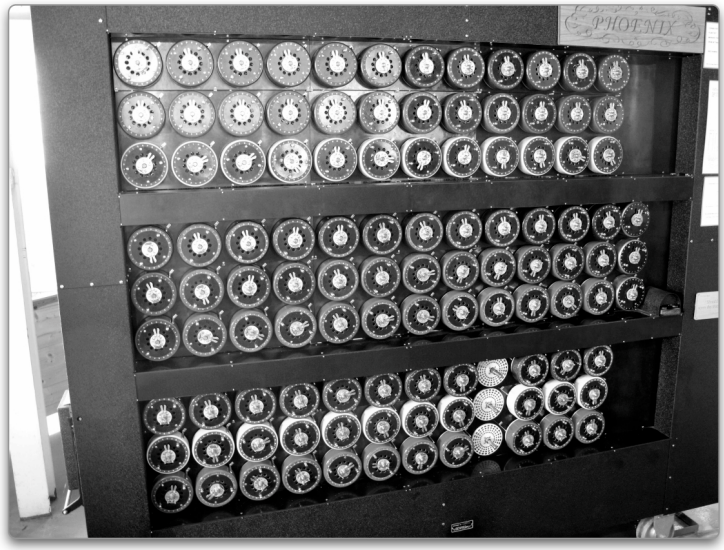


Abbildung 1.9: Die Turing-Bombe war eine Spezialkonstruktion des britischen Militärs, die im Zweiten Weltkrieg zur Dechiffrierung des deutschen Nachrichtenverkehrs eingesetzt wurde. Die elektromechanisch arbeitende Apparatur bestand aus mehreren rotierenden Trommeln, von denen jeweils 3 eine Einheit bildeten. Jede der insgesamt 36 Einheiten war in der Lage, die Verschlüsselung einer Enigma zu simulieren. Die erste Bombe wurde im März 1940 in Betrieb genommen und im August des gleichen Jahres durch eine leistungsfähigere Variante ersetzt. Mit ihr gelang es den Briten, den deutschen Code in wenigen Stunden zu brechen. Die Abbildung zeigt einen Nachbau der Turing-Bombe aus Bletchley Park.

einem entsprechenden Algorithmus wäre vergebens; die Arbeit von Turing lehrt uns, dass ein solcher aus fundamentalen Überlegungen heraus nicht existieren kann. In Kapitel 6 werden wir uns mit der Berechenbarkeitstheorie im Detail auseinandersetzen und auch den Aufbau und die Funktionsweise der Turing-Maschine ausführlich kennen lernen.

1.2.3 Die ersten Rechenmaschinen

Die folgenden Jahre standen ganz im Zeichen der aufkeimenden Computertechnik. Im Jahr 1941 konstruierte Konrad Zuse mit der Z3 einen voll funktionsfähigen Rechner aus ca. 2000 elektromechanischen Relais. Die Maschine wurde mit einer Taktfrequenz von 5 bis 10 Hertz betrieben, kam auf ein Gesamtgewicht von ca. 1000 kg und besaß eine Leistungsaufnahme von 4000 Watt. Rückblickend gehört der Bau der Z3 zu den bedeutendsten Meilensteinen auf dem Weg in das Informationszeitalter.

In den kommenden Jahren diktierte der politische Umbruch den Lauf der Dinge. Der aufkeimende Zweite Weltkrieg holte die Wissenschaft in die reale Welt zurück – eine Welt, in der Not und Furcht keinen Platz für „mathematische Spielereien“ ließen. Turing, der mit seinem abstrakten Rechnermodell ein paar Jahre zuvor die Grenzen der Berechenbarkeit auslotete, konnte sich den weltgeschichtlichen Ereignissen nicht entziehen. Auch für ihn war die Zeit gekommen, sich aus der theoretischen Gedankenwelt zu verabschieden. Zusammen mit einer Reihe anderer

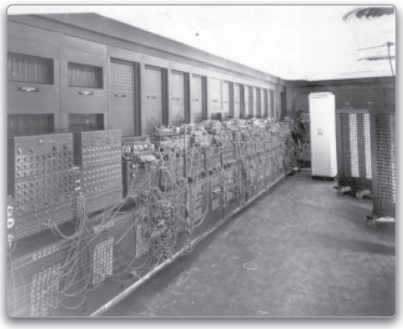
Wissenschaftler begab sich Turing nach Bletchley Park, einem idyllischen Landsitz in der Grafschaft Buckinghamshire. Dort suchte er als Mitglied eines geheimen Konsortiums im Dienst der britischen Regierung nach Möglichkeiten, um den militärischen Verschlüsselungscode der deutschen Truppen zu brechen.

Während des Zweiten Weltkriegs chiffrierte die deutsche Wehrmacht den Nachrichtenverkehr mithilfe der *Enigma* – einer Maschine, die auf dem Prinzip der *polyalphabetischen Substitution* beruht [6, 29, 48]. Obwohl die Funktionsweise zunächst primitiv wirkt, hätte die manuelle Dechiffrierung viele Monate in Anspruch genommen.

Um den deutschen Nachrichtenverkehr zeitnah zu entschlüsseln, konstruierten die britischen Wissenschaftler mit der *Turing-Bombe* eine Rechenmaschine, die auf das Brechen des Enigma-Codes spezialisiert war (Abbildung 1.9). Nach anfänglichen Schwierigkeiten gelang es den Briten schließlich, die Funksprüche der deutschen Truppen in nur noch wenigen Stunden zu entschlüsseln. Für Turing hatte der Bau dieser Apparatur einen ganz besonderen Aspekt. Auch wenn sie sich in vielen Punkten von seinem abstrakten Maschinenmodell unterschied, wurde ein großer Teil seiner Idee plötzlich real. Realer als ihm wahrscheinlich selbst lieb war, denn natürlich hatte auch er sich unter den gegebenen Umständen gewünscht, dass eine solche Maschine nie hätte gebaut werden müssen.

Die Turing-Bombe war kein Computer im heutigen Sinne, da sie für die Lösung einer ganz speziellen Aufgabe konzipiert war. Die erste voll funktionsfähige Rechenmaschine, die nahezu allen Definitionen des modernen Computer-Begriffs standhält, war der *Electronic Numerical Integrator and Computer*, kurz ENIAC (Abbildungen 1.10 bis 1.12). Der Rechnerkoloss wurde unter der Leitung von J. Presper Eckert und John W. Mauchly an der Moore School of Electrical Engineering der University of Pennsylvania gebaut und 1946 der Öffentlichkeit vorgestellt [40, 93]. Die ENIAC beeindruckte bereits aufgrund ihrer Größe. Sie bestand aus insgesamt 30 Einheiten, die U-förmig über eine eigens errichtete Halle verteilt angeordnet waren. Die ca. 18.000 verbauten Vakuumröhren der knapp 30 Tonnen wiegenden Apparatur verursachten eine Leistungsaufnahme von sagenhaften 174.000 Watt.

Intern arbeiten alle Computer im Binärsystem, d. h., jede Berechnung lässt sich auf eine Folge von Verknüpfungen der Binärziffern 0 und 1 zurückführen. Formal lassen sich die Vorgänge innerhalb eines Computers mithilfe der *Aussagenlogik* beschreiben – einem Teilgebiet der mathematischen Logik, das bereits sehr gut untersucht war, bevor der erste Computer das Konstruktionslabor verließ. Damit wurde die mathemati-

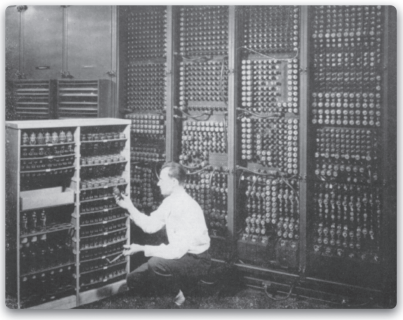


U.S.-Armee-Foto

Abbildung 1.10: Teilansicht der ENIAC (linke Raumhälfte)

U.S.-Armee-Foto

Abbildung 1.11: Teilansicht der ENIAC (rechte Raumhälfte)



U.S.-Armee-Foto

Abbildung 1.12: ENIAC-Techniker beim Austausch einer defekten Trioden-Röhre

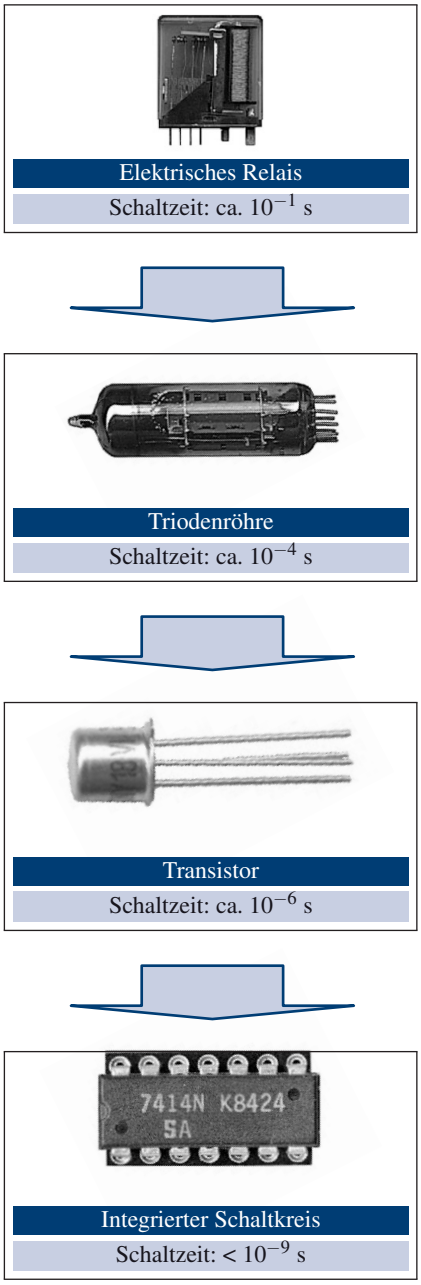


Abbildung 1.13: Von der ersten Rechenmaschine bis zum Supercomputer erlebte die Schaltkreis-Technik drei große Technologiewechsel.

sche Logik, die zu Beginn des Jahrhunderts die große Krise der Mathematik auslöste, plötzlich zur treibenden Kraft in der rasanten Fortentwicklung der Rechnertechnik. Seither ist die Logik nicht von der Informatik zu trennen. Sie war und ist die formale Grundlage für die Spezifikation, den Entwurf und die Analyse digitaler Hardware-Schaltungen. Kapitel 3 ist vollständig der Logik gewidmet und wird die grundlegende Bedeutung für die moderne Informatik aufzeigen.

Natürlich war es von der ENIAC bis zum modernen Computer noch ein langer Weg. Zur damaligen Zeit waren Rechenanlagen mit Triodenröhren bestückt, die den Entwicklern neben ihrer enormen Leistungsaufnahme vor allem wegen ihrer vergleichsweise geringen Zuverlässigkeit Kopfzerbrechen bereiteten. Erst durch die Erfindung des *Transistors* konnte die Computertechnik in Leistungsbereiche vordringen, die wir heute als selbstverständlich erachten (vgl. Abbildung 1.13). Der erste praxistaugliche Transistor wurde im Jahr 1948 von den Bell Laboratories in New York vorgestellt, allerdings sollten noch ein paar Jahre vergehen, bis er die alte Vakuumröhre vollständig verdrängen konnte. Der erste Computer auf Transistor-Basis wurde an der Manchester University Anfang der Fünfzigerjahre gebaut. Dem 1953 fertiggestellten Prototyp folgte eine deutlich erweiterte Variante, die zwei Jahre später in Betrieb genommen werden konnte.

1.2.4 Der Computer wird erwachsen

Der Übergang von der Röhre zum Transistor machte den Weg für die Computer der *zweiten Generation* frei. Die Technologietransition wurde von weiteren wichtigen Entwicklungen begleitet: Steuer- und Rechenwerke nahmen an Komplexität zu und bis dato fortschrittliche Konzepte wie die Verwendung indizierbarer Register oder der Einsatz von Gleitkomma-Hardware gehörten schnell zum Stand der Technik. Ferner hielt der *Ferritkernspeicher* Einzug, der bereits mehrere Kilobyte Daten aufnahm, allerdings in mühevoller Handarbeit gefertigt werden musste.

Die zunehmende Leistungsfähigkeit der konstruierten Computersysteme führte zu einer kontinuierlichen Veränderung in ihrer Bedienung. Wurden die Rechenmaschinen in den Pioniertagen noch direkt auf der Hardware-Ebene programmiert, arbeiteten auf den Computern der zweiten Generation bereits rudimentäre Betriebssysteme. Zeitgleich nahm die Entwicklung von Programmiersprachen an Fahrt auf. 1957 wurde der erste FORTRAN-Compiler ausgeliefert und 1960 die Programmiersprache COBOL verabschiedet. Verschiedene Spezialanwendungen sind auch heute noch in diesen Sprachen programmiert.

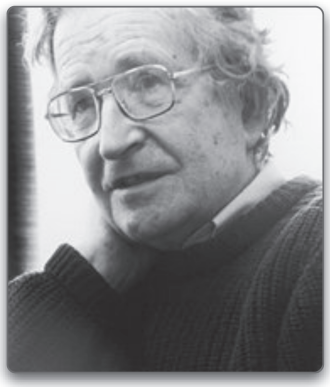
Hand in Hand mit der Evolution der Programmiersprachen wuchsen neue Teilbereiche heran, zu denen auch die Theorie der *formalen Sprachen* gehört. Dieser Teilbereich bündelt alle Methoden und Techniken, die sich mit dem Umgang mit künstlichen Sprachkonstrukten beschäftigen. Er stellt das theoretische Grundgerüst zur Verfügung, auf dem der moderne Compilerbau beruht [3].

Die Forschung auf diesem Gebiet wurde insbesondere durch die Arbeiten von Noam Chomsky (Abbildung 1.14) inspiriert, der im Jahr 1957 die *generative Linguistik* begründete [17]. Diese besagt im Kern, dass natürliche Sprachen elementaren Regeln unterliegen, die von uns unbewusst in der Analyse und Konstruktion von Sätzen eingesetzt werden. Auf diese Weise gelang es Chomsky zu erklären, dass wir richtige und falsche Sätze auch dann unterscheiden können, wenn wir sie noch nie in unserem Leben vorher gehört haben. Darüber hinaus führten Chomskys Arbeiten zu dem Ergebnis, dass die von Menschen gesprochenen Sprachen auf der unteren Ebene fast alle den gleichen Regeln unterliegen. Im Zuge seiner Analyse führte er den Begriff der *generativen Grammatik* ein, der kurze Zeit später Einzug in die theoretische Informatik hielt. Auch wenn Chomsky bei seiner Arbeit stets natürliche Sprachen im Sinn hatte, schuf er genau das Begriffsgerüst, das für den systematischen Umgang mit Computersprachen unabdingbar ist. In Kapitel 4 werden wir uns ausführlicher mit der Theorie der formalen Sprachen beschäftigen und die aus Sicht der Informatik wichtigsten Ergebnisse zusammenfassen.

Eng verwandt mit dem Gebiet der formalen Sprachen ist die *Automatentheorie*. In der Informatik beschreibt der Begriff des *endlichen Automaten* ein abstraktes Modell eines Zustandsübergangssystems, das in vielen Aspekten den Turing-Maschinen ähnelt, aber nicht an deren Ausdrucksstärke herankommt. Endliche Automaten stehen in einer engen Beziehung zu den formalen Sprachen, da sich die meisten Sprachklassen in ein äquivalentes Automatenmodell überführen lassen. In diesem Sinne wird der endliche Automat zu einer konkreten Beschreibungsform einer formalen Sprache. Moderne Compiler setzen Software-Implementierungen endlicher Automaten ein, um die Syntax eines Programmtextes zu überprüfen; Textverarbeitungen bedienen sich dieser Methodik, um Suchtexte in großen Texten effizient aufzuspüren.

Auch im Bereich des Hardware-Entwurfs spielt der endliche Automat eine zentrale Rolle. So lässt sich jede getaktete Hardware-Schaltung in einen funktional äquivalenten endlichen Automaten übersetzen und auf diese Weise einer formalen Analyse unterziehen. Umgekehrt beschreibt jeder endliche Automat eine Hardware-Schaltung und lässt sich automatisiert in ein *Schaltwerk* transformieren. Neben der Aussagenlo-

„There are two central problems in the descriptive study of language. One primary concern of the linguist is to discover simple and 'revealing' grammars for natural language. At the same time, by studying the properties of such successful grammars are clarifying the basic conceptions that underlie them, he hopes to arrive at a general theory of linguistic structure." [16]



Noam Chomsky
(1928)

Abbildung 1.14: Viele Erkenntnisse über formale Sprachen gehen auf die Arbeiten des amerikanischen Linguisten Noam Chomsky zurück. Bereits in den frühen Jahren seiner akademischen Laufbahn suchte Chomsky einen theoretischen Zugang zur Linguistik. Ein Kernelement seiner Herangehensweise war die Verwendung formaler Grammatiken – Metasprachen, die einen rekursiven Aufbau der Sprachausdrücke aus einer Menge von Grundelementen erlauben. Der formale Charakter seiner Methodik machte es erstmals möglich, mathematisch präzise Aussagen über linguistische Sprachen zu treffen. Insbesondere gelang es Chomsky, Sprachen anhand gewisser Eigenschaften ihrer Metasprache in verschiedene Typklassen einzuteilen. Die entstehende *Chomsky-Hierarchie* gehört heute zu den wichtigsten Klassifikationsmerkmalen formaler Sprachen.

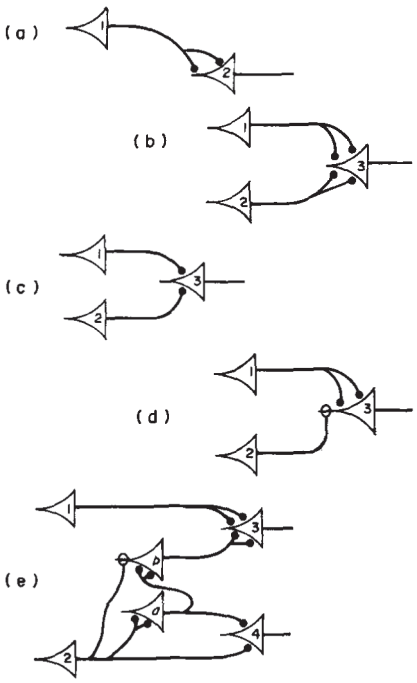


Abbildung 1.15: Nervennetze von McCulloch und Pitts [69]

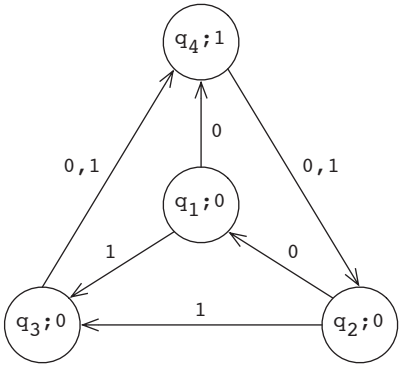


Abbildung 1.16: Endlicher Automat aus der Originalarbeit von Moore [75]

gik ist die Automatentheorie die zweite tragende Säule des gesamten Hardware-Entwurfs. Aufgrund der großen Bedeutung widmet sich Kapitel 5 ausschließlich diesem Teilgebiet der theoretischen Informatik.

Im Gegensatz zu vielen anderen Begriffen der theoretischen Informatik ist die Entstehungsgeschichte des endlichen Automaten nicht an ein einzelnes Ereignis gebunden. Vielmehr haben wir es mit einem Begriff zu tun, der in einem evolutionären Prozess Schritt für Schritt entwickelt wurde. Zu den bemerkenswertesten Vorgängern zählen die *Nervennetze*, die im Jahr 1943 von Warren McCulloch und Walter Pitts zur Untersuchung neuronaler Strukturen eingeführt wurden (vgl. Abbildung 1.15). Die Arbeit ist für die Entwicklung des Automatenbegriffs in zweierlei Hinsicht von Bedeutung. Zum einen finden wir im Neuronenmodell von McCulloch und Pitts bereits viele Konzepte vor, die auch den modernen endlichen Automaten auszeichnen. Zum andere besaß die Arbeit für viele andere Wissenschaftler eine geradezu inspirierende Wirkung. So auch für den US-amerikanischen Mathematiker Stephen Cole Kleene, der das Nervennetz im Jahr 1956 zu einem allgemeinen Zustandsübergangsmodell weiterentwickelte [60]. In diesem Zusammenhang führte Kleene die *regulären Ausdrücke* ein, die bis heute die Standardnotation für die Beschreibung von Suchmustern bilden. Weitere bedeutende Beiträge zur Automatentheorie wurden nahezu zeitgleich von George H. Mealy und Edward F. Moore publiziert [70, 75]. Aus Moores Arbeit stammt unter anderem das grafische Transitionsmodell, das wir auch heute noch zur Darstellung endlicher Automaten verwenden (vgl. Abbildung 1.16).

Den vielleicht größten Beitrag zur Automatentheorie lieferten Michael Oser Rabin und Dana Scott im Jahr 1959. Unter anderem sorgten sie für eine klare Abgrenzung zwischen endlichen Automaten und Turing-Maschinen. Hierzu wiesen sie nach, dass sich die zahlreichen, im Laufe der Zeit entstandenen Automatenmodelle allesamt aufeinander abbilden lassen und damit eine geringere Ausdrucksstärke besitzen als die Turing-Maschine. Des Weiteren führten sie das Konzept des Nichtdeterminismus ein, mit dem sie eine völlig neue Denkrichtung im Bereich der Berechenbarkeitstheorie schufen. Im Jahr 1976 wurden Rabin und Scott für ihre bedeutende Arbeit mit dem Turing-Award ausgezeichnet (Tabelle 1.1).

1.2.5 Berechenbarkeit versus Komplexität

Die zunehmende Rechenleistung machte es in der zweiten Hälfte des zwanzigsten Jahrhunderts möglich, auch komplexe Probleme computer-

Jahr	Preisträger
1966	Alan J. Perlis
1967	Maurice V. Wilkes
1968	Richard Hamming
1969	Marvin Minsky
1970	James H. Wilkinson
1971	John McCarthy
1972	Edsger W. Dijkstra
1973	Charles W. Bachman
1974	Donald E. Knuth
1975	Allen Newell,
	Herbert A. Simon
1976	Michael O. Rabin,
	Dana S. Scott
1977	John Backus
1978	Robert W. Floyd
1979	Kenneth E. Iverson
1980	C. Antony R. Hoare
1981	Edgar F. Codd
1982	Stephen A. Cook
1983	Ken Thompson,
	Dennis M. Ritchie
1984	Niklaus E. Wirth
1985	Richard M. Karp
1986	John E. Hopcroft,
	Robert E. Tarjan
1987	John Cocke
1988	Ivan Sutherland
1989	William Kahan
1990	Fernando J. Corbató
1991	Robin Milner
1992	Butler W. Lampson
1993	Juris Hartmanis,
	Richard E. Stearns
1994	Edward Feigenbaum,
	Raj Reddy
1995	Manuel Blum
1996	Amir Pnueli
1997	Douglas Engelbart
1998	Jim Gray
1999	Frederick P. Brooks, Jr.
2000	Andrew Chi-Chih Yao
2001	Ole-Johan Dahl,
	Kristen Nygaard
2002	Leonard M. Adleman
	Ronald L. Rivest,
	Adi Shamir
2003	Alan Kay
2004	Vinton G. Cerf,
	Robert E. Kahn
2005	Peter Naur
2006	Frances E. Allen
2007	Edmund M. Clarke,
	E. Allen Emerson,
	Joseph Sifakis
2008	Barbara Liskov
2009	Charles P. Thacker

Tabelle 1.1: Der Turing-Award existiert seit 1966 und wird seitdem jährlich vergeben. Er ist die höchste Auszeichnung im Bereich der Informatik und hat eine ähnliche Bedeutung wie der Nobelpreis in anderen Wissenschaftsdisziplinen.

gestützt zu bearbeiten. Die Forschung auf dem Gebiet der Algorithmentechnik nahm an Fahrt auf und brachte immer neue Rechenvorschriften hervor, mit denen Probleme aus ganz unterschiedlichen Anwendungsbereichen maschinell gelöst werden konnten. Mit der Fähigkeit, immer größere Eingabemengen verarbeiten zu können, rückten die Laufzeit und der Platzverbrauch eines Programms stärker in das Bewusstsein der Soft- und Hardware-Entwickler. Damit ein Algorithmus für die Lösung praktischer Probleme eingesetzt werden konnte, musste er nicht nur *effektiv* sein, d. h. stets das korrekte Ergebnis berechnen, sondern auch *effizient*. Anders als in den Pioniertagen der theoretischen Informatik, in denen die Berechenbarkeitstheorie zu ihrer vollen Blüte heranreifte, war es nicht mehr länger ausreichend, die prinzipielle Lösbarkeit eines Problems zu zeigen. Eine algorithmische Lösung war faktisch nutzlos, wenn ein Computer mehrere Jahre oder Jahrzehnte für ihre Ausführung benötigte.

■ Problem

■ Algorithmus

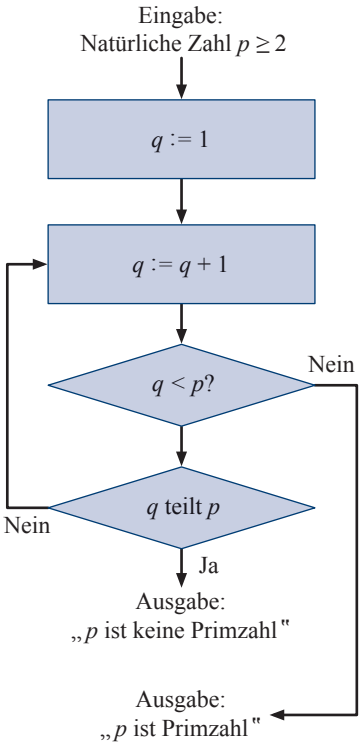


Abbildung 1.17: Das Problem PRIME lässt sich auf einfache Weise lösen, indem wir die Eingabe p nacheinander durch alle Zahlen q mit $2 \leq q < p$ teilen. Geht die Division niemals ohne Rest auf, so ist p eine Primzahl. Während das Programm für kleine Zahlen praktikabel arbeitet, nimmt die Rechenzeit für größere Werte von p drastisch zu. Kurzum: Der abgebildete Algorithmus ist effektiv, aber nicht effizient.

Die praktische Verwertbarkeit von Algorithmen lenkte die Aufmerksamkeit der Wissenschaftsgemeinde langsam, aber sicher von der Berechenbarkeitstheorie weg und hin zur Komplexitätstheorie. Die Grundlagen für die Untersuchung der Laufzeit- und Platzkomplexität von Algorithmen wurden in den Sechzigerjahren geschaffen [21, 26] und nachhaltig durch die Arbeit von Juris Hartmanis und Richard Stearns aus dem Jahr 1965 beeinflusst [42]. Mit ihren mathematisch präzisen Komplexitätsuntersuchungen von Turing-Maschinen legten die beiden US-amerikanischen Computerwissenschaftler den Grundstein für die formale Komplexitätstheorie, wie wir sie heute kennen.

Eine kleine Auswahl konkreter Beispiele soll verdeutlichen, mit welchen Fragestellungen sich dieser Zweig der theoretischen Informatik im Kern befasst. Als Erstes betrachten wir mit PRIME ein Problem der Zahlentheorie, das Mathematiker seit tausenden von Jahren beschäftigt. Für eine beliebige natürliche Zahl p gilt es zu entscheiden, ob es sich um eine Primzahl handelt oder nicht. Dass PRIME lösbar ist, liegt auf der Hand, schließlich können wir die Zahl p als Primzahl identifizieren, indem wir p nacheinander durch alle potenziellen Faktoren dividieren. Abbildung 1.17 fasst die Vorgehensweise in einem Flussdiagramm zusammen. Obwohl der konstruierte Algorithmus für alle Eingabewerte die korrekte Antwort liefert, nimmt die Rechenzeit für größere Eingabewerte dramatisch zu. Bezeichnet n die Anzahl der Bits in der Binärdarstellung von p, so müssen im schlimmsten Fall 2^n Divisionen berechnet werden, bis der Algorithmus terminiert. Selbst wenn wir eine Billion Einzeldivisionen pro Sekunde ausführen könnten, stünde das Ergebnis für eine 128-Bit-Binärzahl erst nach ca. 10^{19} Jahren fest. Wir müssten damit länger auf die Antwort warten, als unser Universum jemals existieren wird.

Als Nächstes wenden wir uns mit SAT einem Problem der Aussagenlogik zu (vgl. Abschnitt 3.1). Konkret geht es um die Frage, ob wir die Variablen einer aussagenlogischen Formel F so mit den Wahrheitswerten 1 (wahr) und 0 (falsch) belegen können, dass F wahr wird. Eine Formel mit dieser Eigenschaft heißt *erfüllbar* (*satisfiable*).

Genau wie PRIME lässt sich auch SAT verblüffend einfach lösen. Da eine Formel F nur endlich viele aussagenlogische Variablen enthält und diese ausschließlich die Werte 1 oder 0 annehmen können, ist die Anzahl der möglichen Variablenbelegungen endlich. Somit können wir die Erfüllbarkeit einer Formel F entscheiden, indem wir alle Belegungskombinationen der Reihe nach durchprobieren (vgl. Abbildung 1.18). Genau dann, wenn wir eine Kombination finden, für die F den Wahrheitswert 1 annimmt ($F \equiv 1$), erfährt der Erfüllbarkeitstest eine positive Antwort. Wie schon im Falle von PRIME ist der gefundene Algorith-

mus effektiv, aber nicht effizient. Bezeichnet n die Anzahl der aussagenlogischen Variablen in F, so müssen im schlimmsten Fall alle 2^n Wertekombinationen betrachtet werden, um eine zuverlässige Aussage über die Erfüllbarkeit von F zu treffen.

Das Problem SAT spielt in der theoretischen Informatik eine weit größere Rolle, als es der ein oder andere Leser an dieser Stelle vermuten mag. In Abschnitt 7.3.3 werden wir zeigen, dass viele Algorithmen auf das Erfüllbarkeitsproblem reduziert werden können, indem der Programmablauf in eine aussagenlogische Formel hineincodiert wird. In diesem Sinne wird sich SAT als Universalproblem erweisen, aus dem sich viele Ergebnisse der modernen Komplexitätstheorie direkt ableiten lassen.

Obwohl die gesamte theoretische Informatik in der zweiten Hälfte des zwanzigsten Jahrhunderts durch die Erkenntnis geprägt war, dass ein praktisch verwertbarer Algorithmus nicht nur effektiv, sondern auch effizient arbeiten muss, waren die Fortschritte auf dem Gebiet der Algorithmentechnik unterschiedlicher Natur. Einige Probleme konnten mit Algorithmen gelöst werden, die auch für große Eingabelängen sehr schnell zu einem Ergebnis führten, andere widersetzten sich vehement einer effizienten Lösung. Es schien, als ob die untersuchten Problemstellungen individuelle Schwierigkeitsgrade besäßen, die selbst mit den größten Anstrengungen nicht umgangen werden konnten.

Wissenschaftler reagierten mit der Definition von *Komplexitätsklassen*, in die sich die untersuchten Problemstellungen anhand ihrer bisher bekannten Lösungsstrategien einordnen ließen. Die weiter oben eingeführten Algorithmen zur Lösung von PRIME und SAT sind sogenannte *Exponentialzeitalgorithmen*, da ihre Rechenzeit exponentiell mit dem Größenparameter n ansteigt. Probleme, für die ausschließlich Exponentialzeitalgorithmen bekannt sind, gelten in der Praxis als unlösbar.

Eine ungleich größere Bedeutung besitzen die *Polynomialzeitalgorithmen*, deren Rechenzeit durch ein Polynom $p(n)$ nach oben begrenzt ist. Da Polynome für steigende Werte von n deutlich langsamer gegen unendlich streben als Exponentialfunktionen, setzen viele Experten die Existenz eines Polynomialzeitalgorithmus mit der praktischen Lösbarkeit eines Problems gleich.

Welche Komplexität sich hinter einem spezifischen Problem verbirgt, ist diesem nicht unmittelbar anzusehen und bereits kleine Veränderungen der Fragestellung können zu einer schlagartigen Änderung der Problemkomplexität führen. Wir wollen dieses Phänomen am Beispiel des *Königsberger Brückenproblems* ergründen, das sich mit der Frage beschäftigt, ob in einem gegebenen Graph G ein *Euler-Kreis* existiert.

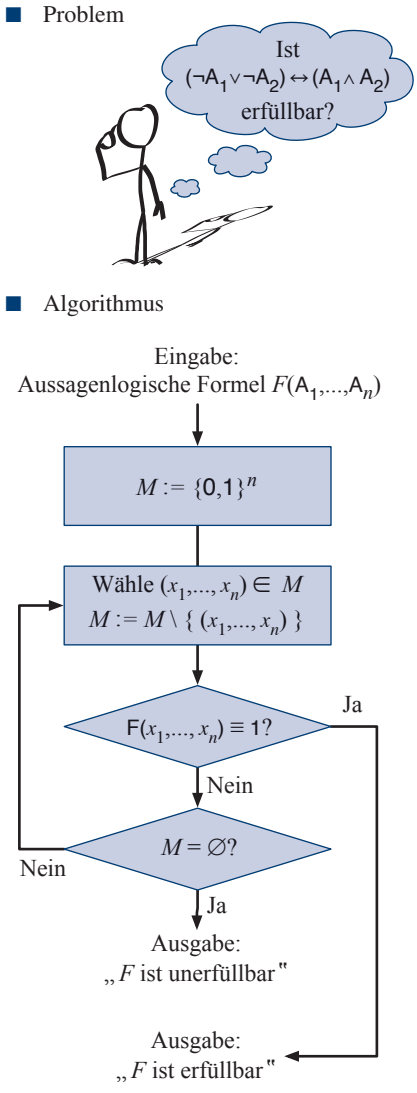
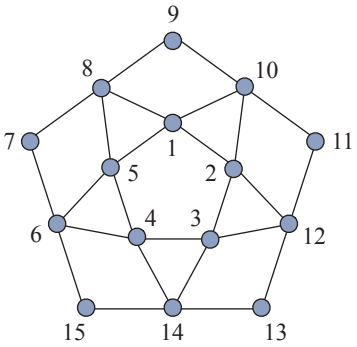
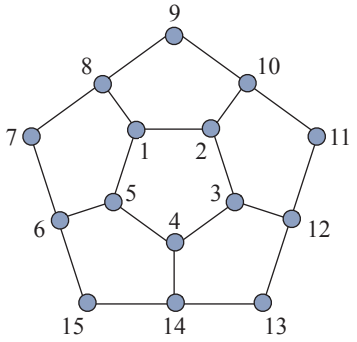


Abbildung 1.18: Das Problem SAT lässt sich lösen, indem der Funktionswert von F nacheinander für sämtliche Variablenbelegungen ausgerechnet wird. Da die Anzahl der Belegungen exponentiell mit der Anzahl der Variablen zunimmt, bleibt die praktische Anwendung dieser Methode auf Formeln mit wenigen Variablen beschränkt. Genau wie im Falle von PRIME ist der Algorithmus effektiv, aber nicht effizient.

■ Graph 1



■ Graph 2



■ Graph 3

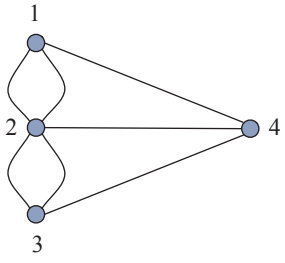


Abbildung 1.19: Hinter dem Königsberger Brückenproblem versteckt sich eine elementare Graph-Eigenschaft, die sich anhand des Euler-Kriteriums effizient nachweisen lässt. Von den dargestellten Graphen erfüllt nur der erste das Euler-Kriterium.

Plakativ gesprochen liegt ein solcher genau dann vor, wenn wir G auf einem Rundweg so durchlaufen können, dass alle Kanten genau einmal besucht werden. 1736 bewies der Schweizer Mathematiker Leonhard Euler, dass ein solcher Kreis genau dann existiert, wenn jeder Knoten eine gerade Anzahl ausgehender Kanten besitzt [7, 30].

Damit können wir das Problem für einen beliebigen Graphen mit n Knoten in linearer Zeit lösen, indem wir nacheinander die Anzahl der ausgehenden Kanten bestimmen. Genau dann, wenn wir in einem der Knoten eine ungerade Anzahl vorfinden, ist die Antwort negativ; einen Euler-Kreis kann es in diesem Fall nicht geben. Abbildung 1.19 demonstriert das Vorgehen anhand zweier Beispielgraphen, von denen nur der erste das Euler-Kriterium erfüllt. In den anderen Graphen existiert kein Euler-Kreis, da gleich mehrere Knoten eine ungerade Anzahl ausgehender Kanten aufweisen.

Euler motivierte seine Ergebnisse, indem er auf die besondere geografische Lage der Stadt Kaliningrad, das frühere Königsberg, zurückgriff. Die ehemalige Hauptstadt Ostpreußens wird durch den Fluss Pregel in vier Gebiete unterteilt, die zu Eulers Zeiten über 7 Brücken miteinander verbunden waren (vgl. Abbildung 1.20). Euler konnte mithilfe seiner Untersuchungen beweisen, dass es unmöglich war, Königsberg auf einem Rundweg zu erkunden, der jede Brücke exakt einmal besucht. Hierzu brauchte er nichts weiter zu tun, als die Kartentopologie in einen Graphen zu übersetzen, in dem jedes Stadtgebiet einem Knoten und jede Brücke einer Kante entspricht. Den entstehenden Graphen haben wir bereits kennen gelernt: Er ist mit dem dritten Beispiel aus Abbildung 1.19 identisch. Da dieser Graph keinen Euler-Kreis besitzt, kann es im ehemaligen Königsberg keinen entsprechenden Rundgang geben.

Für die Komplexitätstheorie ist das Königsberger Brückenproblem von großer Bedeutung, da eine geringfügige Abwandlung ein faktisch unlösbares Problem entstehen lässt. Es wurde im Jahr 1859 von Sir William Hamilton formuliert und ist dem Königsberger Brückenproblem zum Verwechseln ähnlich. Für einen gegebenen Graphen G ist zu entscheiden, ob wir diesen auf einem Rundweg so durchlaufen können, dass alle *Knoten* genau einmal besucht werden. Einen Weg mit dieser Eigenschaft bezeichnen wir als *Hamilton-Kreis*. Bezogen auf die Stadtkarte von Königsberg lautet das Hamilton-Problem wie folgt: Gibt es einen Rundweg durch die Stadt, so dass kein Stadtteil zweimal betreten wird? In unserem konkreten Beispiel reicht ein gezielter Blick, um einen Hamilton-Kreis zu erkennen. Starten wir beispielsweise im Norden, so gelangen wir über die Pregel-Insel in den südlichen Stadtteil. Anschließend können wir über den östlichen Teil in den Norden zurückkehren, ohne die Insel erneut zu betreten.

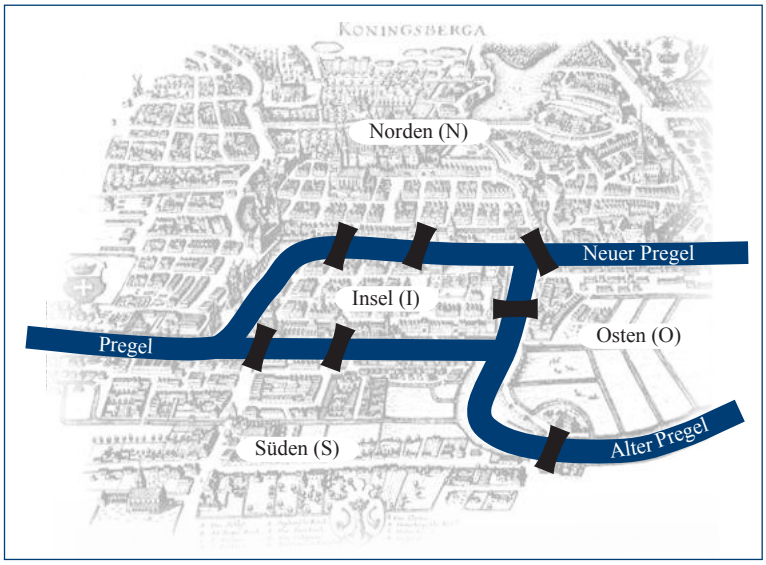


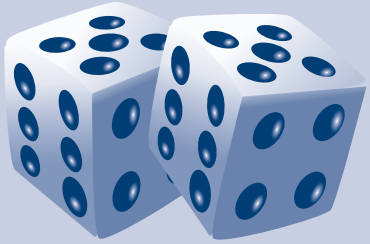
Abbildung 1.20: Im achtzehnten Jahrhundert wurde die Stadt Königsberg durch den Fluss Pregel in vier Gebiete aufgeteilt, die durch sieben Brücken miteinander verbunden waren. Bekannt wurde das Stadtbild durch den Schweizer Mathematiker Leonhard Euler, der es zur Veranschaulichung eines von ihm gelösten Graphenproblems benutzte. Er motivierte seine Arbeit mit der Frage, ob Königsberg auf einem Rundweg erkundet werden kann, auf dem jede Brücke genau einmal überquert wird. Im Jahr 1736 bewies er, dass ein solcher Weg nicht existiert. Mit seinen Untersuchungen begründete Euler die Graphentheorie, die heute sowohl in der theoretischen als auch in der praktischen Informatik ihren festen Platz eingenommen hat.

Obwohl die Lösung des Hamilton-Problems für kleine Graphen wie eine Fingerübung wirkt, ist es noch niemandem gelungen, einen deterministischen Polynomialzeitalgorithmus dafür zu formulieren. Ob ein solcher Algorithmus überhaupt existieren kann, ist gegenwärtig unbekannt. Um es vorwegzunehmen: Die Chancen, dass sich das Hamilton-Problem auf realen Rechenanlagen in Polynomialzeit lösen lässt, stehen aus heutiger Sicht nicht allzu gut, da es in die Klasse der *NP-vollständigen* Probleme fällt.

Die Klasse der NP-vollständigen Probleme ist eine Teilmenge der Klasse NP. Probleme aus NP zeichnen sich dadurch aus, dass eine potenzielle Lösung sehr einfach auf Korrektheit geprüft werden kann, das Finden derselben jedoch ungleich schwerer ist. Am Beispiel des Hamilton-Problems lässt sich diese Eigenschaft besonders gut beobachten. Ist eine Sequenz von Knoten vorgegeben, so können wir leicht feststellen, ob diese einen Hamilton-Kreis beschreibt. Die Berechnung der Knotensequenz gestaltet sich dagegen deutlich komplexer. Die Teilmenge der NP-vollständigen Probleme vereint alle Elemente aus NP, die mächtig genug sind, um damit alle anderen NP-Probleme ebenfalls zu lösen. Sie gehören damit zu den schwersten Problemen überhaupt und eine effiziente Lösung eines einzigen NP-vollständigen Problems würde die effiziente Lösbarkeit aller anderen Probleme aus NP nach sich ziehen.

Für diesen Moment soll uns diese vage Beschreibung des Vollständigkeitsbegriffs genügen. Eine genaue Definition werden wir in Kapitel 7

Die Rechenzeit eines Exponentialzeitalgorithmus nimmt so schnell zu, dass seine Praxistauglichkeit deutlich eingeschränkt ist. Um größere Eingaben verarbeiten zu können, wurden für viele Probleme *randomisierte Algorithmen* entwickelt. Diese verfolgen die Grundidee, eine gewünschte Eigenschaft nicht für *alle*, sondern nur für die *meisten* Eingaben zu garantieren. Auf diese Weise entstehen Algorithmen, die aus theoretischer Sicht Lücken aufweisen, in der Praxis aber gute Ergebnisse liefern. Randomisierte Algorithmen lassen sich in zwei Klassen einteilen:
Las-Vegas-Algorithmen berechnen immer das korrekte Ergebnis, konsumieren für vereinzelte Eingaben jedoch überproportional viel Rechenzeit. In diese Gruppe fällt der bekannte Sortieralgorithmus Quicksort. Im statistischen Mittel benötigt dieser $n \cdot \log n$ Operationen, um eine Liste mit n Elementen zu sortieren. Bei einer ungünstigen Werteverteilung steigt die Laufzeit dagegen quadratisch mit der Anzahl der zu sortierenden Elemente an.
Monte-Carlo-Algorithmen garantieren für alle Eingaben die gleiche Laufzeitkomplexität, liefern dafür in manchen Fällen eine falsche Antwort. Mit dem Primzahltest von Robert Solovay und Volker Strassen enthält diese Klasse einen Algorithmus, der vor allem im Zusammenhang mit RSA-Kryptosystemen einen hohen Bekanntheitsgrad erreichte [92].
Der Primzahltest ist deutlich schneller als alle nicht-randomisierten Verfahren, in einzelnen Fällen werden jedoch auch faktorisierbare Zahlen als Primzahl klassifiziert.



nachholen, sobald das zum Verständnis notwendige Begriffsgerüst geschaffen wurde. So viel vorweg: Ob überhaupt ein NP-vollständiges Problem existiert, war lange Zeit unbekannt. Erst im Jahr 1971 gelang Stephen Cook und Leonid Levin unabhängig voneinander der Nachweis, dass das weiter oben skizzierte SAT-Problem NP-vollständig ist [24, 65, 66].

Ein Jahr später wies Richard Karp die NP-Vollständigkeit von 20 weiteren Problemen nach [54] und das im Jahr 1979 publizierte Buch von Michael Garey und David Johnson enthält bereits eine Zusammenfassung von ca. 300 NP-vollständigen Problemen. Können wir für nur ein einziges einen Polynomialzeitalgorithmus konstruieren, so sind alle anderen NP-vollständigen Probleme auf einen Schlag ebenfalls in Polynomialzeit lösbar. Damit steht fest: Sollte ein solcher Algorithmus tatsächlich irgendwann gefunden werden, wären seine Auswirkungen bis in alle Teilbereiche der Informatik hinein spürbar.

Die steigende Anzahl gefundener Probleme, die sich vehement einer effizienten Lösbarkeit entziehen, nährt jedoch in vielen Experten die Vermutung, dass es unmöglich ist, einen Polynomialzeitalgorithmus für ein NP-vollständiges Problem zu finden. Ein mathematischer Beweis dafür steht aber bis heute aus und wir dürfen gespannt sein, was die zukünftige Forschung in diesem Bereich an Überraschungen hervorbringen wird.

1.3 Theoretische Informatik heute

Die theoretische Informatik gehört zu den wenigen Teilgebieten der Informationswissenschaft, die über einen gefestigten Stoffumfang verfügen. Insbesondere im Vergleich mit der sich kontinuierlich wandelnden Software-Technik wirken die Erkenntnisse und Methoden überaus stabil. Trotzdem handelt es sich bei der theoretischen Informatik keinesfalls um einen abgeschlossenen Wissenschaftszweig und viele Problemstellungen sind nach wie vor ungelöst.

Eine davon ist die endgültige Klärung der weiter oben diskutierten Frage, ob sich NP-vollständige Probleme auf realen Rechenanlagen in Polynomialzeit lösen lassen. Dieses *P-NP-Problem* ist von so großer Bedeutung, dass es in die Riege der 7 *Millenium-Probleme* aufgenommen wurde (Abbildung 1.21). Diese wurden am 24. Mai 2000 in Paris vorgestellt; in derselben Stadt, in der David Hilbert hundert Jahre zuvor in seiner historischen Rede die 23 dringlichsten Probleme der Mathematik formulierte. Die Forschung auf diesem Gebiet ist durchaus attraktiv.

Für jedes der 7 Millenium-Probleme hat das in Cambridge beheimatete Clay Mathematics Institute (CMI) ein Preisgeld von einer Million US-Dollar ausgelobt.

Wie auch immer die Antwort auf das P-NP-Problem lauten wird: Die Folgen sind nicht nur positiver Natur. Gelingt es, die heute mehrheitlich gehegte Vermutung zu bestätigen, dass sich NP-vollständige Probleme nicht in Polynomialzeit lösen lassen, so wäre die Existenz theoretisch beantwortbarer, aber praktisch unlösbarer Fragestellungen gesichert. Wir wüssten dann, dass Probleme wie das Hamilton'sche niemals effizient gelöst werden können. Doch auch die gegenteilige Antwort würde uns Kopfzerbrechen bereiten, da sich viele sicherheitskritische Algorithmen auf die praktische Unlösbarkeit NP-vollständiger Probleme stützen. Sollten wir jemals in der Lage sein, auch nur eine einzige NP-vollständige Fragestellung in Polynomialzeit zu beantworten, so ließen sich neben dem Hamilton-Problem auch Algorithmen aus dem Gebiet der Kryptographie effizient lösen. Die im Internetzeitalter so unabdingbar gewordene Verschlüsselung stände mit einem Schlag auf wackligen Füßen.

Wie Sie sehen, ist die theoretische Informatik weit mehr als mathematische Spielerei. Sie besitzt weitreichende Konsequenzen, die sich in alle Bereiche der Informatik auswirken. Dass die theoretische Informatik keine tote Wissenschaft ist, wird durch Ergebnisse der jüngsten Zeit untermauert. Eines davon bezieht sich auf das weiter oben eingeführte Problem PRIME, also auf die Frage, ob es sich bei einer gegebenen Zahl p um eine Primzahl handelt oder nicht. Obwohl Primzahlen seit tausenden von Jahren intensiv untersucht werden, wurde die Komplexität von PRIME erst im Jahr 2002 vollständig geklärt. In diesem Jahr gelang es den indischen Computerwissenschaftlern Manindra Agrawal, Neeraj Kayal und Nitin Saxena, einen polynomiellen Algorithmus für PRIME zu konstruieren [2]. Der *AKS-Algorithmus* erzeugte ein Echo weit über die Wissenschaftsgemeinde hinaus. Sogar die New York Times publizierte einen Artikel über die *„New Method Said to Solve Key Problem in Math"* [86].

Die Arbeit von Agrawal, Kayal und Saxena ist aus zweierlei Gründen von Bedeutung. Zum einen zeigt sie, dass wir mit unseren Vermutungen oft falsch liegen. Viele Wissenschaftler waren vor 2002 der Überzeugung, dass für das Problem PRIME kein Polynomialzeitalgorithmus existiert. Zum anderen demonstriert sie, wie groß unsere Wissenslücken im Fundament der Informatik noch immer sind. Die Ansicht, wir hätten es hier mit einer vollständig untersuchten, in sich abgeschlossenen Wissenschaft zu tun, trügt. Ganz im Gegenteil: Die theoretische Informatik lebt!

„Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. [...] This apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. [...]"

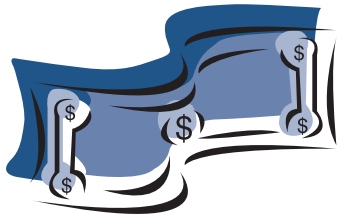


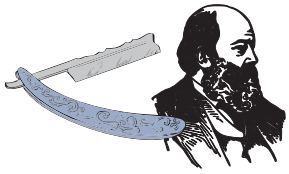
Abbildung 1.21: Für die endgültige Klärung des P-NP-Problems lobte das Clay Mathematics Institute im Jahr 2000 ein Preisgeld von einer Million US-Dollar aus.

1.4 Übungsaufgaben

Aufgabe 1.1

Webcode 1433

Anhand des *Barbier-Paradoxons* lässt sich die Russell'sche Antinomie mit Begriffen des Alltags veranschaulichen.



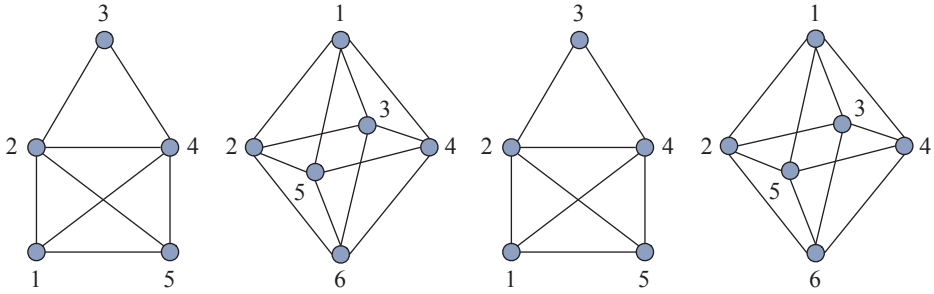
„Der Barbier von Sevilla rasiert genau diejenigen Männer von Sevilla, die sich nicht selbst rasieren."

Stellen Sie fest, ob sich der Barbier selbst rasiert.

Aufgabe 1.2

Webcode 1861

Gegeben seien die folgenden Graphen:



a) Versuchen Sie, in den beiden linken Graphen einen Euler-Kreis einzuzeichnen.

b) Versuchen Sie, in den beiden rechten Graphen einen Hamilton-Kreis einzuzeichnen.

Aufgabe 1.3

Webcode 1181

Im Gegensatz zum Königsberger Brückenproblem haben wir in diesem Kapitel keine konkrete Berechnungsvorschrift für die Lösung des Hamilton-Problems angegeben.

a) Skizzieren Sie einen Algorithmus, der für jeden Graphen korrekt entscheidet, ob ein gegebener Rundweg ein Hamilton-Kreis ist oder nicht.

b) Skizzieren Sie einen Algorithmus, der für jeden Graphen korrekt entscheidet, ob er einen Hamilton-Kreis besitzt oder nicht.

c) Wie hängt der Aufwand der formulierten Algorithmen mit der Größe des Graphen zusammen? Könnten Sie Ihren Algorithmus aus Teil b) beschleunigen, wenn Sie die Möglichkeit hätten, Rundwege mit bestimmten Eigenschaften zu erraten?

Aufgabe 1.4

✍✍✍

Webcode 1904

In diesem Kapitel haben Sie erfahren, wie der Logizismus die Mathematik zu Beginn des zwanzigsten Jahrhunderts verändert hat. Die zuvor physikalisch geprägte Mathematik wurde durch *formale Systeme* verdrängt, die sich aus *Axiomen* und *Regeln* zusammensetzen. Ein Theorem ist bewiesen, wenn es sich durch die Anwendung einer endlichen Folge von Regelanwendungen aus den Axiomen herleiten lässt. In einem solchen *Kalkül* wird das Führen eines mathematischen Beweises zu einem mechanischen Prozess.

Um das Gesagte mit Leben zu füllen, wollen wir ein konkretes formales System betrachten. Es stammt aus Douglas Hofstadters Meisterwerk *Gödel, Escher, Bach* und gibt einen erstklassigen Eindruck von der Grundidee des logischen Schließens [51]:

- Axiome
  1. **MI** ist ein Satz
- Schlussregeln
  1. Mit *x***I** ist auch *x***IU** ein Satz
  2. Mit **M***x* ist auch **M***xx* ein Satz
  3. In jedem Satz darf **III** durch **U** ersetzt werden
  4. In jedem Satz darf **UU** entfernt werden

In dem soeben definierten *MIU-System* lässt sich z. B. der Satz **MUIIU** wie folgt beweisen:

	MI	(Axiom)
⇒	**MII**	(Regel 2)
⇒	**MIIII**	(Regel 2)
⇒	**MUI**	(Regel 3)
⇒	**MUIU**	(Regel 1)
⇒	**MUIUUIU**	(Regel 2)
⇒	**MUIIU**	(Regel 4)

a) Lassen sich die Sätze **MIU** und **MIIUIIU** innerhalb des Kalküls ableiten?

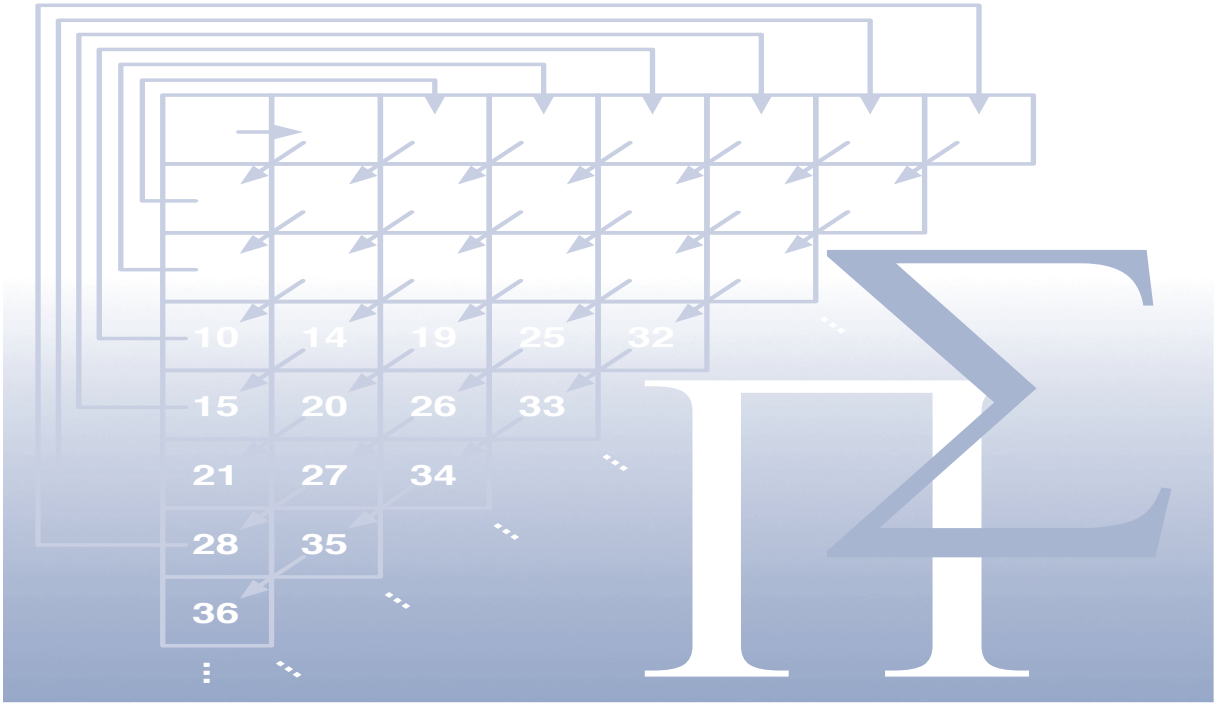
b) Ist die Reihenfolge der angewendeten Regeln immer eindeutig bestimmt?

c) Versuchen Sie zu ermitteln, ob sich der Satz **MU** aus den Axiomen ableiten lässt. Konstruieren Sie einen Beweis in Form einer konkreten Ableitungssequenz oder begründen Sie, warum ein solcher Beweis nicht existieren kann.

2 Mathematische Grundlagen

In diesem Kapitel werden Sie ...

- die Cantor'sche Definition der Menge ergründen,
- die grundlegenden Eigenschaften von Relationen und Funktionen kennen lernen,
- die natürlichen, rationalen und reellen Zahlen untersuchen,
- den systematischen Umgang mit der Unendlichkeit erlernen,
- induktive Definitionen und Beweise verstehen.



Georg Cantor
(1845 – 1918)

Abbildung 2.1: Der deutsche Mathematiker Georg Cantor wurde am 3. März 1845 in Sankt Petersburg geboren. Nach seiner Ausbildung in Zürich, Göttingen und Berlin folgte er einem Ruf an die Universität Halle, an der er über 40 Jahre lang lehrte und forschte. Cantor gehört zu den bedeutendsten Mathematikern des späten neunzehnten und frühen zwanzigsten Jahrhunderts. Mit seiner *Mannigfaltigkeitslehre* begründete er die Mengenlehre und legte mit dem Begriff der *Kardinalität* den Grundstein für den Umgang mit der Unendlichkeit. Der Begriff der *Abzählbarkeit* geht genauso auf Cantor zurück wie die *Diagonalisierungsmethode*, mit deren Hilfe sich viele Erkenntnisse der theoretischen Informatik auf anschauliche Weise erklären lassen. Im Alter von 39 Jahren erkrankt Cantor an manischer Depression – ein Leiden, das ihn bis zu seinem Lebensende begleiten sollte. Kurz nach seinem siebzigsten Geburtstag wird er nach einem erneuten Krankheitsausbruch in die Universitätsklinik Halle eingewiesen. Dort stirbt Georg Cantor am 6. Januar 1918 im Alter von 72 Jahren.

2.1 Grundlagen der Mengenlehre

2.1.1 Der Mengenbegriff

Wir beginnen unseren Streifzug durch die Grundlagen der Mathematik mit einem Abstecher in das Gebiet der Mengenlehre. Für jeden von uns besitzt der Begriff der *Menge* eine intuitive Interpretation, die nicht zuletzt durch unser Alltagsleben geprägt ist. So fassen wir die 22 Akteure auf dem Fußballplatz wie selbstverständlich zu zwei Elfergruppen zusammen und wissen auch in anderen Lebenslagen Äpfel von Birnen zu unterscheiden. Die Zusammenfassung einer beliebigen Anzahl von Dingen bezeichnen wir als *Menge* und jedes darin enthaltene Objekt als *Element*.

Definition 2.1 (Mengendefinition nach Cantor)

„Unter einer *Menge* verstehen wir jede Zusammenfassung M von bestimmten wohl unterschiedenen Objekten unserer Anschauung oder unseres Denkens (welche die *Elemente* von M genannt werden) zu einem Ganzen." (Georg Cantor)

Dies ist der Originalwortlaut, mit dem der deutsche Mathematiker Georg Cantor (Abbildung 2.1) im Jahr 1885 den Mengenbegriff formulierte [13]. Die hierauf begründete mathematische Theorie wird als *Cantor'sche Mengenlehre* bezeichnet. Ebenfalls üblich sind die Begriffe der *anschaulichen*, *intuitiven* oder *naiven Mengenlehre*, um sie von den später entwickelten, streng axiomatisch definierten Mengenbegriffen abzugrenzen.

Wir schreiben $a \in M$, um auszudrücken, dass a ein Element von M ist. Entsprechend drückt die Notation $a \notin M$ aus, dass a nicht zu M gehört. Die abkürzende Schreibweise $a, b \in M$ bzw. $a, b \notin M$ besagt, dass sowohl a als auch b Elemente von M sind bzw. beide nicht zu M gehören. Zwei Mengen M_1 und M_2 gelten als gleich ($M_1 = M_2$), wenn sie exakt dieselben Elemente enthalten. Im Umkehrschluss existiert für zwei ungleiche Mengen M_1 und M_2 stets ein Element in M_1 oder M_2, das nicht in der anderen Menge enthalten ist. Wir schreiben in diesem Fall $M_1 \neq M_2$. Offensichtlich gilt für jedes Objekt a und jede Menge M entweder $a \in M$ oder $a \notin M$.

Im Gegensatz zur umgangssprachlichen Bedeutung des Begriffs der Menge spielt es im mathematischen Sinn keine Rolle, ob darin wirklich

viele Objekte zusammengefasst sind. Wir reden selbst dann von einer Menge, wenn diese überhaupt keine Elemente enthält. Für diese *leere Menge* ist das spezielle Symbol $\emptyset$ reserviert.

Mengen können ein einzelnes Objekt niemals mehrfach beinhalten und genauso wenig besitzen ihre Elemente einen festen Platz; Mengen sind inhärent ungeordnet. Im Übungsteil dieses Kapitels werden Sie sehen, dass der Mengenbegriff trotzdem stark genug ist, um geordnete Zusammenfassungen zu modellieren, die zudem beliebig viele Duplikate enthalten dürfen.

In der Praxis haben sich zwei unterschiedliche Schreibweisen etabliert, um die Elemente einer Menge zu definieren:

- Aufzählende Beschreibung

  Die Elemente einer Menge werden explizit aufgelistet. Selbst unendliche Mengen lassen sich aufzählend (enumerativ) beschreiben, wenn die Elemente einer unmittelbar einsichtigen Regelmäßigkeit unterliegen. Die nachstehenden Beispiele bringen Klarheit:

  $$\begin{aligned} \mathbb{N} &:= \{0,1,2,3,\ldots\} \\ \mathbb{N}^+ &:= \{1,2,3,4,\ldots\} \\ \mathbb{Z} &:= \{\ldots,-2,-1,0,1,2,\ldots\} \\ M_1 &:= \{0,2,4,6,8,10,\ldots\} \\ M_2 &:= \{0,1,4,9,16,25,\ldots\} \end{aligned}$$

  $\mathbb{N}$ heißt die Menge der *natürlichen Zahlen* oder die Menge der *nichtnegativen ganzen Zahlen*. $\mathbb{N}^+$ beginnt mit der 1 und wird die Menge der *positiven ganzen Zahlen* genannt. $\mathbb{Z}$ ist die Menge der *ganzen Zahlen*. M_1 enthält alle geraden natürlichen Zahlen und die Menge M_2 die Quadrate der ganzen Zahlen.

- Deskriptive Beschreibung

  Die Mengenzugehörigkeit eines Elements wird durch eine charakteristische Eigenschaft beschrieben. Genau jene Elemente sind in der Menge enthalten, auf die die Eigenschaft zutrifft.

  $$\begin{aligned} M_3 &:= \{n \in \mathbb{N} \mid n \bmod 2 = 0\} \\ M_4 &:= \{n^2 \mid n \in \mathbb{N}\} \end{aligned}$$

  Demnach enthält die Menge M_3 alle Elemente $n \in \mathbb{N}$, die sich ohne Rest durch 2 dividieren lassen, und die Menge M_4 die Werte n^2 für alle natürlichen Zahlen $n \in \mathbb{N}$. Die Mengen M_3 und M_4 sind damit nichts anderes als eine deskriptive Beschreibung der im vorherigen Beispiel eingeführten Mengen M_1 und M_2.

Auf den ersten Blick scheint der Mengenbegriff intuitiv erfassbar zu sein, auf den zweiten entpuppt er sich als komplexes Gebilde. Bereits im Einführungskapitel konnten wir den Cantor'schen Mengenbegriff mithilfe der *Russell'schen Antinomie* als widersprüchlich entlarven.
Damit die Mathematik nicht auf wackligen Füßen steht, wurde mit der *axiomatischen Mengenlehre* eine formale Theorie geschaffen, die Inkonsistenzen der Russell'schen Art beseitigen soll. Einen wichtigen Grundstein legte der deutsche Mathematiker Ernst Zermelo, als er im Jahr 1907 ein entsprechendes Axiomensystem formulierte. Die *Zermelo-Mengenlehre* bestand aus insgesamt 7 Axiomen, die noch umgangssprachlich formuliert waren [111]. Das System wurde 1921 von Abraham Fraenkel um das Ersetzungsaxiom und 1930 von Zermelo um das Fundierungsaxiom ergänzt [32, 112]. Die 9 Axiome bilden zusammen die *Zermelo-Fraenkel-Mengenlehre*, kurz ZF, wie sie heute in weiten Teilen der Mathematik Verwendung findet (Abbildung 2.2).
Wird ZF zusätzlich um das *Auswahlaxiom* (*axiom of choice*) erweitert, so entsteht die ZFC-Mengenlehre (*Zermelo-Fraenkel with Choice*). Das zusätzliche zehnte Axiom besagt das Folgende: Ist M eine Menge von nichtleeren Mengen, dann gibt es eine Funktion f, die aus jeder Menge $M' \in M$ genau ein Element auswählt. Das Auswahlaxiom ist unabhängig von allen anderen. 1937 zeigte Kurt Gödel, dass es sich widerspruchsfrei zu den ZF-Axiomen hinzufügen lässt [37]. 1963 kam Paul Cohen zu dem erstaunlichen Ergebnis, dass die Negation des Auswahlaxioms die Widerspruchsfreiheit von ZF ebenfalls nicht zerstört [23].
Mit dem ehemaligen Mengenbegriff von Cantor hat die axiomatische Mengenlehre nur wenig gemein. Sie gehört heute zu den schwierigsten Teilgebieten der Mathematik und nur wenigen ist es vergönnt, sie vollständig zu durchdringen.

- Axiom der Bestimmtheit (Zermelo, 1908)

$$\forall x\, \forall y\, (x = y \leftrightarrow \forall z\, (z \in x \leftrightarrow z \in y))$$

„Ist jedes Element einer Menge M gleichzeitig Element von N und umgekehrt, ist also gleichzeitig $M \subset N$ und $N \subset M$, so ist immer $M = N$. Oder kürzer: jede Menge ist durch ihre Elemente bestimmt.“

- Axiom der leeren Menge (Zermelo, 1908)

$$\exists x\, \forall y\, y \notin x$$

„Es gibt eine (uneigentliche) Menge, die ‚Nullmenge‘ $\emptyset$, welche gar keine Elemente enthält.“

- Axiom der Paarung (Zermelo, 1908)

$$\forall x\, \forall y\, \exists z\, \forall u\, (u \in z \leftrightarrow u = x \vee u = y)$$

„Sind a, b irgend zwei Dinge des Bereichs, so existiert immer eine Menge $\{a, b\}$, welche sowohl a als [auch] b, aber kein von beiden verschiedenes Ding x als Element enthält.“

- Axiom der Vereinigung (Zermelo, 1908)

$$\forall x\, \exists y\, \forall z\, (z \in y \leftrightarrow \exists (w \in x)\, z \in w)$$

„Jeder Menge T entspricht eine Menge $\mathfrak{S}T$ (die ‚Vereinigungsmenge‘ von T), welche alle Elemente der Elemente von T und nur solche als Elemente enthält.“

- Axiom der Aussonderung (Zermelo, 1908)

$$\forall x\, \exists y\, \forall z\, (z \in y \leftrightarrow z \in x \wedge \varphi(z))$$

„Durch jede Satzfunktion $\mathfrak{f}(x)$ wird aus jeder Menge m eine Untermenge $m_{\mathfrak{f}}$ ausgesondert, welche alle Elemente x umfasst, für die $\mathfrak{f}(x)$ wahr ist. Oder: Jedem Teil einer Menge entspricht selbst eine Menge, welche alle Elemente dieses Teils enthält.“

- Axiom des Unendlichen (Zermelo, 1908)

$$\exists x\, (\emptyset \in x \wedge \forall (y \in x)\, \{y\} \in x)$$

„Der Bereich enthält mindestens eine Menge Z, welche die Nullmenge als Element enthält und so beschaffen ist, dass jedem ihrer Elemente a ein weiteres Element der Form $\{a\}$ entspricht.“

- Axiom der Potenzmenge (Zermelo, 1908)

$$\forall x\, \exists y\, \forall z\, (z \in y \leftrightarrow z \subseteq x)$$

„Jeder Menge m entspricht eine Menge $\mathfrak{U}m$, welche alle Untermengen von m als Elemente enthält, einschließlich der Nullmenge und m selbst.“

- Axiom der Ersetzung (Fraenkel, 1922)

$$(\forall (a \in x)\, \exists_1 b\, \varphi(a, b)) \rightarrow (\exists y\, \forall b\, (b \in y \leftrightarrow \exists (a \in x)\, \varphi(a, b)))$$

„Ist M eine Menge und wird jedes Element von M durch ‚ein Ding des Bereichs $\mathfrak{B}$‘ ersetzt, so geht M wiederum in eine Menge über.“

- Axiom der Fundierung (Zermelo, 1930)

$$\forall x\, (x \neq \emptyset \rightarrow \exists (y \in x)\, x \cap y = \emptyset)$$

„Jede (rückschreitende) Kette von Elementen, in welcher jedes Glied Element des vorangehenden ist, bricht mit endlichem Index ab bei einem Urelement. Oder, was gleichbedeutend ist: Jeder Teilbereich T enthält wenigstens ein Element t_0, das kein Element t in T hat.“

- Optional: Axiom der Auswahl (Zermelo, 1904)

$$(\forall (u, v \in x)\, (u \neq v \rightarrow u \cap v = \emptyset) \wedge \forall (u \in x)\, u \neq \emptyset) \rightarrow \exists y\, \forall (z \in x)\, \exists_1 (w \in z)\, w \in y$$

„Ist T eine Menge, deren sämtliche Elemente von $\emptyset$ verschiedene Mengen und untereinander elementfremd sind, so enthält ihre Vereinigung $\cup T$ mindestens eine Untermenge S_1, welche mit jedem Element von T ein und nur ein Element gemein hat.“

Abbildung 2.2: Axiome der Zermelo-Fraenkel-Mengenlehre

Von den ganzen Zahlen $\mathbb{Z}$ wissen wir, dass sie sich mit den Vergleichsoperatoren $\leq$ und $\geq$ in eine Ordnung bringen lassen. Mengen lassen sich mithilfe der *Teil- oder Untermengenbeziehung* $\subseteq$ und der *Obermengenbeziehung* $\supseteq$ auf ähnliche Weise ordnen:

$$M_1 \subseteq M_2 \;:\Leftrightarrow\; \text{Aus } a \in M_1 \text{ folgt } a \in M_2$$
$$M_1 \supseteq M_2 \;:\Leftrightarrow\; M_2 \subseteq M_1$$

Beachten Sie, dass die Teilmengenbeziehung nach dieser Definition immer auch dann gilt, wenn die linke Menge überhaupt keine Elemente enthält. Mit anderen Worten: Die leere Menge $\emptyset$ ist eine Teilmenge jeder anderen Menge. Des Weiteren ist jede Menge auch eine Teilmenge von sich selbst. Folgerichtig gelten die Beziehungen $\emptyset \subseteq M$, $M \supseteq \emptyset$, $M \subseteq M$ und $M \supseteq M$.

Mithilfe der eingeführten Operatoren können wir die Mengengleichheit wie folgt charakterisieren:

$$M_1 = M_2 \;\Leftrightarrow\; M_1 \subseteq M_2 \text{ und } M_2 \subseteq M_1$$

Zusätzlich vereinbaren wir die Operatoren $\subset$ (*echte Teilmenge*) und $\supset$ (*echte Obermenge*):

$$M_1 \subset M_2 \;:\Leftrightarrow\; M_1 \subseteq M_2 \text{ und } M_1 \neq M_2$$
$$M_1 \supset M_2 \;:\Leftrightarrow\; M_1 \supseteq M_2 \text{ und } M_1 \neq M_2$$

Offensichtlich gilt für die weiter oben eingeführten Mengen die Beziehung $M_1 \subset \mathbb{N} \subset \mathbb{Z}$. Dagegen gilt weder $M_1 \subset M_2$ noch $M_2 \subset M_1$.

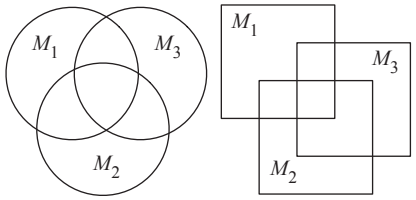
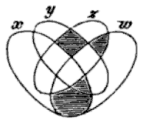


Abbildung 2.3: *Venn-Diagramme* sind ein anschauliches Hilfsmittel, um Beziehungen zwischen Mengen zu visualisieren. Eine Menge wird durch eine Fläche beschrieben, die durch einen geschlossen Linienzug begrenzt wird. Jeder diskrete Punkt innerhalb der Fläche entspricht einem Element der Menge.



Viergliedriges Venn-Diagramm aus der Originalarbeit von 1881

John Venn (1834 – 1923)

Abbildung 2.4: Das Venn-Diagramm wurde im Jahr 1881 durch den britischen Logiker und Philosophen John Venn eingeführt [101,102]. Es bildet heute die am häufigsten eingesetzte Darstellungsform für die bildliche Repräsentation einer Menge.

2.1.2 Mengenoperationen

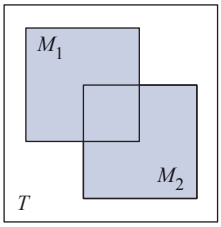
Bestehende Mengen lassen sich durch die Anwendung von *Mengenoperationen* zu neuen Mengen verknüpfen. In den nachstehenden Betrachtungen seien M_1 und M_2 Teilmengen einer nichtleeren *Universal-* bzw. *Trägermenge* T. Die *Vereinigungsmenge* $M_1 \cup M_2$ und die *Schnittmenge* $M_1 \cap M_2$ sind wie folgt definiert:

$$M_1 \cup M_2 := \{a \mid a \in M_1 \text{ oder } a \in M_2\}$$
$$M_1 \cap M_2 := \{a \mid a \in M_1 \text{ und } a \in M_2\}$$

Zwei Mengen M_1 und M_2 heißen *disjunkt*, falls $M_1 \cap M_2 = \emptyset$ gilt.

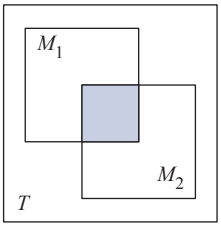
Die Definition lässt sich auf die Vereinigung bzw. den Schnitt beliebig vieler Mengen verallgemeinern. Für die endlich vielen Mengen

- Vereinigung



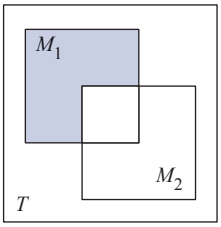
$M_1 \cup M_2$

- Schnitt



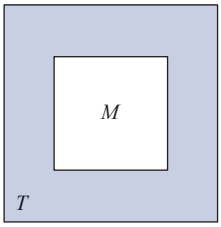
$M_1 \cap M_2$

- Differenz



$M_1 \setminus M_2$

- Komplement



$\overline{M}$

Abbildung 2.5: Elementare Mengenoperationen

$M_1, \ldots, M_n$ bzw. die unendlich vielen Mengen $M_1, M_2, \ldots$ vereinbaren wir die folgende Schreibweise:

$$\bigcup_{i=1}^{n} M_i := M_1 \cup \ldots \cup M_n \quad \text{bzw.} \quad \bigcup_{i=1}^{\infty} M_i := M_1 \cup M_2 \cup \ldots$$

$$\bigcap_{i=1}^{n} M_i := M_1 \cap \ldots \cap M_n \quad \text{bzw.} \quad \bigcap_{i=1}^{\infty} M_i := M_1 \cap M_2 \cap \ldots$$

Zusätzlich definieren wir die *Differenzmenge* $M_1 \setminus M_2$ sowie die *Komplementärmenge* $\overline{M}$ wie folgt:

$$M_1 \setminus M_2 := \{a \mid a \in M_1 \text{ und } a \notin M_2\}$$
$$\overline{M} := T \setminus M$$

Viele Mengenbeziehungen lassen sich intuitiv mithilfe von *Venn-Diagrammen* veranschaulichen. Die Elemente einer Menge werden durch diskrete Punkte und die Mengen selbst als geschlossene Gebiete in der Ebene repräsentiert (vgl. Abbildungen 2.3 bis 2.5).

Die Vereinigungs-, Schnitt- und Komplementoperatoren begründen zusammen die *Mengenalgebra*. In der entstehenden algebraischen Struktur gilt eine Reihe von Gesetzen, die sich direkt aus der Definition der Operatoren ergeben. Insbesondere lassen sich die folgenden vier Verknüpfungsregeln ableiten:

- Kommutativgesetze

$$M_1 \cup M_2 = M_2 \cup M_1$$
$$M_1 \cap M_2 = M_2 \cap M_1$$

- Distributivgesetze

$$M_1 \cup (M_2 \cap M_3) = (M_1 \cup M_2) \cap (M_1 \cup M_3)$$
$$M_1 \cap (M_2 \cup M_3) = (M_1 \cap M_2) \cup (M_1 \cap M_3)$$

- Neutrale Elemente

$$M \cup \emptyset = M$$
$$M \cap T = M$$

- Inverse Elemente

$$M \cup \overline{M} = T$$
$$M \cap \overline{M} = \emptyset$$