

The cover features a central wireframe head, resembling a mesh or a complex network of lines, set against a dark blue background. The background is dominated by a large, intricate fractal pattern that spirals inward, creating a sense of depth and complexity. The overall aesthetic is technical and abstract, reflecting the book's focus on computation and its limits.

OXFORD

COMPUTATION AND ITS LIMITS

Paul Cockshott, Lewis Mackenzie,
Greg Michaelson

Computation and its Limits

This page intentionally left blank

Computation and its Limits

Paul Cockshott
University of Glasgow

Lewis M. Mackenzie
University of Glasgow

Greg Michaelson
Heriot-Watt University

OXFORD
UNIVERSITY PRESS

OXFORD

UNIVERSITY PRESS

Great Clarendon Street, Oxford OX2 6DP

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide in

Oxford New York

Auckland Cape Town Dar es Salaam Hong Kong Karachi
Kuala Lumpur Madrid Melbourne Mexico City Nairobi
New Delhi Shanghai Taipei Toronto

With offices in

Argentina Austria Brazil Chile Czech Republic France Greece
Guatemala Hungary Italy Japan South Korea Poland Portugal
Singapore Switzerland Thailand Turkey Ukraine Vietnam

Oxford is a registered trade mark of Oxford University Press
in the UK and in certain other countries

Published in the United States
by Oxford University Press Inc., New York

© Paul Cockshott, Lewis M. Mackenzie and Greg Michaelson 2012

The moral rights of the authors have been asserted
Database right Oxford University Press (maker)

First published 2012

All rights reserved. No part of this publication may be reproduced,
stored in a retrieval system, or transmitted, in any form or by any means,
without the prior permission in writing of Oxford University Press,
or as expressly permitted by law, or under terms agreed with the appropriate
reprographics rights organization. Enquiries concerning reproduction
outside the scope of the above should be sent to the Rights Department,
Oxford University Press, at the address above

You must not circulate this book in any other binding or cover
and you must impose this same condition on any acquirer

British Library Cataloguing in Publication Data
Data available

Library of Congress Cataloging in Publication Data
Library of Congress Control Number: 2011945375

Typeset by Cenveo, Bangalore, India
Printed and bound by
CPI Group (UK) Ltd, Croydon, CRO 4YY

ISBN 978-0-19-964032-4

1 3 5 7 9 10 8 6 4 2

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Overview | 1 |
| 1.2 | Summary | 4 |
| 1.3 | Acknowledgements | 5 |
| 2 | What is computation? | 6 |
| 2.1 | The apparent mystery of maths | 6 |
| 2.2 | Counting sheep | 11 |
| 2.3 | Counting materialised in our own bodily movements | 16 |
| 2.4 | From ‘ <i>aides-memoire</i> ’ to the first digital calculating devices | 23 |
| 3 | Mechanical computers and their limits | 28 |
| 3.1 | Antikythera | 28 |
| 3.2 | Late mechanical computers | 35 |
| 3.3 | Analogue mechanical multiply/accumulate | 39 |
| 3.4 | Mechanizing the abacus | 42 |
| 4 | Logical limits to computing | 47 |
| 4.1 | Introduction | 47 |
| 4.2 | Propositional logic | 47 |
| 4.3 | Set theory | 51 |
| 4.4 | Predicate logic | 52 |
| 4.5 | Recursion | 55 |
| 4.6 | Peano arithmetic | 55 |
| 4.7 | Paradoxes | 58 |
| 4.8 | Arithmetizing mathematics and incompleteness | 61 |
| 4.9 | Infinities | 64 |
| 4.10 | Real numbers and Cantor diagonalization | 66 |
| 4.11 | Turing machines | 67 |
| 4.12 | Universal TM and undecidability | 71 |
| 4.13 | Computational procedures | 74 |
| 4.14 | The Church–Turing thesis | 78 |
| 4.15 | Machines, programs, and expressions | 79 |
| 5 | Heat, information, and geometry | 83 |
| 5.1 | The triumph of digital computation | 83 |
| 5.2 | Analogue computing with real numbers | 84 |
| 5.3 | What memories are made of | 86 |
| 5.4 | Power consumption as a limit | 91 |

| | | |
|----------|--|------------|
| 5.5 | Entropy | 95 |
| 5.6 | Shannon's information theory | 111 |
| 5.7 | Landauer's limit | 114 |
| 5.8 | Non-entropic computation | 117 |
| 5.9 | Interconnection | 122 |
| 6 | Quantum computers | 128 |
| 6.1 | Foundations of quantum theory | 128 |
| 6.2 | The quantum rules | 136 |
| 6.3 | Qubits | 142 |
| 6.4 | Entanglement and quantum registers | 146 |
| 6.5 | Quantum computers | 153 |
| 6.6 | Quantum algorithms | 155 |
| 6.7 | Building a quantum computer | 157 |
| 6.8 | Physical limits to real number representations | 167 |
| 6.9 | Error rates in classical and quantum gates | 171 |
| 7 | Beyond the logical limits of computing? | 173 |
| 7.1 | Introduction | 173 |
| 7.2 | Oracles, complexity, and tractability | 174 |
| 7.3 | Beyond the Turing Machine? | 176 |
| 7.4 | Numberology | 177 |
| 7.5 | What is real about the reals? | 180 |
| 7.6 | Real measurement | 181 |
| 7.7 | Back to Turing | 184 |
| 7.8 | Reservations about Cantor | 185 |
| 8 | Hypercomputing proposals | 187 |
| 8.1 | Infinite Turing Machines | 187 |
| 8.2 | Infinitely precise analogue computers | 190 |
| 8.3 | Wegner and Eberbach's super-Turing computers | 201 |
| 8.4 | Interaction Machines | 202 |
| 8.5 | π -Calculus | 206 |
| 8.6 | $\$$ -Calculus | 211 |
| 8.7 | Conclusions | 214 |
| | Bibliography | 216 |
| | Index | 226 |

Introduction



| | | |
|-----|------------------|---|
| 1.1 | Overview | 1 |
| 1.2 | Summary | 4 |
| 1.3 | Acknowledgements | 5 |

1.1 Overview

Although we are entirely unaware of it, *computation* is central to all aspects of our existence; that is, in the application of *rules* to *information* to produce new information, usually to make a decision about what to do next.¹ Every day, we solve, or try to solve, a myriad of problems, from the utterly trivial to the bafflingly complex. To do this, we deploy some processes of reasoning, often unarticulated, and often aided by large doses of what we might call ‘common sense’, ‘gut feeling’, or ‘intuition’. And often, our reasoning is not as definitive as we might like, leading us to conclusions that are wrong or tentative, or contrary to what we might prefer.

¹Here, we are using ‘information’ in the general sense of facts or knowledge about some circumstance, rather than the precisely defined technical conception of information discussed in the rest of this book.

It is striking that our reasoning is most accurate where the information and rules are most precise, in particular where we use *numbers* and rules deriving from *mathematics*, such as those for arithmetic, geometry, and logic. Again, we carry out most of this sort of computation quite unconsciously, except when we get it wrong. And we do so using *standard* representations and rules that we learnt as children, and that we expect other people to share. One big strength of this standardization of numerical reasoning is that it can easily be made explicit, and so repeated and checked, by ourselves or other people, until it is right. Another big strength is that we can build *machines*, such as calculators and computers, to carry out computations for us, and we can agree that these machines really do embody the rules and information that we hold in common.

Now, we actually perform very sophisticated computations all the time. In the course of a day, we might, say:

- check whether we have the right change for an automatic ticket machine
- decide how much food to buy in order to cook for guests as well as the household
- make sure that the shop bill has accounted for everything that we bought
- decide what order to cook the food in, to make sure that the whole of each course is ready at the appropriate time

- work out how many days we can afford to spend on our spring vacation, if we've already committed some to a summer holiday and need to reserve some for a family celebration
- decide whether it's cheaper to take the train and spend a night in a hotel on the way or fly directly to our destination

These sorts of computations seem trivial; certainly, they are small and simple enough for us to perform in our heads. Nonetheless, they actually require considerable competence in mental arithmetic and sequencing of activities, as becomes clear if we try to explain how to do them, step by step, to someone else.

It is important to note that we do not generally need fine precision for these sorts of daily computations. In the above examples, we want *enough* ticket machine change, food or time for cooking, holiday days left, or travel cost savings, where 'enough' allows a reasonable margin of error. So, we accept that we may make mistakes but are happy with 'ballpark' figures.

For more complex—or more costly—computations, however, such as planning a new kitchen or an extensive holiday, we require greater assurance and precision. Here, we readily turn first to pencil and paper, next to a calculator, and, increasingly, to standard software on a personal computer, such as a spreadsheet. We do so, perhaps, because we believe that these methods offer a hierarchy of increasing checkability and reliability as mechanization increases and human computation decreases. Indeed, these tools enable us to focus on making *models* of problems—an area in which human beings are supremely accomplished—rather than performing complex computations, where we are prone to inaccuracies and mistakes.

Now, sometimes things still go wrong when we use apparently reliable tools. Almost always, the mistakes are our own fault—for example, if we enter the wrong data or faulty equations into a spreadsheet.

Nonetheless, we are plainly aware of the limitations to our tools. Thus, it's surprisingly easy to try to carry out a calculation to more decimal places than a pocket calculator can support. More to the point, we constantly notice that our personal computers are going 'too slowly', or running out of memory, as we demand more and more from them. However, these limitations do not seem insurmountable, because we have got used to the continuous cheapening of computers and memory, accompanied by increases in their speed and capacity, either through the use of new technologies or by making better use of existing ones.

For example, just from looking at advertisements, the speed of a single computer processor seems to have stayed at around 3 GHz since the turn of the century. However, we are now offered personal computers made up of *multiple cores*—first two, next four and eight—with no apparent limit to the number. Thus, the limited speed of one processor can be compensated by the use of lots of them.

Furthermore, digital cameras and personal music players now use solid state memory, whereas previously they were based on magnetic tape,

magnetic disks, or CDs or DVDs. And laptops and ‘web books’ no longer necessarily have hard disks, let alone CD or DVD writers.

We have also got used to a very different style of accessing and storing information, on remote machines run by other people, rather than on some tangibly physical device on a desk or in a pocket. Just as CDs killed cassette tapes and LPs for music, and DVDs displaced video for films, now direct downloading to media players of intangible codes is replacing personal ownership of physical copies.²

²And books may yet succumb to eReaders.

Contrariwise, we trust more and more material that might once have been considered our personal responsibility to external repositories of unknown provenance, collectively known as ‘clouds’. Thus, we not only implicitly accept remote maintenance of our email folders by our service providers, but also explicitly upload what we still regard as ‘private’ photographs and documents to unknown physical locations.

Finally, we routinely take advantage of complex computations performed remotely on our behalf. For example, we increasingly optimize flights for multi-location holidays, check weather forecasts, or find routes for road journeys, all using online services rather than consulting timetables, barometers, or maps.

Thus, even if there are restrictions on the machines that we can own ourselves, there seem to be no bounds to the capacities of the shadowy world of services and servers at the other end of the Internet. And even if we constantly run up against the limits of current technology, or what we can afford, then all our experience suggests that the former will eventually be overcome, if not the latter.

Alas, as we explore in the rest of this book, there are deep reasons why such optimism in unlimited technological progress is absolutely unfounded, even if it seems relatively appropriate. First, when we try to give a precise characterization to the notion of *abstract computation*, we discover that certainty in our reasoning is undermined by troubling paradoxes at the core of mathematics. And secondly, when we look at the physical characteristics of our universe, we find that there are fundamental limits to the properties of the *concrete computers* that we might construct.

Now, we are all familiar with the idea that some things are just not physically feasible—at least, not within physics as we currently understand it. For example, we generally accept that time travel, perpetual motion, and faster-than-light speeds are only possible in the realm of science fiction. Nonetheless, every year people come up with new attempts to transcend these physical barriers. And every year, scientists and engineers explore and ultimately refute such attempts.

Similarly, there have been many attempts to develop abstract *hypercomputational* systems or concrete *hypercomputers*, which seek to transcend the mathematical and physical limits to which we have just alluded. Thus, as well as exploring limits to computation and to computers, we will also discuss attempts to overcome them, and consider why they are ultimately ill-founded.

1.2 Summary

2. **What is computation?** In this chapter, we address the abiding mystery of why mathematics has been so effective in science. We attempt to answer it by shifting attention from maths as something to do with ideal numbers on to the practical history of computation, starting with the foundations of our ability to count, and proceeding through the use of early aides to calculation.
3. **Mechanical computers and their limits.** Here, we focus on the idea of the computer as a distinct physical system that is set up to emulate another part of reality. This is easier to see with some of the earlier mechanical computers, so we pay attention
4. **Logical limits to computing.** We then go on to discuss how the modern idea of the universal computer arose from the history of mathematics in the late nineteenth and early twentieth centuries, and the attempt to provide a satisfactory basis to number theory. This leads up to a description of the Turing Machines and the halting problem.
5. **Heat, information, and geometry.** Computers are physical devices and as such are subject to constraints imposed by the laws of thermodynamics and geometry. In this chapter, we explain the concept of entropy and how it relates to information. We examine the extent to which entropy generation limits what computers can do.
6. **Quantum computers.** One of the most exciting developments in theoretical computer science in the last 30 years has been the proposal to apply quantum mechanisms to computation. We look at what quantum theory is, how it allows computation in terms of qubits, and how these might be harnessed to perform novel algorithms. We also look at the way in which quantum physics limits what can, in principle, be attained by analogue computing devices.
7. **Beyond the logical limits of computing?** Having covered classical computing theory and looked at the constraints imposed by physics, we then re-examine issues in classical computation in the light of these new limits and—if we take them seriously—ask what this means for our understanding of real-numbered calculation.
8. **Hypercomputing proposals.** In our final chapter, we examine a number of blue-skies proposals that have recently been suggested for going beyond the Turing Limit in computation. These involve esoteric relativistic physics and new paradigms for computation using more conventional physics.

1.3 Acknowledgements

We would like to thank the following people for discussions about and help with various ideas in this book: Gregory Chaitin, Naom Chomsky, Allin Cottrell, Eugene Eberbach, and Victor Yakovenko.

We alone are responsible for any mistakes in this book. We may be contacted at:

- Paul Cockshott, University of Glasgow,
`William.Cockshott@glasgow.ac.uk`
- Lewis M. Mackenzie, University of Glasgow,
`Lewis.Mackenzie@glasgow.ac.uk`
- Gregory Michaelson, Heriot-Watt University,
`G.Michaelson@hw.ac.uk`

2

What is computation?

| | | |
|-----|--|----|
| 2.1 | The apparent mystery of maths | 6 |
| 2.2 | Counting sheep | 11 |
| 2.3 | Counting materialized in our own bodily movements | 16 |
| 2.4 | From <i>'aides-memoire'</i> to the first digital calculating devices | 23 |

2.1 The apparent mystery of maths

In an influential paper, Wigner (1960) asked why it was that maths was so ‘unreasonably effective’ in describing physical reality. The question he asked is similar to those that we are dealing with in this book. Our topic is, on the surface, slightly narrower. We are dealing with computing rather than maths in general. But if we look at what Wigner has to say, we find that he too is concerned primarily with the practical application of maths, and that practical application to physics always involves doing real computations. Therefore, Wigner’s concerns make a good starting point.

He starts with a perhaps apocryphal tale:

There is a story about two friends, who were classmates in high school, talking about their jobs. One of them became a statistician and was working on population trends. He showed a reprint to his former classmate. The reprint started, as usual, with the Gaussian distribution and the statistician explained to his former classmate the meaning of the symbols for the actual population, for the average population, and so on. His classmate was a bit incredulous and was not quite sure whether the statistician was pulling his leg. ‘How can you know that?’ was his query. ‘And what is this symbol here?’ ‘Oh,’ said the statistician, ‘this is pi.’ ‘What is that?’ ‘The ratio of the circumference of the circle to its diameter.’ ‘Well, now you are pushing your joke too far,’ said the classmate, ‘surely the population has nothing to do with the circumference of the circle.’

Posed in this way, the fact that mathematics proves so useful to science and to understanding the world in general does seem quite remarkable. But look at this example. Is it really mathematics, or is it computation that is proving useful?

The Gaussian distribution, π , and so on are being used in order to perform particular calculations, by means of which the statistician intends to make predictions about the future population. On most occasions when we come across examples where maths proves useful, it is not abstract maths that is directly useful, but applied mathematics. Applied maths is used to perform calculations, and these either guide us in our practical engineering or allow us to make predictions in experimental science. So we can shift Wigner’s question and ask instead about the remarkable usefulness of applied mathematics.

Prior to the universalization of automatic computers, the contemporary distinction between computation and mathematics did not exist. The one was seen as just a necessary component part of the other. Both were intellectual activities, and both were performed by mathematicians, albeit with different specialisms. Now, when computing is done by machines, we tend to think of them as distinct things: maths as an intellectual discipline, computation as a rote automatic process. Of course, everyone concedes that the actual programming of computers is a tricky intellectual task that may involve mathematical skill. But it is on computing that science and engineering now depend for practical predictions—whether the personnel are working at Boeing on the design and control of an airliner, or in the Met Office on long-term climate forecasts. So we can now focus in a bit on Wigner’s question and ask why it is that computers can be so unreasonably effective in the practice of science.

So long as the question was posed in its original form, it seemed an insuperable enigma. True enough, Wigner did give an answer. He attributed the effectiveness of maths to the spatial and temporal invariance of the laws of physics. That no doubt plays its part, but the answer remains one that would appeal mainly to physicists, whose vision of the practical application of mathematics is very much limited to their own field. It does really answer the query about π in his initial anecdote.

We can allow that the maths required to do physics would be far more complicated were the laws of physics not spatially invariant, without gaining much insight into why maths should work in other domains. Why does maths work for predictions about populations, or in the analysis of genetic inheritance?

Following up on Wigner, Hamming asked:

Furthermore, the simplicity of mathematics has long been held to be the key to applications in physics. Einstein is the most famous exponent of this belief. But even in mathematics itself the simplicity is remarkable, at least to me; the simplest algebraic equations, linear and quadratic, correspond to the simplest geometric entities, straight lines, circles, and conics. This makes analytic geometry possible in a practical way. How can it be that simple mathematics, being after all a product of the human mind, can be so remarkably useful in so many widely different situations?

(Hamming, 1980).

We can visualize what Hamming is talking about in Fig. 2.1. There, we have Isaac Newton using the ideas of geometry to elaborate his *Mathematical Principles of Natural Philosophy*. These are pure thoughts, but, remarkably, the mathematics mirrors what is happening in the solar system. There seems an uncanny correspondence between some simple principles of geometry in one man’s mind and the movements of planets millions of miles away.

To explain his ideas to others, Newton had to resort to pictures, diagrams, and arguments on the printed pages of his great book. The book itself was material, and copies survive to this day. While the thinking that went into writing the book was fleeting and perished along

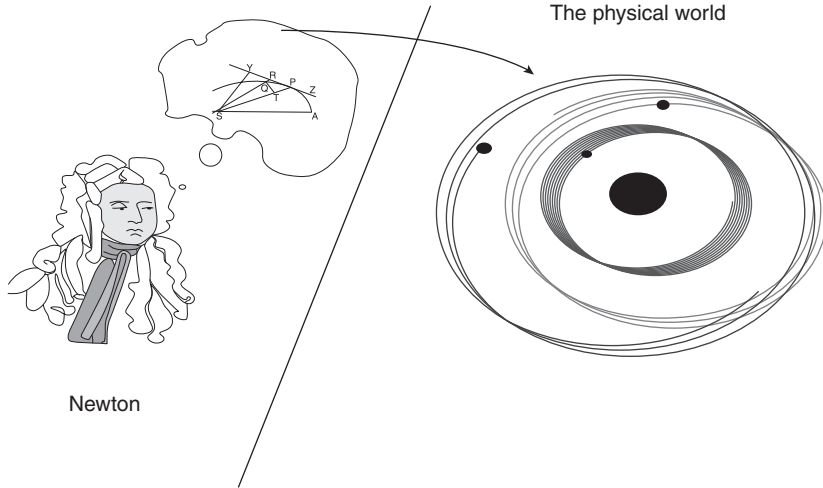


Fig. 2.1 Newton thinks.

with the man, the maths seems eternal, independent of the man and of his physical printed book. It is just as able to predict heavenly motions now as it was in the seventeenth century.

If we think of things in this way, as a correspondence between two quite different domains—that of thought and that of reality—the whole process seems so remarkable as to tempt us to ascribe some mystical properties to mathematics:



But now look at Fig. 2.2, which depicts the early days of space exploration. A boffin sits at his old valve computer and gets it to work out the course that will take a probe to Mars.

At one level, this shows the same process as in Fig. 2.1, but the very physicality of that big grey metal box in front of the boffin hints at something different. The similarity is that Newton’s laws, and their

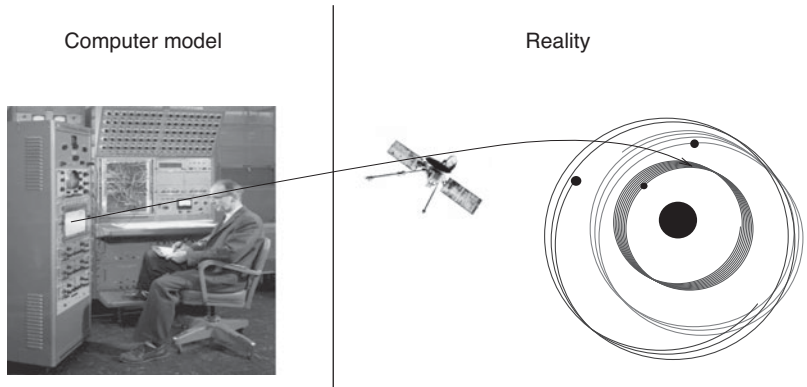
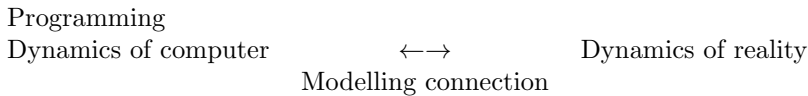


Fig. 2.2 NASA computes.

associated mathematics, are being used in each case. But the fact that the calculations are now taking place in a machine makes it harder to see the process as being one of a correspondence between mathematical thought and reality.

To do his calculations, the NASA scientist would have had to feed the computer with a whole lot of data obtained by astronomers. He would have had to develop programs to represent the dynamics in question. And then he would have set the machine working. We say that he would have had to develop programs, but that is not strictly necessary. The computer in the picture is actually an analogue one, which was programmed by rewiring it using the patch panel behind the operator.

So the correspondence here is actually between the dynamics of one physical system, the analogue computer, and the dynamics of another—the rocket that the designer wishes to control:



The above diagram is representative of a lot of what we now call ‘computing’. In this book, we will be examining just how it is that we can set up a physical system to emulate another system. Our approach is that of practical computer scientists concerned with real computing systems, ones that are either buildable now or could in principle be built. We think that this computer science perspective also helps with other problems raised by Wigner:

The great mathematician fully, almost ruthlessly, exploits the domain of permissible reasoning and skirts the impermissible. That his recklessness does not lead him into a morass of contradictions is a miracle in itself: certainly it is hard to believe that our reasoning power was brought, by Darwin’s process of natural selection, to the perfection which it seems to possess.

(Wigner, 1960).

Does it demand a very high level of evolved reasoning power to do what a great mathematician does?

Does this make unreasonable demands on evolution?

The objection that we never had much selective pressure on us to master calculus during the Palaeolithic is fair enough. But need we imagine a specific selective pressure in the past for every behavioural trait that we observe in the present?

In one sense, the answer has to be no. The diversity of human cultural behaviour is so great that nobody seriously contends that it is all in the genes—the social behaviour of the bee, yes, but the social behaviour of humanity, no. The calculus is clearly a cultural product. It is something that arose at a specific time with Newton and Leibniz, something that is learned by individuals rather than being an innate skill. This, however, still leaves open the question as to why we are able to do the reasoning

required by calculus. The specific details of the calculus had to be developed by Newton, but why did he or any other person have the reasoning ability to deal with something so abstract?

One possibility is to consider the implications of a very simple computer, the Turing Machine, described in Chapter 4. This simple device has a universal mathematical ability.

The key point is that we do not need anything more than a Turing Machine equivalent process on the part of the mathematician. A mathematician does not master calculus just by innate mental power. He or she uses a set of learned rules plus external aids—chalk and blackboards, and so on. Our dependence on external computing aids, from simple paper and pencils to digital computers, is another theme that we explore.

The computational power of the Turing Machine is universal. Similarly, a person, with a sufficient starting set of mathematical rules plus external notational aids, also has a universal mathematical ability. We do not need to assume a series of specific evolved abilities in the brain to handle the whole diversity of maths. All we need is a certain threshold of ability. Provided that a certain threshold, the Turing Universality Threshold, is passed, then the computational system composed of mathematician, innate reasoning primitives, and external notations will also be universal:

It is true, of course, that physics chooses certain mathematical concepts for the formulation of the laws of nature, and surely only a fraction of all mathematical concepts is used in physics.

...

A possible explanation of the physicist's use of mathematics to formulate the laws of nature is that he is a somewhat irresponsible person. As a result, when he finds a connection between two quantities that resembles a connection well-known from mathematics, he will jump at the conclusion that the connection is that discussed in mathematics simply because he does not know of any other similar connection. It is not the intention of the present discussion to refute the charge that the physicist is a somewhat irresponsible person. Perhaps he is. However, it is important to point out that the mathematical formulation of the physicist's often crude experience leads in an uncanny number of cases to an amazingly accurate description of a large class of phenomena. This shows that the mathematical language has more to commend it than being the only language that we can speak; it shows that it is, in a very real sense, the correct language.

(Wigner, 1960).

A point made by Wolfram (2002) is that there may be multiple different ways of approaching this, with different mathematical models. What determines how it is represented is a matter of what mathematical tools were available to the modeller. Consider the problem of feedbacks occurring in cellular biology. This can be modelled using differential equations, process algebra, or Boolean algebra. Because the modelling is now done using a computer, all of these choices come down to the use of different software to model a real system.

We are no longer surprised to find multiple software packages available for some task. The mathematical techniques that software packages use

are one way in which these packages may differ. Wigner agrees that only a fraction of all the available mathematical concepts have turned out to be of use in physics. Similarly, only a tiny fraction of all the possible programs are suitable for a given computing task. What is interesting is not that some maths and some programs are useful, but the process by which useful programs and mathematical techniques are developed. Hamming observes that:

The idea that theorems follow from the postulates does not correspond to simple observation. If the Pythagorean theorem were found to not follow from the postulates, we would again search for a way to alter the postulates until it was true. Euclid's postulates came from the Pythagorean theorem, not the other way round.

(Hamming, 1980).

This is a very good point, and it fits in with the view that the development of mathematics should be seen a form of software development. Chapter 4 will look at the relationship between issues in software development and those that arise in mathematical proof.

One answer to Wigner's problem is that the simple maths and the simple geometric entities that he describes are systems with a low information content, and that they can be generated by processes with a low information content. This argument can be developed on the basis of material in Chapter 5.

2.2 Counting sheep

I have tried, with little success, to get some of my friends to understand my amazement that the abstraction of integers for counting is both possible and useful. Is it not remarkable that 6 sheep plus 7 sheep make 13 sheep; that 6 stones plus 7 stones make 13 stones? Is it not a miracle that the universe is so constructed that such a simple abstraction as a number is possible? To me this is one of the strongest examples of the unreasonable effectiveness of mathematics. Indeed, I find it both strange and unexplainable.

(Hamming, 1980).

In the above quotation from his paper 'The unreasonable effectiveness of mathematics', Hamming points to how computing got started. People learned to count out of practical necessity. Once animals had been domesticated and were being herded, the shepherds needed to know whether they had lost any sheep. Suppose you set off with a herd in the morning and then return to the pen at sunset. How, if you can't count, can you tell if you have lost any?

If sheep were more placid, you could line up your sheep and put stones in front of them, one per sheep. More practically, you could set up a pile of pebbles outside their pen. As the sheep come out, take pebbles from the pile and put them in a bag. Allocate six sheep to one shepherd and seven to another, and get them to lead the sheep up to the pasture and back. When the sheep come back, line the sheep up (or let them into

a pen one at a time) and pull one stone out of the bag for each sheep. If they do not correspond, then you know a sheep has gone missing (or another sheep has been picked up along the way). At this stage of development, the society need not have any specific language to refer to numbers; the stones substitute for words. Menninger (1992) cites the Wedda of Ceylon as still being at this level in the 1960s.

This can be extended to deal with the shepherd's two sons, each of whom takes part of the flock to different pastures. The elder son is given the greater part of the flock and a jar of his stones is set aside, one stone for each sheep that he led away—seven stones in Hamming's example. The younger son takes the rest, putting a stone in his own jar for each sheep that he takes. When they come back, the father can determine if either son has lost a sheep by comparing the stones with the sheep as they are penned for the night.

Hamming identifies in this simple procedure a real marvel. 'Is it not a miracle that the universe is so constructed that such a simple abstraction as a number is possible?' he writes. But is it really such a marvel?

What is required for it to work?

Suppose, instead of mere pebbles, that the shepherds went one better and made little clay models of their sheep, each in the likeness of a particular sheep. Given that people can become sufficiently familiar with their sheep to recognize them as individuals, this is not impossible. Now when Snowflake steps out of the pen, the shepherd puts the clay Snowflake into the jar; as Misty trips along, in goes clay Misty; and so on. In the evening, clay Snowflake comes out of the jar as the real Snowflake comes home. When, at dusk, clay Misty is left in the jar, the shepherd fears that poor Misty has met Mr Wolf.

This system is more complicated, since individual sheep must be recognized. But it brings out the essential properties of sheep and tokens that are exploited in other token-based counting mechanisms.

Is there anything remarkable about this?

Not really: all it requires is that stones and clay tokens don't evaporate, and that the bag or jar you kept them in has no holes. There is nothing mysterious about pebbles. The mystery only arises if you think that there is an abstract domain of numbers quite separate from counting technologies. If we focus instead on the actual historical development of enumeration systems, each step is pragmatic and understandable. From the use of tokens by herders stored in perishable bags, a second step developed. Suppose that we have a more advanced form of society, with a temple-state that administers economic resources. A herder is despatched from one settlement to another with a herd of sheep. He is given a bag to record, using tokens, the number of sheep he set out with. At the other end, the officials compare the sheep he delivered with the tokens. The problem is that if he simply takes a bag of sheep tokens, there is nothing to stop him meeting his brother along the way, diverting a couple of sheep, and throwing away a couple of tokens. Everything still tallies at the end of the journey.

To get round this, bags were replaced by sealed clay containers inside which the tokens were placed (Nissen *et al.*, 1993). These containers were marked with the temple seal to verify who had inserted the tokens. This provided a tamper-proof way of sending and keeping a count of stock.

Next, it was realized that it was possible to mark the outside of the container with impressions showing how many tokens were inside. If there were four sheep tokens inside, then a sheep token would be pressed into the outside of the container four times whilst the clay was still soft. It was but a small step, then, to dispense with the tokens inside and just use solid clay tablets, with an indicator on the front of what would previously have gone into the container. Several systems were initially used to write down numbers of tokens. Base 60 was used for inanimate objects, as in Fig. 2.3. A base 10 notation was used for animals, and an alternating base 60 and base 120 system for rations (Englund, 1996).

At this point, you have the development of a written number system, and with the symbols for the numbers, the idea of the numbers existing independently of the things that they counted took hold of people's imaginations.

If you look at the number system in Fig. 2.3, it uses a number of distinct signs for ones, tens, sixties, and so on. Within one of these scales, a number is denoted just by repeating the symbol. So the number three is three indentations using the stem of a small stylus, 30 is three indentations using the end of a small stylus, and so on. There is a limit to the number of units that we can take in at a glance. Our ability to take in a small number at a glance is called subitizing. Up to four dots, arranged randomly, can be recognized at a quick glance (Mandler and Shebo, 1982), but above four the time taken to recognize the number rises, Fig. 2.5 and above five dots people have to count. If the dots are arranged in regular patterns, however, this aids our ability to quickly judge how many there are. A number system such as the proto-Elamite one builds on our inherent subitizing ability to recognize the number of units and tens in a written number. If a people lack both the language to describe numbers and a technology to represent them, their ability to handle numerosity is limited to what is provided by our inherited subitizing faculty (Gordon, 2004). The subitizing faculty

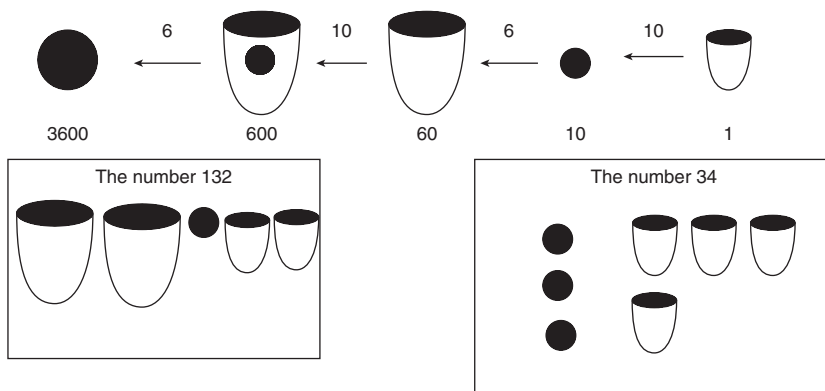


Fig. 2.3 The proto-Elamite numeral system using alternate base 10 and 6 for its numbers.

seems to be primitive in primates, rather than being a derived human characteristic. Chimpanzees share the ability to recognize small numbers and can even be taught to associate them with written numerals (Suzuki and Matsuzawa, 1997).

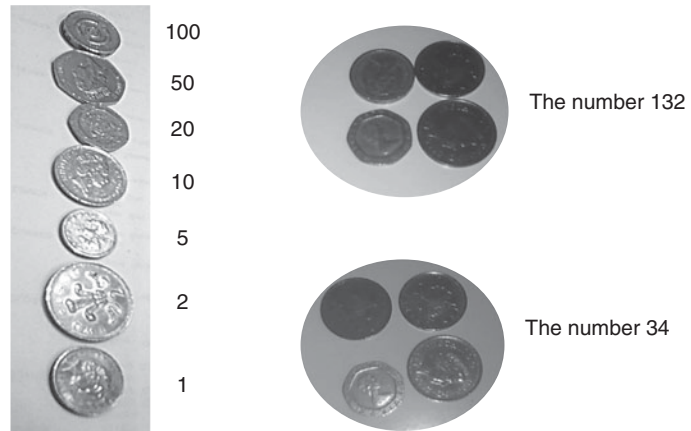
Subitizing is limited to very small numbers and seems to be a purely visual process. We have overcome these initial limits by the use of our language ability and through the use of external aids.

The use of multiple symbols, with a circle representing 10, is more complicated than simply putting stones in bags, or model sheep in jars. An intermediate step in this development was the use of special tokens representing ten sheep, or ten goats, and so on (Nissen *et al.*, 1993). We are so used to doing sums with Arabic numbers that computational systems based on tokens or pictures of tokens seem a bit cumbersome, until we realize that many of our daily computations are still done using these primitive computational aids. Figure 2.4 shows a token computing system that is still in widespread use. If you compare it to Fig. 2.3, you can see that the same basic principles are present in both systems. Different token types or symbols represent different scales of quantity, and the accumulation of tokens models accumulations of the things that they represent.

Let us consider how these could be used for our original example of counting sheep. Suppose that there are what we would call 32 sheep in the pen. The farmer has a bag of unit sheep tokens and another bag of tokens, each of which represents ten sheep. As the beasts are led out, the farmer puts unit sheep tokens into a jar one by one. When all the sheep are out, he piles all the tokens in his jar on the ground. He then takes groups of ten of the unit sheep tokens from his pile and puts them back in his unit bag. Each time he does this, he puts a ten-sheep token into his jar. At the end, he is left with two unit tokens in his original pile, and three of the ten-sheep tokens in his jar. To record the number of sheep being sent out with the shepherd, he now makes two cup-shaped marks and three dots.

The invention of tokens of different value overcame the limit to calculation posed by the weight of a mass of individual tokens.

Fig. 2.4 A token-based computing system used in part of Northern Europe.



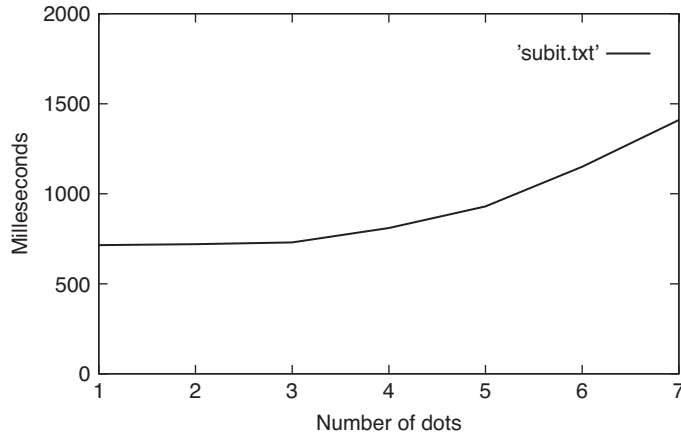


Fig. 2.5 The time taken to recognize how many dots are present in a group is almost constant up to four, and above that the recognition time increases.

So the process of recording a number on the clay tablet would have required some prior computational aids—tokens in our assumption. But we have skipped over the question of how the farmer got the piles of ten tokens. Well, the obvious computational aide for this is shown in Fig. 2.6—we are all born equipped to count to ten. This explains why ten occurs so often in notations for numbers. Think of the Roman or European number system in Fig. 2.7. The first two rows show the close relationship between the symbols for the numbers up to five and the corresponding hand-signs. The rows below that, for the tens and to 50 and 100s to 500, show a recursive application of the principle derived from the ones and fives.

Another technology that may have fed into the Roman number system is that of tallying, the carving of notches on sticks to count things (Ifrah, 1995). These markings remain in common use, using paper and pencil. They rely on our subitizing ability to recognize patterns of up to four strokes.

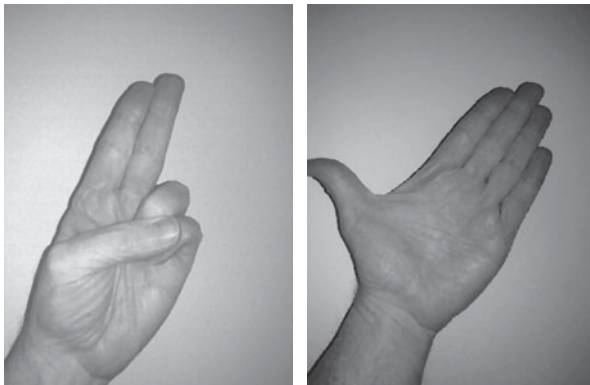


Fig. 2.6 A primitive digital computer displaying the numbers two and five.

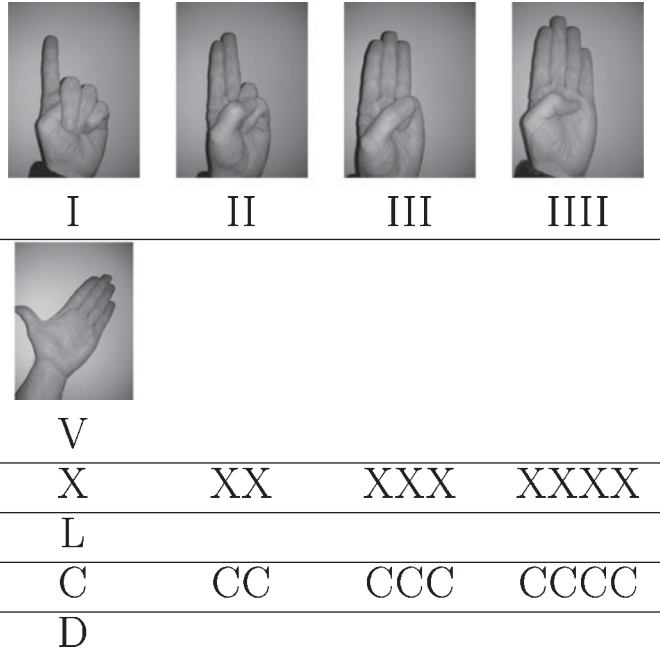


Fig. 2.7 Roman numbers and their origins in finger counting according to Ifrah.

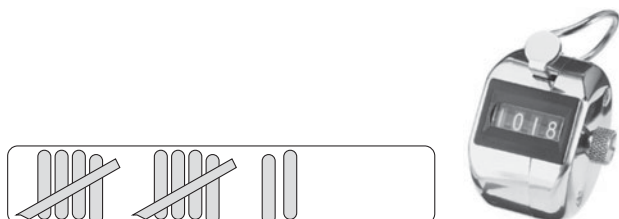
2.3 Counting materialized in our own bodily movements

2.3.1 Fingers

Counting on the fingers is both a motor process and a visual process, with the visual process acting as a reminder for the motor sequence. If the use of the fingers is as shown in Fig. 2.7, then visual reminders are not needed; the thumb holds down the fingers until they released. This allows one-handed counting whilst the visual attention and the right hand are used for another task. The thumb imposes a sequence on the release of the fingers, diminishing the mental effort involved. The same basic mechanism of counting to five that the fingers provide is carried over to tallying. This was initially done by carving notches in tally sticks as shown in Fig 2.8, but the word has persisted in our language to mean any process of counting objects.

Lakoff and Nunez (2001) argue that much of our mathematical ability is grounded in primitive conceptual and motor schema primitive to the

Fig. 2.8 Notched tally sticks are another way of counting in multiples of five. A tally for 12 is shown alongside a modern mechanical tally.



nervous system. Motor control processes, they argue, have a basic schema of components, some of which may be omitted in particular actions:

1. Readiness to act.
2. Starting up.
3. The main process.
4. Possible interruption or resumption.
5. Iteration.
6. Checking for goal achievement.
7. Performing completion actions.
8. The finished state.

A motor schema may include nested subschema that have a similar structure to the main schema. In the case of finger counting, this schema would map on to the following steps: (1) place thumb over index finger, tense index finger, watch for event; (2) observe an event—for example, a sheep leaving a pen; (3) move thumb, causing finger to fly up; (5) iterate, going back to step 1; (6) check whether all four fingers are released; (2) observe an event; (7) complete process by raising the thumb.

This is a comparatively simple iterative process, with the nervous sequencing proceeding independently of the count—the count itself being held on the hand as a mechanical register.

2.3.2 Voice

What computational capacity must our brains have to allow us to count aloud?

Counting aloud is a motor process, a more complex one than finger counting, since it is one in which the sequencing has to be all internally driven by the nervous system. The fingers are no longer there outside the brain to act as mnemonics. It presupposes a process something like that shown in Fig. 2.9. This involves a type of sequencing that electrical engineers have studied under the rubric of finite state automata. A finite state automaton is a physical system that has a finite number of states, transitions between those states, and actions. They were examined in the context of electronic control circuits by (Moore, 1956) and extended to linguistics by (Chomsky, 1956). They have since become a foundation block for the analysis of computer languages and the design of computer hardware.

Chomsky had been investigating the possibility of formally specifying natural languages. He had classified grammars into classes. These classes of grammars are now referred to as Chomsky class 0, class 1, class 2, and class 3 grammars. It turned out that Chomsky class 2 and class 3 grammars are most suitable for describing programming languages, and the concepts involved in these grammars are also relevant to understanding what goes on when a person counts out loud. To grasp what these different classes of grammars are, we need to go into a little formal notation.

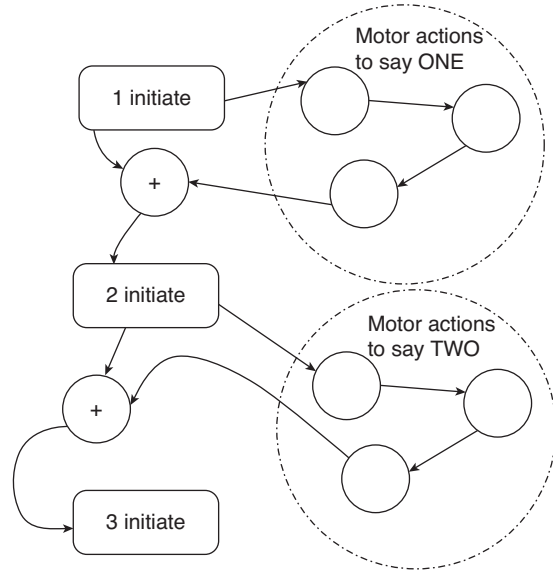


Fig. 2.9 Sequencing for counting.

The syntax or grammar of a language can be thought of as being made up of a four-tuple $(T; N; S; P)$, where:

- T stands for what are called the terminal symbols of the language. In a human language, these terminal symbols are the words or lexicon of the language. In a computer language, they are things such as identifiers, reserved words, and punctuation symbols.
- N stands for what are called the non-terminal symbols of the language. In a human language, a non-terminal would be grammatical constructs such as a sentence, a noun clause, or a phrase. A computer language is likely to have a large number of non-terminals, with names such as clause, statement, and expression.
- S is the start symbol of the grammar. It is one of the non-terminals, and its meaning will become clear shortly.
- P is a set of productions or rewrite rules. These tell you how to expand a non-terminal in terms of other terminals and non-terminals.

This sounds a bit dry, but it will be clearer if we give an example. Suppose that we wish to define a grammar that describes the ‘speech’ of a traffic light. A traffic light has a very limited vocabulary. It can say ‘red’, ‘amber’, or ‘green’, or ‘red-and-amber’. These are the terminal symbols of its language.

$$T = \{ \text{Red, Green, Amber, Red-and-amber} \}$$

At any moment in time, the traffic light is in a current state and after some interval it goes into a new state that becomes its current state.

Each state is described by one of the colours of T . This can be expressed as a set of non-terminal symbols, which we will call

$$N = \{ \text{going-red, going-green, going-amber, going-red-and-amber} \}$$

We will assume that when the power is applied for the first time, the light enters state going-red. Thus

$$S = \text{going-red}$$

A traffic light has to go through a fixed sequence of colours. These are the syntax of the traffic light language. Which sequence it goes through is defined by the productions of the traffic light language. If the light is in going-red, then it must output a red light and go into going-red-and-amber. We can write this down as follows:

$$\text{going-red} \rightarrow \text{Red going-red-and-amber}$$

This is an individual production in the traffic light language. The whole set of productions is given in Fig. 2.10.

This combination of $(T; N; S; P)$ ensures that the only sequences of colours allowed by the traffic light are the cycles shown in the diagram.

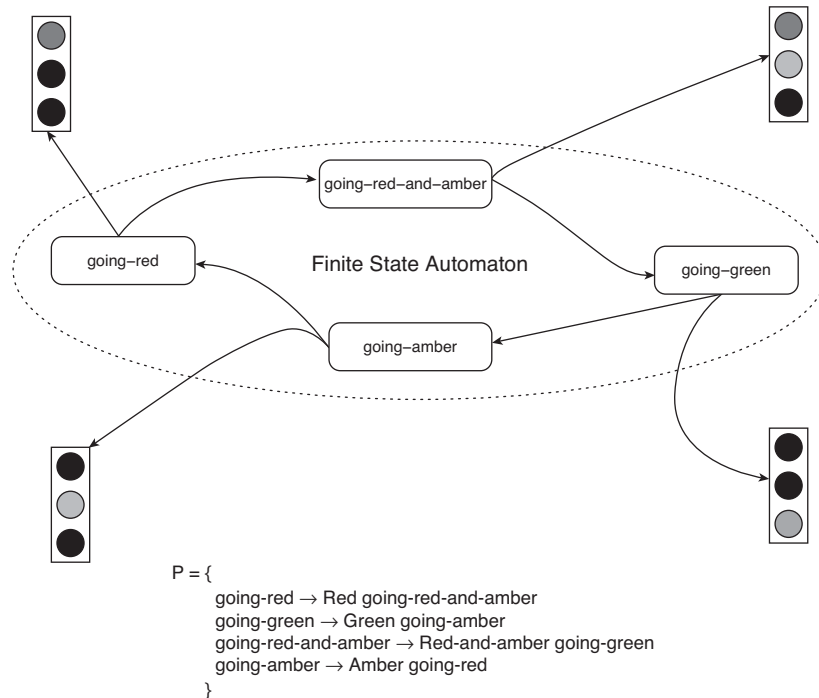


Fig. 2.10 Production rules and behaviour of a traffic light.

The traffic light grammar is what Chomsky classified as type 3 and what computer scientists now call a regular grammar. Regular grammars can be produced by finite state machines. Finite state machines are popular among electrical engineers constructing simple control devices, since they can be constructed using very few components as shown in Fig. 2.11. If we look back at Fig. 2.9, you can see that the motor control structure shown there is similar to the finite state machine in Fig. 2.10. The difference is that the traffic light control process goes in a cycle, whereas counting is potentially unlimited. What sort of grammar is required to express, say, counting up to 100 in English?

Clearly, we could do it with a class 0 grammar, but that would require the brain to hold a finite automaton with 100 states for the purpose. This is obviously doable, since people can learn poems that are much longer than that, but it does not seem plausible that it should be done that way, since to count to 1000 we would have to learn a finite state machine with 1000 states. Instead, we make use of patterns to save on learning so many things by heart. We make use of the repeating pattern from one to nine in counting aloud from 21 to 29, or 31 to 39. The technical issue is how this sequence is made use of. If we counted as follows:

twenty, one, two, three, four, five, six, seven, eight, nine, thirty, one, two, three,...

then what Chomsky called a class 2 or *context-free* grammar would suffice for counting. Class 2 grammars have productions of the form

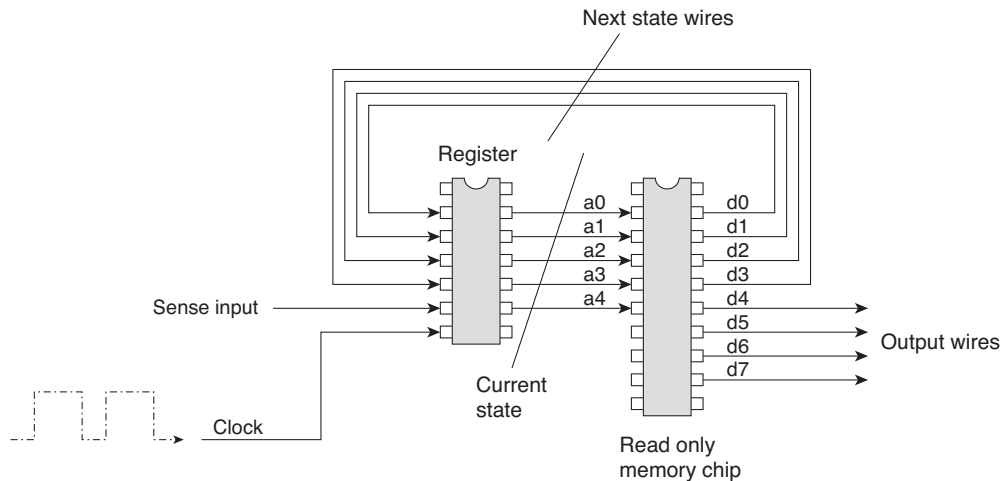


Fig. 2.11 A finite state machine suitable for simple sequencing can be built out of a register chip and a read only memory (ROM) chip. In the picture, the register latches a new address for the ROM each clock cycle. The ROM outputs a next state and a set of wires that go to control things such as the lights in Fig. 2.10. Thus each cycle it jumps to a new state. The behaviour of the machine is then defined by the table loaded into the ROM. One or more sense inputs can also be provided, which provide additional address inputs. The sense inputs could be fed by things such as the pedestrian crossing button. The presence of the sense inputs allows the next state transitions to be *conditional* on the sense input. Since the sense inputs are addresses, each sense input used means a doubling of the table size to handle both the true and false variants of the sense input.

a → **b**

where **a** is a non-terminal symbol and **b** is some combination of terminals and non-terminals. Class 2 grammars are more powerful than class 3 grammars because they allow the use of nested patterns. We can, for example, write down context-free grammar that defines how we say the numbers from 1 to 999 999 in English. The grammar below is an extension and clarification of one given by Power and Longuet-Higgins (1978):

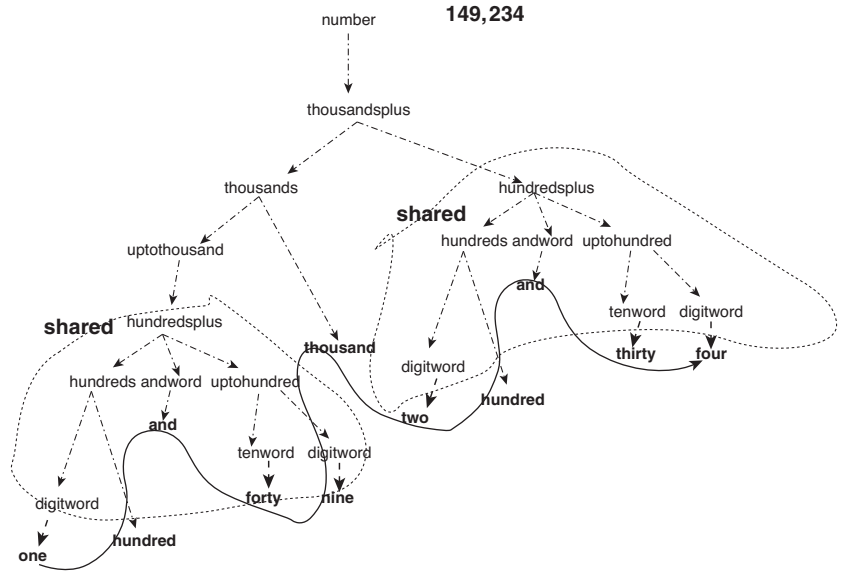
| | word | number | role |
|------------------------------------|----------|--------|--------------|
| | one | 1 | digitword |
| | two | 2 | digitword |
| | ten | 10 | firstten |
| | nine | 9 | digitword |
| | ... | | |
| <i>Lexicon of English Numbers.</i> | eleven | 11 | teenword |
| | ... | | |
| | nineteen | 19 | teenword |
| | twenty | 20 | tenword |
| | ... | | |
| | ninety | 90 | tenword |
| | hundred | 100 | hundredword |
| | thousand | 1000 | thousandword |

Production rules. number → uptothousand|thousands|thousandsplus
 andword → and
 numberword → tenword|teenword|digitword|firstten
 uptohundred → numberword| tenword digitword
 hundreds → digitword hundredword
 hundredsplus → hundreds andword uptohundred
 uptothousand → uptohundred|hundreds|hundredsplus
 thousands → uptothousand thousandword
 thousandsplus → thousands andword uptohundred|
 thousands hundredsplus

In the above rules, we use the standard convention that | marks an alternative production. Consider the number 149 234, which we pronounce as one hundred and forty nine thousand two hundred and thirty four. We can draw a parse tree of how this can be produced by the above grammar as shown in Fig. 2.12.

The same grammatical derivation allows us to say the hundreds of thousands as allows us to say the hundreds. This grammar is certainly no more complex than we use in day-to-day speech, so saying an individual large number just involves learning a few additional rules of the type we

Fig. 2.12 Parse tree for the linguistic production ‘one hundred and forty nine thousand two hundred and thirty four’.



are already familiar with. It is a well-established principle of grammar theory (Hopcroft and Ullman, 1979) that context-free languages can be produced by a stack automaton. A stack machine is the composition of an FSM and a stack on to which the state word can be pushed or from which the state word can be popped. This allows the automaton to perform nested operations. In the above case, the nested operation is the saying of a number in the range 1–999. Is this enough for counting?

No, because a classical stack machine can only look at the top of the stack. Suppose that we are counting as follows:

one hundred and forty nine thousand two hundred and thirty four
one hundred and forty nine thousand two hundred and thirty five
 ...
one hundred and forty nine thousand two hundred and thirty nine
one hundred and forty nine thousand two hundred and forty
one hundred and forty nine thousand two hundred and forty one

The italic letters indicate the words that we have to ‘stack’ in our minds to go on to the next step. The transitions between the final digits could be carried out using the simple automaton in Fig. 2.13. This may look like a simple stacking operation, but it is not.

In order to say each number in the correct order, we have to have access to all the stacked words so that we can repeat them, whereas a classical stack machine can only see the top word of the stack. This means that the grammar required for counting, as opposed to saying one number, is *context sensitive*. But since natural languages do contain context-sensitive components to their grammars, it is plausible that the

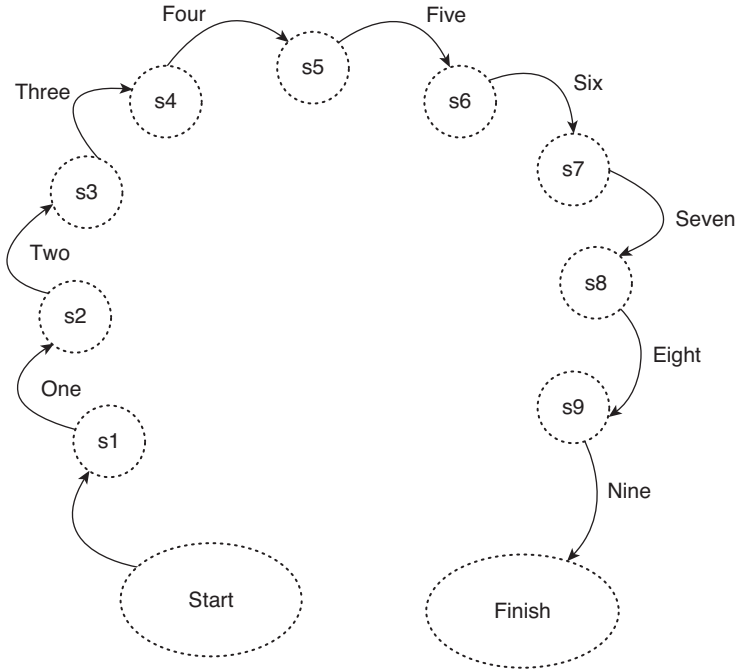


Fig. 2.13 The repeated pattern invoked when counting aloud in English.

process of counting rests upon the same linguistic computing ability that is used to produce normal speech.

If we relied on the primitive ability of the motor control system to perform sequences of actions that correspond to regular grammars, then the ability to count would have been greater than that allowed by subitizing, but still small. By using our more advanced grammatical abilities, we can overcome these limits and count much higher.

2.4 From ‘*aides-memoire*’ to the first digital calculating devices

The process of counting aloud is error prone. We can easily lose track of the numbers we are reciting. So the fallibility of our short-term memory poses another limit, this time to unaided mental computation—hence the reliance on tallies and counters for practical tasks. Tallies can be simple marks as in Fig. 2.8 or they can be mechanical devices. But these just count, the most basic computing operation. How did we get beyond counting to adding?

Well, if you are counting with pebbles in jars, then all you need to do is pour the contents of one jar into the other and you have done an addition. This operation is still reflected in language. When baking a cake, you add the sugar to the flour in the bowl. Physical addition as pouring is the primitive operation, arithmetic addition the derivative operation. If the materials you are pouring are counting pebbles, then pouring has

the same effect as addition. Take two flocks of sheep into a single pen one after the other; supposing that you have pebble jars to count each flock on its own, then pouring one jar into the other gives you the total number of sheep in the pen. There is no mystery to this—all it requires is that our reliance on sheep and pebbles doesn't vanish mysteriously.

Next, assume that you have tokens in your jars, tokens that can represent different numbers of sheep as introduced on page 14. Again, you simply add the contents of one jar to the other and you will have an accurate record of the number of sheep in the two flocks. Why does this work?

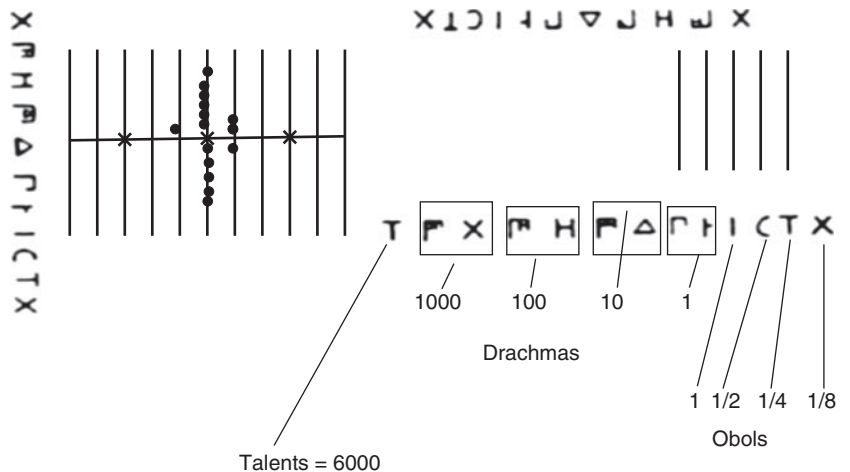
It works because we have a procedure for translating the high-valued tokens back into unit tokens.

We could either take both jars individually and whenever we find a ten-sheep token in the jar replace it with ten individual sheep tokens, or we could wait until after we have put all the mixed tokens into a single jar before we do the translation. Whichever order we do it in, we end up with the same number of unit tokens in the final jar. Nowadays, we teach children that $10(b + c) = 10b + 10c$, which is just an abstract way of talking about the same process.

The old practices persist. Who, when collecting the coins from both trouser pockets into one pocket, bothers to count the coins first and add their numbers?

There's no need—the old addition-as-pouring method still works well enough. For business or state accountants though, more formal ways of dealing with collections of coins had to be developed, and for this purpose people reverted to using beads or stones, the value of which depended on their position on a reckoning board or counting table, *abakion* in ancient Greek, an early example of which was found in Salamis, and is shown in Fig. 2.14. It appears to have been designed to do additions in units of the then Greek monetary system of talents, drachmas, and obols. Unlike counting with coins, where the numerical value is marked on the coin,

Fig. 2.14 The Salamis reckoning table. The Greek letters denote units of coinage and indicate the values of counters in different columns: one drachma = six obols. It is shown ready to add 51 drachmas (lower half) to 162 drachmas (upper half). To perform the addition, the counters on the lower half are pushed up to the upper half, moving the corresponding row of counters in the upper half up to make space. Subsequently the representation is renormalized by removing ten counters from any drachma column with more than nine and adding one to the column to the left.



with the abakion, the value assigned to a counter depended the row in which it was placed.

If two numbers are to be added, tokens are arranged above and below the line as shown. To perform the actual addition, counters on the lower half are slid to the upper half, sliding the corresponding rows of counters in the upper half up to make space. This step has already performed the physical addition. The top half of the columns now contains the correct value, but it may be denormalized. That is to say, some columns may have too many tokens in them—more than nine for the drachmas, but more than one for the $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ obol columns. Subsequently, the drachma representation is renormalized by removing ten counters from any column with more than nine and adding one to the column to the left. The obol representation is renormalized by adding one drachma and taking away six obols if there are more than five obols in total.

The step from the Greek *abakion* to the more recent abacus is a small one. The main shift is that the Salamis *abakion* was a two-register device, whereas most modern abacuses are single register. According to Menninger (1992, pp. 305–315), the Roman hand abacus (Fig. 2.15), derived from the *abakion*, spread to Asia, where modified variants of it are still used. In the twentieth century a hand-operated device that was, in essence, an abacus was mass produced as a pocket calculator, as shown in Fig. 2.15(b). One of the authors remembers using these in school.

The Romans continued to use large reckoning tables like the Salamis one, on which they ‘calculated’ using *calculi* or small pebbles. The pebbles that first served as counters were later replaced by discs like those used in the modern game of draughts.

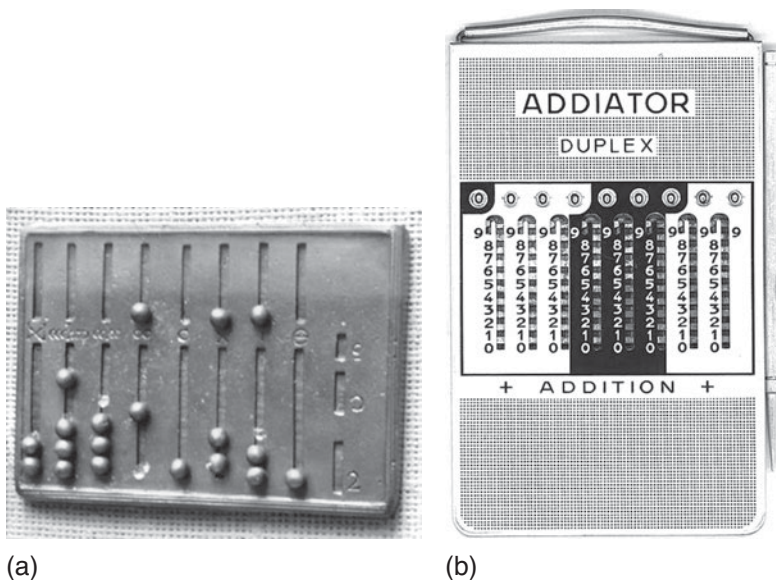


Fig. 2.15 Pocket abacuses two millennia apart. (a) A diagram of a Roman hand abacus. Note that it holds two rows, one to hold the units and the other the fives; successive columns indicate powers of ten. The body was a flat bronze plate with movable studs that slid in slots. (b) The Addiator, a pocket abacus from the early twentieth century that used sliders rather than beads and that was operated with a stylus. To add, one inserted the stylus and moved the appropriate slider down—unless a red colour showed, in which case one moved it up and round to perform a carry.

2.4.1 Multiplication

We have been arguing that computing techniques were ways in which people predicted or modelled the world. This is clear enough with counting and adding, which originate with the need to model movable possessions. The requirement to multiply arises as society becomes more complex. Architecture requires multiplication to determine quantities of bricks needed to construct walls, pyramids, tiled floors, and so on. The organization of work teams requires calculation of the total quantities of rations that people will eat. Extensive trade involves calculating the price of a cargo, given the price of a unit. Suppose that we know the sides of a rectangular floor to be tiled as ‘12 by 17’. The 12 and the 17 can be represented in some way as counters, tokens, or in a written notation—the answer likewise. A simple technique is to lay out pebbles in a regular rectangular pattern, 12 pebbles by 17; the resulting group of pebbles can then be carried to the tile maker, or counted and the result taken to the tile maker. This is direct physical modelling of the floor to be tiled with the pebbles, but at the same time it performs what we would now call multiplication, though we would scarcely recognize it as such.

Doing multiplication this way is slow. We tend to think of multiplication as being some form of shorthand procedure that can achieve the same result. Menninger describes how reckoning tables were used to compute multiplications by a much quicker method. Figure 2.16 shows a four-register reckoning table whilst 54 is being multiplied by 32.

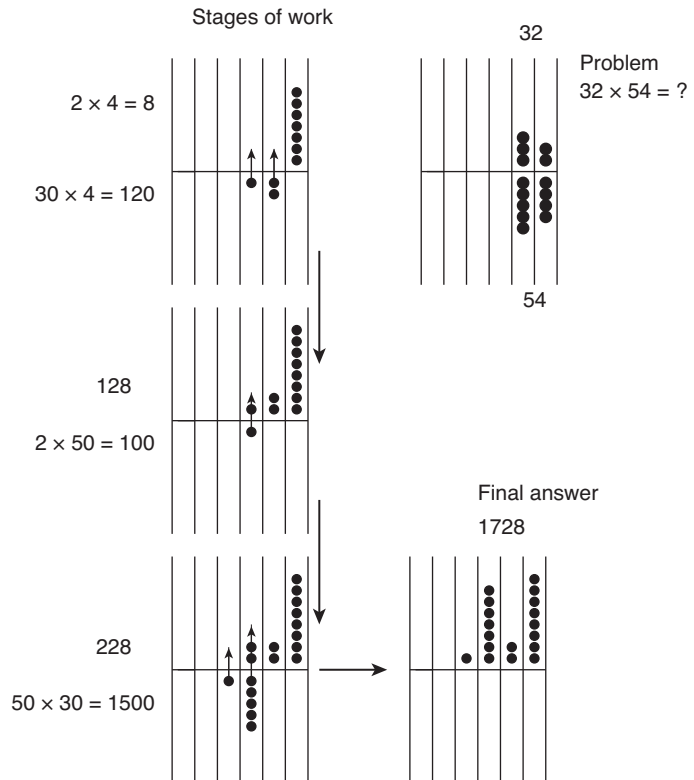


Fig. 2.16 Multiplication on a four-register reckoning table. The left pair of registers is used for working; the right to hold the two numbers to be multiplied. The working registers are shown in the successive phases of the calculation.

Doing the multiplications requires that the operator learn by heart the multiplication tables up to ten—just as children have to do now. Here, we encounter another of the great abiding accelerators of computation—the look-up table. Tables can be memorized, or they can be written down—a large part of the Rhind papyrus, one of the first maths texts known, consists of tables. Their great merit is that they remove the cost of repeated calculation, substituting for it the much easier task of looking something up. Here, a temporal limit to calculation is avoided by the use of a less complex process.

We can summarize the technologies for computation described so far in the following table:

| Technique | Skill | Operations | Range |
|---------------------|--------------------|----------------------|--------------|
| Subitizing | Inbuilt | Counting | Only to four |
| Counting on fingers | Learned | Counting adding | To ten |
| Counting aloud | Rote learning | Count | To hundreds |
| Unary tokens | Simple learned | Count, add | To tens |
| Scaled tokens | Recursive, learned | Count, add | To thousands |
| Base 60 | Recursive, learned | Count, record | Millions |
| Reckoning tables | Skilled, recursive | $+$, $-$, \times | Millions |

3

Mechanical computers and their limits

| | |
|---|----|
| 3.1 Antikythera | 28 |
| 3.2 Late mechanical computers | 35 |
| 3.3 Analogue mechanical multiply/accumulate | 39 |
| 3.4 Mechanizing the abacus | 42 |

In principle, with large enough reckoning tables, papyrus for notes, and sufficient time, very large and complex calculations could be done. But the process of reckoning is slow and, when dealing with extended calculations, human error creeps in. To overcome these limits, mechanism was needed.

Mechanical computing started surprisingly early. The earliest known mechanical computer has been dated to between 150 BC and 100 BC (Freeth *et al.*, 2006)—but given its sophistication, it may be supposed that it had earlier predecessors.

3.1 Antikythera

In 1900, a group of sponge divers sheltering from a storm anchored off the island of Antikythera. Diving from there, they spotted an ancient shipwreck, with visible bronze and marble statuary. A number of valuable artefacts were recovered and taken to the National Museum in Athens. Further diving in 1902 revealed what appeared to be gearwheels embedded in rock. On recovery, these were found to be parts of a complicated mechanism, initially assumed to be a clock. Starting in the 1950s and going on to the 1970s, the work of de Solla Price (1959, 1974) established that it was not a clock but some form of calendrical computer. A key to this clarification was the use first of gamma ray photography and then of computer tomography (Freeth *et al.*, 2006) to reveal details of the mechanism beneath the accretions of millennia of marine corrosion. A number of reconstructions have since been made, both physical and in software, one of which is shown in Fig. 3.1. It is now clear what the machine did, although why it was built remains a matter for speculation. Our description of its function follows Spinellis (2008) and de Solla Price (1974).

The machine had two faces: the front with a central dial, the back with two major and two subsidiary dials (Fig. 3.1(a)). The front dial had two pointers, one of which bore a solar and the other a lunar globe. The pointers showed the positions of the sun and moon, both in the zodiac and relative to the calendar. The calendar was denoted by a scale ring with 365 days. The scale could be advanced by one day every 4 years, using a retaining pin that could be fitted into one of 365 holes in the frame behind the movable scale.

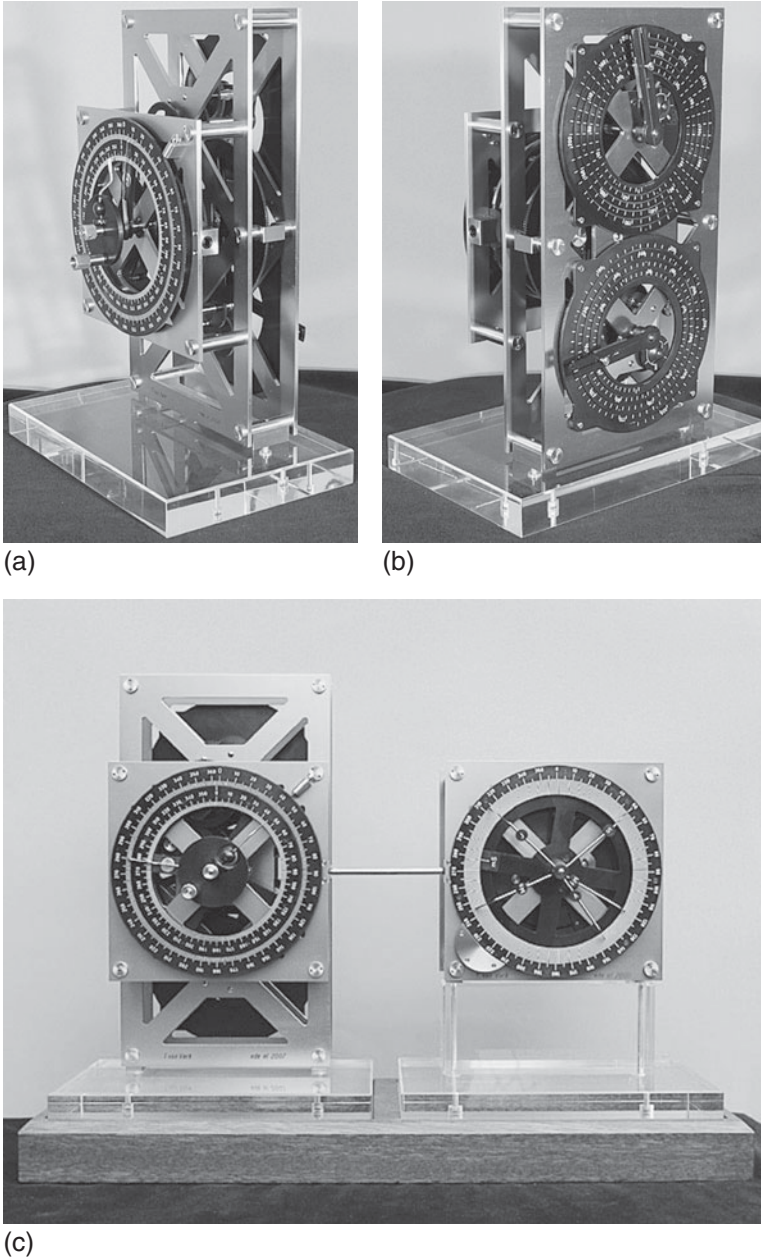


Fig. 3.1 A modern physical reconstruction by T. van Vark, reprinted by permission of the builder. (a) The front dial, showing the lunar and solar pointers against zodiac and calendar rings. (b) The rear face, showing the Metonic and Saros cycle dials with spiral scales. (c) The mechanism linked to and driving the hypothesized planetary dials.

The back dials are more complex. The upper back dial shows what is called the Metonic cycle of 235 lunar months.¹ This is the period in which the the lunar and solar calendars come into synchrony. Nineteen solar years correspond to 235 lunar months, so after the end of this cycle the phases of the moon will again be the same on the corresponding calendar day. In order to gain greater precision on the dial, it is arranged in a five-deep spiral, with a pin in a moving slider on the pointer going into

¹A lunar month is the period between corresponding phases of the moon in successive lunar cycles.

the spiral. As the pointer turns, the slider moves gradually outwards. The spiral arrangement increases the angular resolution of the dial.

The lower back dial uses the same spiral dial mechanism, but in this case it displays the Saros cycle of 223 lunar months. The Saros cycle is relevant for predicting lunar and solar eclipses. If a lunar or solar eclipse occurs, then another similar one will occur 223 lunar months later. The Saros cycle is $6585\frac{1}{3}$ days. Since it is not an integer number of days, the geographical position from which eclipses are seen shifts by 120° each cycle. The 54-year Exeligmos cycle is needed to bring the geographical position from which lunar eclipses can be seen back into alignment. For solar eclipse paths of visibility, the calculation is more complex. A small auxiliary dial shows the Exeligmos cycle. Another auxiliary dial shows the Callipic cycle, which is four Metonic cycles less one day. The Callipic cycle improves the accuracy of reconciliation of the lunar and solar calendars.

The interpretation of the dials is based both on the fragmentary captions that have survived on their surfaces and on a mutually consistent reconstruction of the internal workings of the machine. The sophistication of the mechanism, when uncovered by Price, was astonishing, given what had previously been known about ancient Greek computing technology. It involves a large number of gear wheels with complex interactions. In order to understand these, we need to grasp certain preliminary principles of how gearwheel computing works.

3.1.1 Addition

The first point is to understand that wheels can implement addition by addition of angles. In Fig. 3.2 we see a pair of concentric wheels, with digits round the edge being used for addition. Assume that the two wheels are free to rotate both absolutely and relative to one another. To add 3 to 4, we first rotate the inner wheel until its zero is lined up with 3 on the other wheel, then read off the number on the outer wheel corresponding to 4 on the inner wheel. This basic mechanism was used in many ready reckoners and circular slide rules.

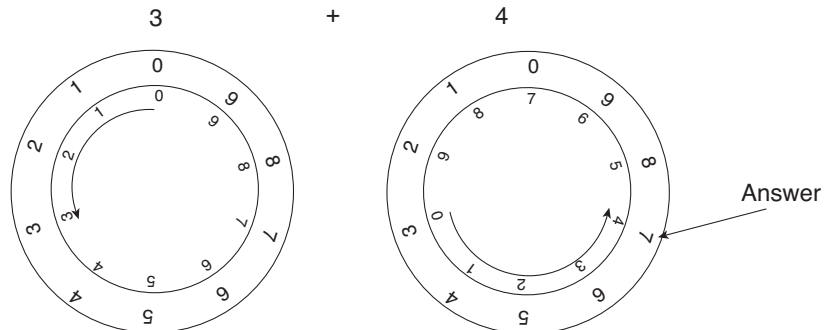


Fig. 3.2 Wheels can do modular addition by addition of angles.

3.1.2 Multiplication by a constant

If two wheels contact tangentially, wheel A has diameter a , and wheel B has diameter b , then wheel A will rotate at a rate of $-b/a$ times the rate of rotation of wheel B. If the wheels are cogwheels with matching sizes of teeth, then the ratio of their diameters will be the ratio of their tooth numbers. This is shown in Fig. 3.3.

Note that this implies multiplication by rational numbers rather than arbitrary real numbers.

3.1.3 Differential gears

If gear C is mounted on gear B, and if B and A rotate at the same rate, then so will C:

$$\Delta A = \Delta B \Rightarrow \Delta A = \Delta C$$

In the frame of reference of B, the rotation of C will be $-c/a$ of the rate of rotation of A also in the frame of reference of B. But the rotation of A in the frame of reference of B is just $\Delta A - \Delta B$. So the full equation for the rotation rate of C is

$$\Delta C = \Delta B + (\Delta A - \Delta B)\frac{-c}{b}$$

or

$$\Delta C = \frac{c+b}{b}\Delta B - \frac{c}{b}\Delta A$$

This mechanism, Fig 3.4, allows the computation of linear functions of differences between rates.

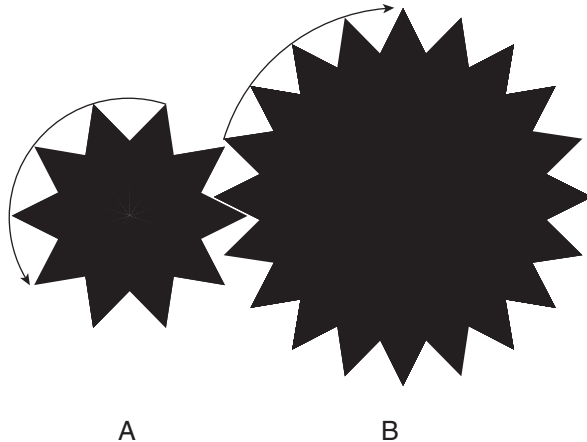


Fig. 3.3 Multiplication by a constant. Rotation of wheel A by -144° will result in rotation of wheel B by 72° .

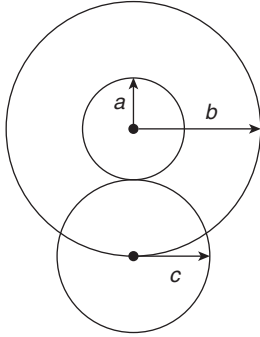


Fig. 3.4 Differential gearing.

3.1.4 Non-linear functions

The above three principles would be sufficient to construct a mechanical model of the positions of the moon, sun, and planets if these all followed uniform circular orbits. A problem arises, however, with the elliptical orbit of the moon. The angular speed of the moon is not constant. Kepler's law, which states that equal areas are swept out in equal periods by its orbit, means that close to perigee its angular velocity against the fixed stars is greater than at apogee. This is termed the 'anomalous motion' of the moon.

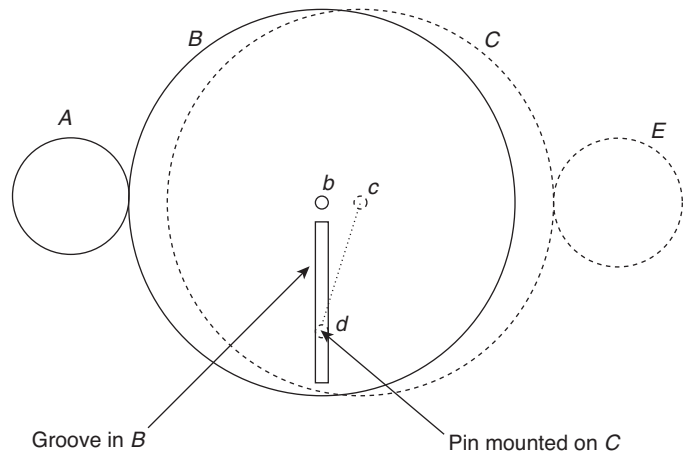
Apollonius of Perga (*c.* 262 BC to *c.* 190 BC) proposed to model the lunar motion by either of two models. The one shown in Fig. 3.6 involves the assumption that the moon rotated in a circular orbit around a point some distance from the centre of the earth (Neugebauer, 1955). The other is the theory that we are more familiar with from Ptolemy, that the moon moved in an epicycle on an otherwise circular orbit. Apollonius proved the equivalence of the epicycle and eccentric models.

The major contribution to the anomalistic motion of the moon is the fact that the elliptical orbit is off centre with respect to the earth. Compared to this, the deviation from circularity of the ellipse is relatively small. Thus an eccentric circular model gives a very good approximation of the lunar position. A computing machine such as the Antikythera, which uses the mechanism in Fig. 3.5 will give predictions of lunar position that are as accurate as the naked-eye astronomy of the ancient world could observe.²

Our current records of ancient astronomy do not indicate knowledge of Kepler's law at the time the Antikythera device was built. Despite this, the device contains a mechanism to approximate the effects that we now attribute to Kepler's law. The technique used is to have two wheels with slightly different axes, coupled via a pin-and-slot mechanism like that illustrated in Fig. 3.5.

²For a heterodox view of the optical technology available to Greek astronomers, see (Temple, 2000).

Fig. 3.5 Variable-speed coupling using eccentric axes. Wheel *B* has a different axis *b* than that of wheel *C*. A slot in *B* is engaged by a pin *d* mounted on *C*. *C* will complete one revolution for every revolution of *B*, but its rotation speed will change as *B* goes round, being fastest relative to that of *B* when the pin is horizontally to the right of *b* and slowest when it is horizontally to the left. When simulating lunar orbital motion, the more rapid orbital motion at perigee would be simulated by having the pin *d* horizontally to the right at perigee.



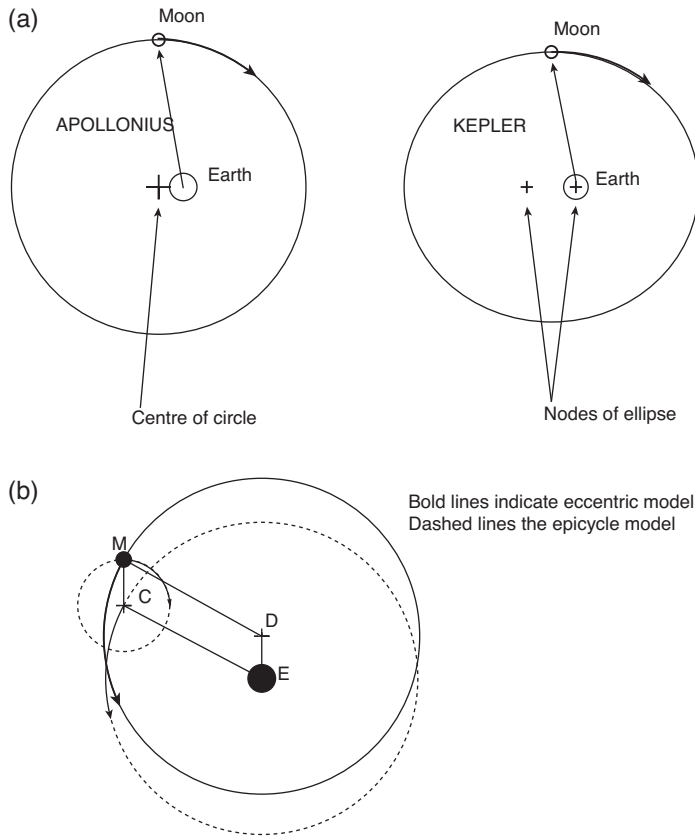


Fig. 3.6 (a) The contrasting models of lunar orbits proposed by Apollonius and Kepler. (b) Apollonius' proof of equivalence of the epicyclic and eccentric models. E is the earth, M the moon, D the centre of orbit of the eccentric model, and C the centre of the epicycle. If the rotation of the epicycle is of exactly the same magnitude but in the reverse direction to the orbital rotation, then CM must be parallel to ED, EDMC must be a parallelogram, and so the path of the moon is in each case a circle centred on D.

The mechanical approximation mechanism used in the Antikythera device corresponds closely with the eccentric model of the lunar orbit. The line of sight from the earth to the moon in Apollonius' model (Fig. 3.6(a)) is physically modelled by the groove in wheel B (Fig. 3.5). The moon in Apollonius' model is represented by the pin d and the centre of the lunar orbit by the axis c (both shown in Fig. 3.5). If we compare the two figures, we can see that the mechanism in Fig. 3.5 directly implements Apollonius' model of the lunar orbit. The mechanism is a simplified representation of a component actually used in the Antikythera device.

It is easier to construct an eccentric coupling of the sort shown in Fig. 3.5 than it is to make a direct mechanical implementation of Kepler's law. The choice of model proposed by the Greek astronomers to explain the lunar orbit may thus have been influenced by what they could feasibly build into a computer. Apollonius of Perga, who proposed the eccentric model for lunar orbits, also developed the theory of ellipses in his work on conic sections, so he would have been well equipped to propose ellipses as a basis for orbits. The preference of the Greek astronomers for circular motions might have been influenced by a concern for mechanization.

We talk about scientists having a ‘model’ for some physical process that they seek to explain. By this, we tend to mean a conceptual model. But a conceptual model does not produce numerical results until the conceptual model is implemented in a physical model. The Antikythera device is a beautiful illustration of the interaction between conceptual and physical models. There is a one-to-one correspondence between the geometrical mechanisms proposed by the ancient astronomers, and the physical geometry of one of the computing machines that they used. Today, the correspondence is less evident, since we use general-purpose computers. But the correspondences will still be there. The conceptual model will now be expressed algebraically, and that algebra will be translated into expressions in Fortran, Matlab, or some other computer notation for explicit calculation. The difference is that we can now have explicit representations of algebra rather than explicit representations of geometry in our machines.

With a device such as the Antikythera mechanism, predicting eclipses became just a matter of turning a handle, something far simpler and less error prone than the numerical solution of kinematic equations using reckoning tables.

3.1.5 Was the Antikythera device an analogue computer?

In the web literature, there are references to the Antikythera device as an analogue computer. Is this correct?

It depends on how we understand the term ‘analogue’. One sense of analogue is ‘to bear analogy to’. The Antikythera device is built in such a way as to be analogous to operations of the cosmos, as understood by its builders. So in this sense it is an analogue device.

Another interpretation of an analogue computer is one that works by means of continuous rather than discrete quantities. Again, in this sense, the Antikythera device is analogue, since the dials on the front give continuous readings of lunar and solar angles.

But if we look at it from a different aspect, the device is digital; that is, it performs mathematics in discrete operations. Look back at Fig. 3.3. The ratio that this pair of gearwheels computes is always going to be a rational number, a ratio of the integral numbers of teeth on the two wheels. The multiplications performed by the machine were thus digital.

It is also arguable that its mode of operation was semi-digital, in that it seems to have been advanced by one ‘clock cycle’ per day. Each day, the handle on its main driving wheel was rotated once, and the date pointer on the face would move on by an amount equal to one whole day. Because the internal multiplications were by rational numbers, the machine would not have drifted over time. Having completed one Metonic cycle, it would have gone on to compute the next with equal accuracy. In so far as the machine drifted with respect to the actual motions of the moon, it would be because the gearing ratios were not specified to a sufficient number of digits. This drift, of the calculated

position of the moon from its observed position over multiple cycles, is characteristic of digital calculations. Digital calculations must always express ratios in terms of whole numbers, and to get more and more accuracy when simulating some natural process, we are forced to use larger and larger numbers in the numerator and denominator.

In modern terms, the Antikythera device did digital arithmetic, but used analogue input and output devices. The analogue scales used for the answer, although in principle continuous, would in practice usually have been read off as discrete numbers of days. Had we wished for more precision, an additional dial, which rotated faster, could have been used. This is the principle used in clocks. A clock has a solar pointer (the hour hand) that completes two rotations for every apparent rotation of the sun. To enable us to be able to tell the time to greater accuracy, we then have minute and second hands that rotate 12 times and 720 times faster. By appropriate multiplication of scales, a cog-wheel computational mechanism can add least significant digits to its output. Clocking down the main output, can add leading digits to the output: Antikythera does this with its Callipic cycle dial.

3.1.6 Limitations

The techniques used in the Antikythera device were limited to the performance of the operations of angular addition and subtraction, along with multiplication by negative constants. Final readout accuracy was limited by the mechanical precision of the dials. Arithmetic is modular.

3.2 Late mechanical computers

The main use of computers in the first half of the twentieth century was for warfare, particularly for naval warfare. It was here that the greatest computational demands lay. The development of long-range guns and steam power meant that by the start of the twentieth century it was no longer possible for navies to simply aim their guns using open or telescopic sights. The ranges at which successful firing between warships occurred increased from 7 km at the Battle of Tsushima in 1905 to 25 km at the Battle of Calabria in 1940. By 1914, the speeds of ships had increased so much that their relative velocities could exceed 100 km h^{-1} .

At ranges above a few kilometres, shot from naval guns came down in a plunging fashion. To hit another ship, it was not sufficient to aim directly at the ship; gunners had to know the range of the target ship and aim their guns high so that the shot would fall on to the target. This problem could be broken down into a number of sub-problems.

1. The bearing of the target had to be found.
2. The range had to be estimated. Initially, this was done using stereo matching (see Fig. 3.7(c)). Later, radar was used to get a more accurate estimate.

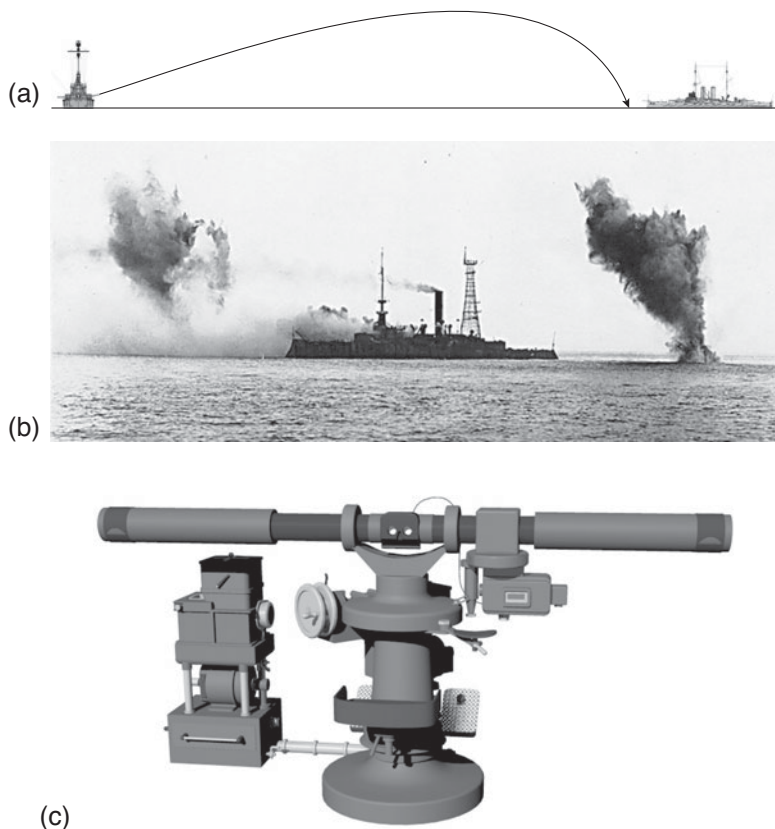


Fig. 3.7 Gunners had to have an accurate estimate of range to prevent shot from falling short or over. (a) A schematic representation of the falling of a shell. (b) A salvo straddling *USS Iowa* in 1923 during ranging practice. (c) A model by R. Brassington of the Argo Pollen gyro-stabilized stereoscopic rangefinder used by the Royal Navy.

3. The relative velocity of the two ships then had to be estimated. The initial shell velocity might be around 800 m s^{-1} . A shell might be in flight for 20 or 30 seconds. A ship moving with a velocity of 60 km h^{-1} relative to the firing ship could have moved some 500 m during the time of flight.
4. The range and bearing of the target at the end of flight had to be estimated.
5. The up-to-date range and bearing had to be translated into an appropriate elevation and deflection of the gun before firing.

All these calculations had to be done in real time, with the time between shots being of the order of 30 seconds. Early in the twentieth century, it was realized that it was quite impractical to do the calculations by hand, both because that would have been beyond the trigonometrical ability of most gunnery officers, and because it would have been far too slow. The well-funded admiralties of the day could command the attention of skilled engineers, scientists, and instrument makers, so it was not long before mechanical solutions were devised.

The techniques that were arrived at, 2000 years after the Antikythera device, show a strong family resemblance to that earlier real-time calculator. They have a similar mode of display, based on dials; similar

construction, using: brass frames and gearing; and many of the same mechanical implementations of arithmetic techniques. We will focus on a couple of the devices invented for the purpose; for a more comprehensive account, readers should consult Brooks (2005).

3.2.1 Estimating rates of change

A moving ship has a course bearing and a speed. Together, these can be represented as a vector. In modern computing, we tend to think of vectors as a list of numbers, so a ship steering a course due south at 10 metres per second could be described by the pair $[0, -10]$, these being the components of its velocity in a Cartesian coordinate system whose x -axis was aligned with the equator. A ship sailing west at the same speed would have a velocity vector $[-10, 0]$; and a ship sailing north-east a velocity vector $[7.07, 7.07]$, since it would be moving at just over 7 metres per second north and a similar amount east. Now let us look at the concrete example in Fig. 3.8. Ship A is moving with velocity vector $[0, -14]$ and ship B is moving with a velocity vector $[15, -3]$. We want to know the combined velocity of B relative to A. To do this, we subtract the velocity of A from that of B: $[15, 3] - [0, -14] = [15, 11]$.

This gives us the velocity of B relative to A. In order to aim a gun, we need to break this velocity down into two components, one along the line of sight and the other perpendicular to the line. If we were doing that nowadays, we would use multiplication by a rotation matrix to get this effect. Back in the early 1900s, this operation would have involved a prohibitive amount of equipment, but a naval officer by the name of Lieutenant Dumaresq (1905) came up with a device (Figures 3.9 and 3.10) that—although, at first sight, a bit baffling—actually uses some beautifully simple principles.

Look at Fig. 3.9(a). It shows the first principle: vector addition by direct geometric implementation. The Dumaresq consisted of a circular frame graduated on its upper edge from 0° to 180° , Red and Green relative to the fore and aft Bar that is mounted on the frame. The frame itself was fixed, with this bar parallel to the fore and aft line of the ship.

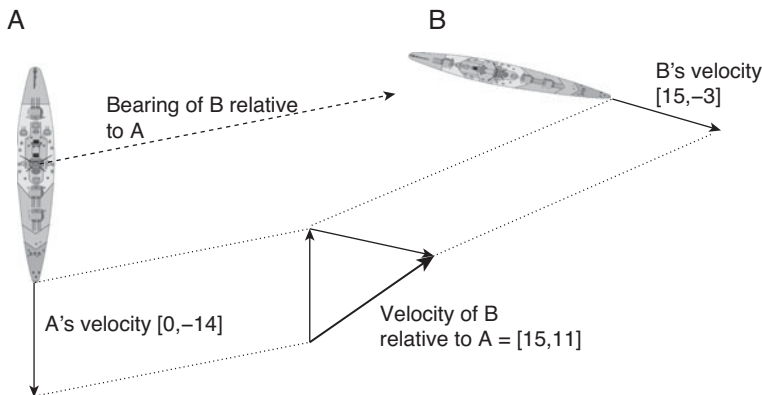


Fig. 3.8 Vector addition of the relative motion of two ships.

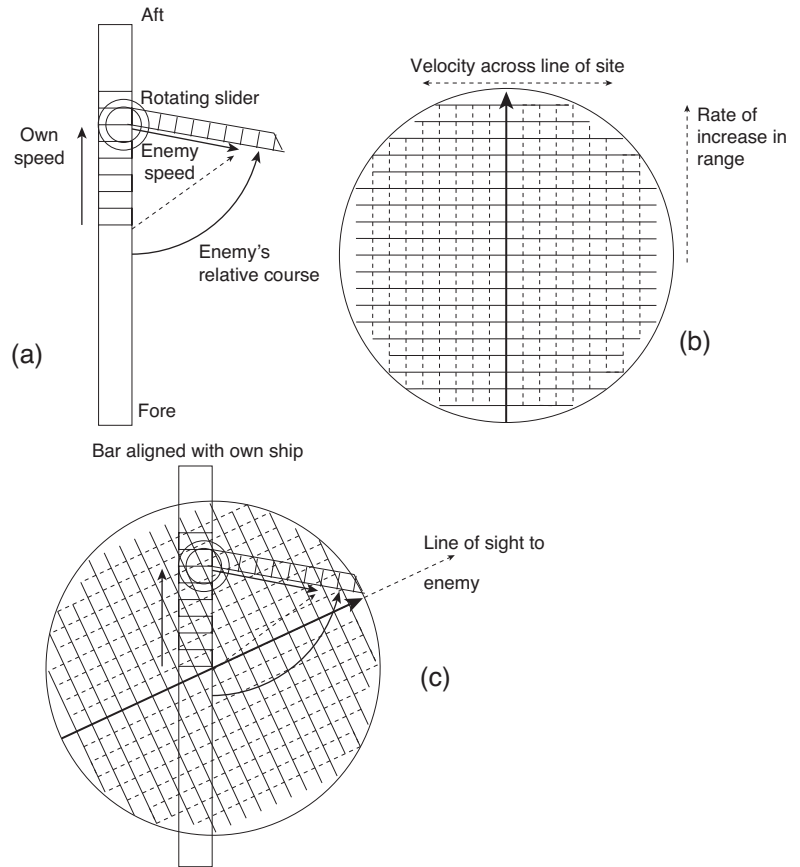


Fig. 3.9 The operating principle of the Dumaresq.

Under this bar, a slider was mounted that travelled along the bar, carrying an index cursor that could be moved aft from the centre along a scale graduated from 0 to 35 knots. This scale measured the ship's own speed. On the slider was a rotating ruler, also marked in a scale of knots. The ruler had to be aligned with the enemy course relative to your own ship's course; that is, the angle between the bar and the ruler had to be the same as the angle between your course and the enemy course. Since the ruler is scaled in knots, a line from the Dumaresq centre to the point on the ruler corresponding to the enemy speed will be the enemy's relative velocity vector.

The Dumaresq modelled geometry with geometry and avoided the detour into pairs of numbers that we would do on a modern computer.

There remained the problem of how to transform a relative velocity vector in the frame of reference of the ship into one in the frame of reference of the gun. We said that the modern technique would be to multiply the vector by a rotation matrix. The 1905 solution was to use a rotating disc on which was marked a system of Cartesian coordinates giving the velocity parallel with and normal to the line of sight. The disc was rotated until the arrow on it was aligned with the target, after

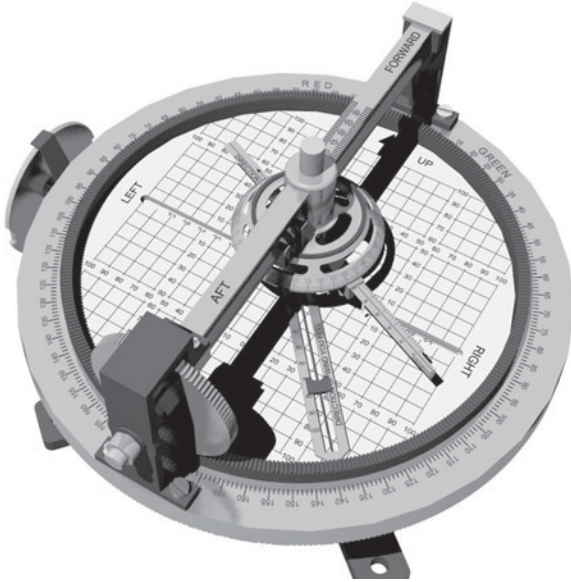


Fig. 3.10 A Dumaresq analogue mechanical computer used in naval gunnery computations. The one illustrated is a Wind Dumaresq, which in addition to giving corrections for relative motion of the ships also provides an additional compensation term due to the wind. Image from a model by R. Brassington, with his permission.

which the range rate and deflection normal to the line of sight could be read off the scale. The task of the modern ‘rotation matrix’ was achieved using a matrix of lines that physically rotated.

3.3 Analogue mechanical multiply/accumulate

We now come to another new technique used in the gunnery computing devices, an integrator. Up to now, we have discussed mechanical multiplication by a constant, achieved via gearing ratios. If we consider the problem faced by those dealing with ranging, we can see that this would not have been enough.

Range estimates via the stereo rangefinders came through intermittently, and the gunners had to have intermediate estimates of range. Suppose that we know the range r_0 at time t_0 and, using the Dumaresq, we have an estimate r' of the rate at which the enemy ship is approaching along the line of sight. We need to calculate

$$r_t = r_0 + \int_0^t r'$$

A mechanical integrator had been developed by William Thomson, later Lord Kelvin, and his brother James Thomson in 1876 (Thomson, 1876 a,b)—an illustration of his design is shown in Fig. 3.11. His one was built into a more complex apparatus that was able to extract the Fourier components of a signal. We will describe the simpler version used in the Vickers Range Clock (Dawson, 1909). Look at Fig. 3.12. We have a flat



Fig. 3.11 The original integrator developed by Thomson.

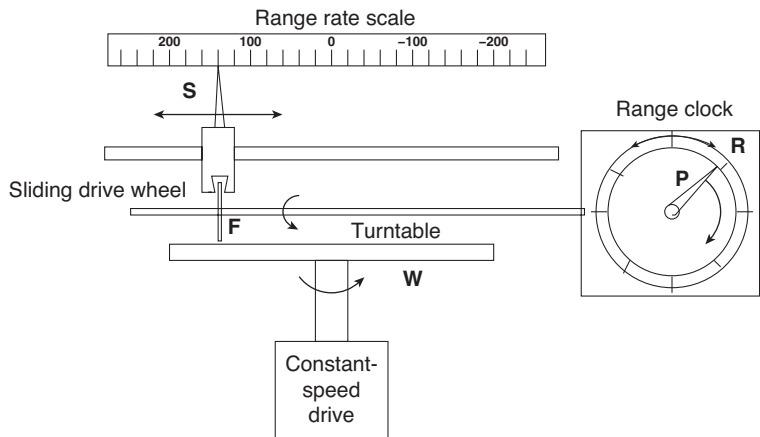


Fig. 3.12 A mechanical integrator as used in range clocks and later in differential analysers.

horizontal wheel or turntable, driven by a clock such that it rotates once every minute, and such that we have a slider **S** that we can use to record r' in, say, 100 metres per minute. The slider carries a friction roller **F** that bears on the surface of **W** and is driven round by it. If we move the slider, then the friction roller moves with it. As it is moved closer to the centre of the turntable, the rotational velocity of **F** will decrease, as the relative gearing rate between **F** and **W** alters. If **F** is moved past the axis of **W** to the other side, it will rotate in the reverse direction. The effect is to provide a variable multiplication of the rotation of the time clock by the range rate. This was then transferred to the range clock dial, which moved at a rate corresponding to the approach or recession of the enemy.

Whenever an actual range observation occurred, the rotating dial of the range clock **R** was moved round in order to bring the observed range into alignment with the range clock pointer **P**. The range clock would then point to the current range and would give estimates of the likely range as it changed.

The representation in Fig. 3.12 is very schematic: actual integrators had a more compact arrangement, often with the range clock dial above and coaxial with the turntable.

The basic technology of Dumaresq and range clock had been developed by 1906. In the next decade, the pressure of the arms race meant that mechanical computer technology advanced rapidly. Whereas originally, the transfers of information between Dumaresque and range clock were manual, they became increasingly automated, with information being transmitted by what amounted to asynchronous digital serial transmission between different parts of the ship. By the middle of the First World War (1914–18), highly integrated machines such as the Dreyer Table and the Argo Plotter Clock were in use (see Fig. 3.13). The naval integrators derived from an earlier one built in 1876 by James Thomson, brother of Lord Kelvin, who acted as a consultant to Arthur Pollen in the development of the Argo clock. Thompson's original mechanism had been used by Kelvin to analyse tidal motions. The integrator mechanism used in the range clock was later applied by Hartree (1938) to more general scientific calculations during the 1930s.

The machines developed during the First World War provided the basis for a later generation of mechanical computers used by leading navies during the Second World War (1939–45). Dumaresqs remained in use on small ships, but vessels above the size of a destroyer were fitted with machines descended from the Argo clock. These typically combined mechanical computations with digital electrical input and output. These mechanical gunnery computers (see Fig. 3.14) were last used in US Navy battleships in 1991, during the first Gulf War.

3.3.1 Limitations

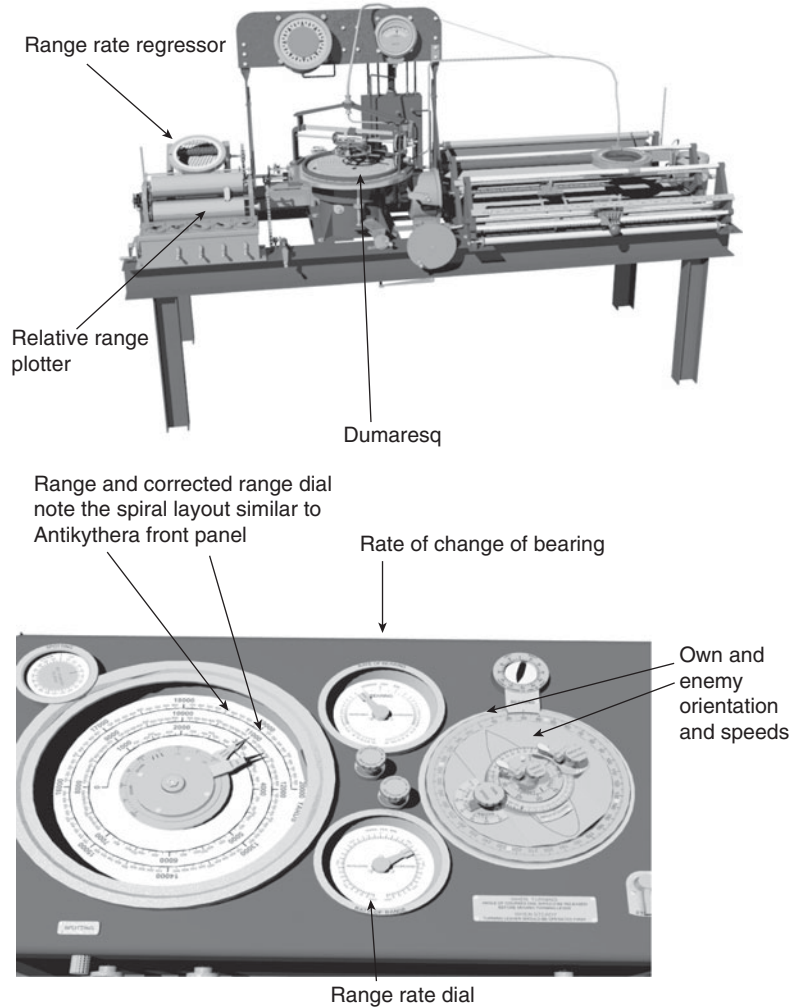
These were all essentially mechanical analogue machines and were limited in that:

- Their accuracy would be limited to around three decimal digits of precision. Given that the margins of error of the original input data were much worse than that, the limited precision of computation was not significant.
- They were dedicated to performing a single suite of tasks.

Their most significant advance was the ability to multiply variables and to integrate.

In the rest of this chapter, we will consider how the limitations in accuracy were addressed by the developers of mechanical computers. In Chapter 4, we will look at how the concept of a general-purpose machine was arrived at.

Fig. 3.13 Mechanical computers used during the First World War. The upper photo shows the Dreyer Table. In the middle is an automatic Dumaresq, which controls a range integrator whose turntable is built into the circular support of the Dumaresq. On either side are graph plotters that produce traces of the relative courses of the enemy ship. Scatter plots on these record the fall of shot, and a regression of these could be estimated using a rotating disc with scribed parallel lines. The lower photo shows the control panel of the Argo Pollen range corrector. Both images are renders of digital models by R. Brassington.



3.4 Mechanizing the abacus

There is a line of development that stretches from the Antikythera device, via clocks, down to the fire control computers of the twentieth century. It is family of computing machines that was developed to meet the needs of astronomy and physics. There was a distinct line of development that took off from the abacus and was initially dedicated to commercial calculations. An abacus performs exact integer arithmetic and can, by adding more rows to the device, be built to calculate to any desired number of significant figures. Such precision was essential to commerce. Suppose you were a seventeenth-century banker, lending 100 000 Thaller at an interest rate of 2.5%. You would expect to get back 2500 Thaller. If you did calculations using a machine accurate to only

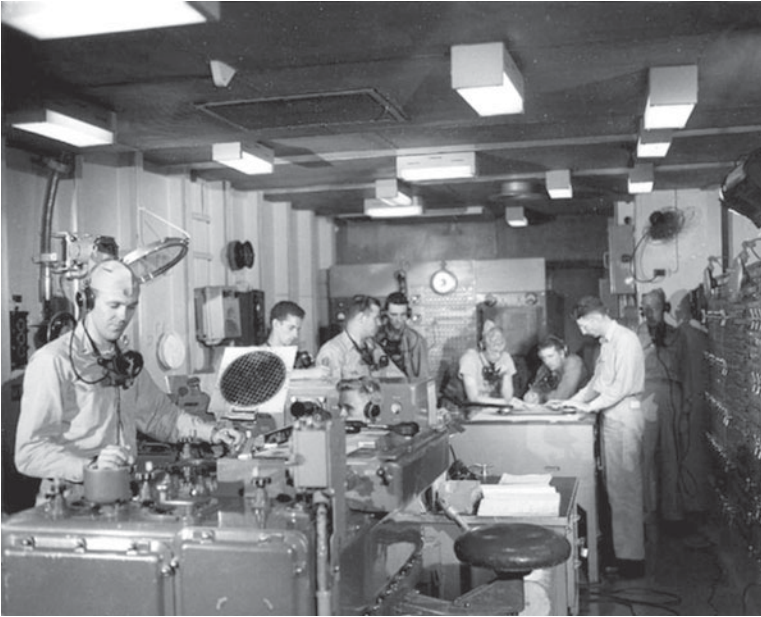


Fig. 3.14 The main computer room on a US warship during the 1950s. These designs of mechanical computers remained in use until the early 1990s. Wikimedia image.

three significant figures, then all it would tell you is that you ought to charge between 2000 and 3000 Thaller, which makes a huge difference to your profit.

A chain of gears linked in successive 1:10 ratios will allow numbers to be recorded accurately to a high degree of precision, but they only allow addition using the least significant digit wheel. If you want to repeatedly add numbers less than ten to a running total, that is fine. But suppose you have a sequence of six such geared wheels and you want to add 20 000 to the total. You try to move wheel 5 on by two positions. What happens?

You find that the wheel is stuck, since in order for wheel 5 to move round by one-tenth of a rotation, the least significant wheel must rotate 1000 times. Friction will make this impossible. A means of overcoming this was invented by Schickard, who in 1623 designed the first digital mechanical calculator, using a gearing mechanism based on the principle shown in Fig. 3.15. This allowed a carry to take place between successive digit wheels such that as the units wheel made one rotation, it would advance the tens wheel by one place. On the other hand, a rotation of the tens wheel by one place would, in general, not move the units wheel, since they would be decoupled. In 1645, Pascal developed an alternative carry mechanism that involved the units wheel raising a weighted arm, which then fell on the tens wheel when the transition from '9' to '0' occurred. This further reduced the mechanical force required to propagate a carry (Pratt, 1987; Kistermann, 1998).

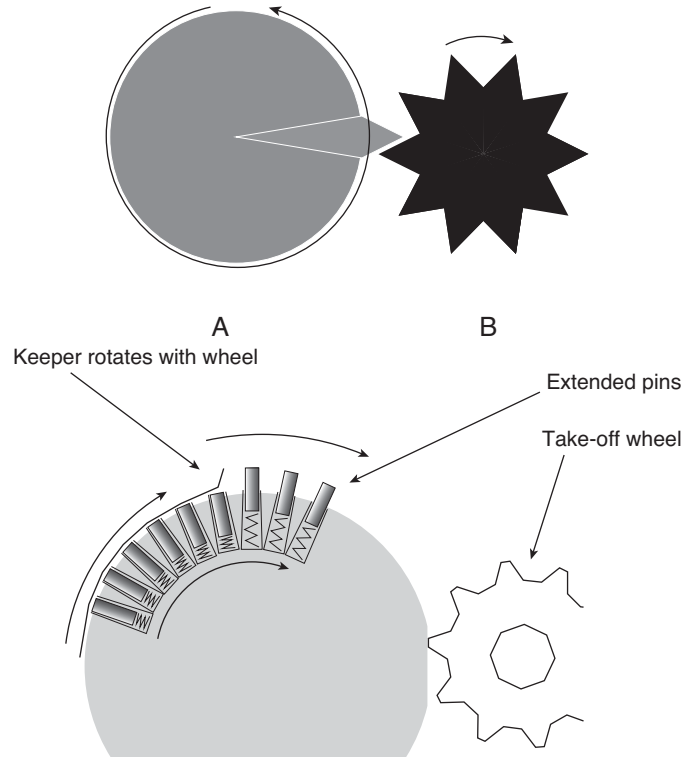


Fig. 3.15 Top: the use of a single toothed wheel as a carry mechanism—one rotation of A generates one-tenth of a rotation of B. This was later generalized by Odhner, who in 1874 invented the pinwheel. Bottom: the use of a pinwheel to add a variable amount to the take-off wheel.

3.4.1 Limitations

These early digital calculators were limited to performing addition. If you wanted to perform subtraction, you had to use complement arithmetic. Suppose that you have a three-digit Schickard or Pascal type machine and want to subtract 7 from 208. You first subtract 7 from 1000 in your head to give 993, and then do the sum $208 + 993 = 1201$ on the machine. But since the machine is only three digit, the thousands digit does not appear and you get the answer 201.

3.4.2 Digital multiplication

Multiplication on an abacus or reckoning table as described in section 2.4 required the user to memorize multiplication tables. It also involved a nested loop algorithm. Given four reckoning table registers, A, B, C, and D, with A and B containing the numbers to be multiplied, D the total, and C a temporary register. Initially, registers D and C contain no beads. The user proceeded as follows:

```

for i = 1 to number of digits in A do
  for j = 1 to number of digits in B do
    in your head set temp = A[i] * B[j] this yields a
    2 digit number
    place this number in positions C[i+j] and C[i+j-1]
  
```

```
Add the contents of register C to register D using
standard method
```

If we wanted to do this mechanically, a major objective would be to obviate the need to memorize multiplications. An algorithm involving no mental arithmetic would involve using repeated addition to obtain the effect of multiplication:

```
for i = 1 to number of digits in A do
  for j = 1 to number of digits in B do
    for k= 1 to A[i] do
      C[i+j-1] = B[j]
      Add the contents of register C to register D using
      standard method
```

This could have been done on a Pascal machine, but it now involves three nested loops. We can reduce this to two loops if we have a more sophisticated addition mechanism. Let n be the number of digits in B :

```
for i = 1 to number of digits in A do
  for k= 1 to A[i] do
    C[i..i+n-1] = B[1..n]
    Add the contents of register C to register D using
    standard method
```

The key step here is to transfer the whole of B into an appropriately shifted position in C and add it to D in a single step. It was Leibniz' achievement to see a way of building a machine to implement this algorithm (Pratt, 1987). A late machine based on his principle is shown in Fig 3.16. Assuming that all numbers are held as gear positions, he had to make two innovations:

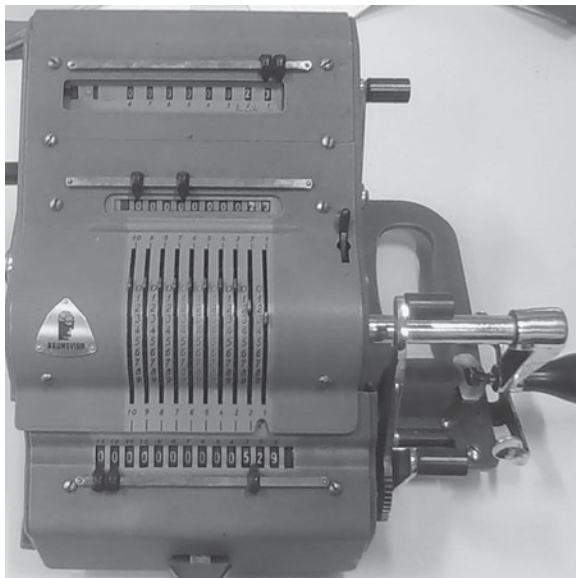


Fig. 3.16 A Brunsviga Model 13-pinwheel calculator, showing the result of multiplying 23 by 23 using the algorithm of Leibniz. The register that we have termed D is at the bottom and can slide sideways with respect to the barrel of pinwheels in the centre. The barrel has 23 set on it. The barrel is the register B in our description of the Leibniz algorithm. A rotation of the large handle on the right will add the contents of the barrel to the appropriately shifted register D . The register at the top, A in our description, holds a count of the number of additions performed in each position. A reverse rotation of the handle performs subtraction. Produced 1952–64.

1. The sliding carriage allowed number B to be placed in a mechanical register that could be moved left and right with respect to D.
2. The stepped gearwheel allowed number B to be held in a form that allowed a single rotation of register B to add register B to register D. The stepped wheel solved the problem of having a variable number of cogs by using a wheel in which the number of cogs varied along its length. A normal gear wheel moved along the stepped one's length would engage a different number of cogs depending on its position.

An improvement on the stepped wheel was the invention of the pinwheel shown in Fig. 3.15, which allowed a more compact device.

Variable-toothed wheels of different designs along with sliding carriages were incorporated in mass-produced commercial calculators from the nineteenth century, until they were replaced by electronic ones in the 1970s. Figure 3.16 shows a widely used model of pinwheel calculator that remained in production until the 1960s.

Logical limits to computing

4

4.1 Introduction

In the previous chapters, we have explored the origins and physical properties of real computing devices. In contrast, we are now going to stand back and look at the origins and abstract properties of ideas of *computation*. Here, we will focus strongly on *mathematical logic* (Kneebone, 1963), which seeks to characterize and formalize logical reasoning, and has strong influences both on how we construct and program computers (Computer Science), and on our attempts to construct machines that emulate human reasoning (Artificial Intelligence).

Within mathematical logic lies *meta-mathematics* (Kleene, 1952), a curious discipline concerned with how mathematics can be used to explain mathematics itself.¹ As we shall see, such seemingly arcane endeavours uncover very practical limits to what we can expect of computers.

In the rest of this chapter, we will survey the relationships between theories of computability and meta-mathematics, drawing on foundational work stretching back over 2000 years. We will first look at the origins of contemporary ideas of computability in propositional and predicate logic, and draw out the connections and distinctions between execution, evaluation, and proof. Next, we will explore how paradoxes arise in systems that are able to describe themselves. We will then consider Peano arithmetic, and how it can be used to encode potential mathematical theorems and proofs, and thus engender deep self-referential paradoxes about mathematics. Next, we will introduce notions of infinity and discuss Cantor's characterization of countable and uncountable quantities. Finally, we will meet both Turing Machines and Church's λ calculus, explore fundamental undecidability results that bound computability, and elucidate the difference between mathematical systems and embodied machines.

4.2 Propositional logic

Logic is concerned with correct reasoning (see Fig. 4.1; see also Swinbourne, 1875).

| | | |
|------|--|----|
| 4.1 | Introduction | 47 |
| 4.2 | Propositional logic | 47 |
| 4.3 | Set theory | 51 |
| 4.4 | Predicate logic | 52 |
| 4.5 | Recursion | 55 |
| 4.6 | Peano arithmetic | 55 |
| 4.7 | Paradoxes | 58 |
| 4.8 | Arithmetizing mathematics and incompleteness | 61 |
| 4.9 | Infinites | 64 |
| 4.10 | Real numbers and Cantor diagonalization | 66 |
| 4.11 | Turing Machines | 67 |
| 4.12 | Universal TM and undecidability | 71 |
| 4.13 | Computational procedures | 74 |
| 4.14 | The Church–Turing thesis | 78 |
| 4.15 | Machines, programs, and expressions | 79 |

¹Here, the Greek prefix *meta* means 'over', as in 'meta-physics', the philosophy about the nature of physical things, as opposed to the science of physics.

Fig. 4.1 Destrawney masters the Man-eater Logic.



| | |
|-----------------------|----------------------|
| true false | truth values |
| <i>var</i> | variable |
| $\neg exp$ | negation |
| $exp \wedge exp$ | conjunction |
| $exp \vee exp$ | disjunction |
| $exp \Rightarrow exp$ | implication |
| (<i>exp</i>) | bracketed expression |

Fig. 4.2 Propositional logic.

Propositional logic (Nidditch, 1962) is a very simple system for looking at arguments built from statements about things that can be either *true* or *false*. Such statements, termed *expressions*, are made up of these truth values themselves, variables that can stand for arbitrary truth values, and operations on and between expressions such as \neg (NOT) and \wedge (AND) and \vee (OR) and \Rightarrow (IMPLIES). Figure 4.2 shows a syntax for propositional logic.

We can then use this logical language to turn simple natural-language arguments into logical expressions. For example, consider:

If Chris eats bananas then Pat doesn't have a beard. Pat has a beard. So Chris doesn't eat bananas.

If we introduce variables to stand for the basic assertions:

Chris eats bananas $\rightarrow C$

Pat has a beard $\rightarrow P$

then we can write the argument as follows:

$$((C \Rightarrow \neg P) \wedge P) \Rightarrow \neg C$$

It is important to note that even though this argument is patently nonsensical, nonetheless we can cast it in logic. That is, propositional logic is not concerned with the real-world meaningfulness of an argument, but with its internal coherence. It is also important to note that we no longer view an assertion such as ‘Chris eats bananas’ as implicitly true, but view it tentatively as something that may be either true or false and so represented by a variable. Thus, we have constructed a general argument that is quite independent of bananas and beards.

4.2.1 Truth tables

Now we can ask whether this is a valid argument. That is, if we assume that the first two sentences are true, then can we establish that the third sentence is necessarily true? Before we can do this, we first need to give our logical expressions some semantics. We do so in Fig. 4.3, using *truth tables* to show all the values of the operations for all possible values of their operands.

Thus:

- \neg flips true to false and false to true
- \wedge is only true if both operands are true
- \vee is true if either or both operands are truth
- \Rightarrow is true if either both operands are true or the first operand is false

Implication (\Rightarrow) deserves a wee bit more explanation. Given our commonsense understanding of ‘*premise* \Rightarrow *conclusion*’, it may seem strange that an implication should be true if the premise is false. The point is that even though the implication is true, the *conclusion* may still be true or false: \Rightarrow is an operation, not a link in a chain of reasoning. Or to put it another way, if the *premise* is false, then it doesn’t matter if the *conclusion* is true or false.

| | | | | | |
|-------|----------|------------|-------|-------|-------------------|
| X | $\neg X$ | | X | Y | $X \wedge Y$ |
| false | true | | false | false | false |
| true | false | | false | true | false |
| | | | true | false | false |
| | | | true | true | true |
| X | Y | $X \vee Y$ | X | Y | $X \Rightarrow Y$ |
| false | false | false | false | false | true |
| false | true | true | false | true | true |
| true | false | true | true | false | false |
| true | true | true | true | true | true |

Fig. 4.3 Truth tables for operations.

Now, we can check the validity of an argument by constructing a truth table systematically from each of its component expressions. Figure 4.4 shows the truth table for our example. So, we can see that the argument is true in all cases.

An expression that is true for all possible assignments of truth values to variables is called a *theorem*. As we shall see, evaluating a logical expression for all possible truth values has strong analogies with executing a computer program for all possible inputs.

Now, truth tables grow very quickly. They require 2^N rows for N variables and a column for each distinct sub-expression. So even moderately sized arguments have very large truth tables that are complex to understand, and time-consuming to construct and check. If we generate a truth table row by row, we may quickly find a case where the argument is false. But we might equally well need to try all the rows only to discover that the last one is false.

4.2.2 Deductive proof

An alternative to using truth tables to tell whether or not a argument is valid is to try to *prove it correct*; that is, to use sound reasoning about its structure without considering individual cases. To do this, we need a formal system consisting of basic *axioms* that are demonstrably always true, and *rules of inference* that demonstrably produce new theorems from axioms and old theorems. Then we try to construct the expression we're interested in, starting with axioms and applying rules.

There are many formal systems for propositional logic. One of the simplest, by Lukasiewicz, requires three axioms and one rule of inference, as shown in Fig. 4.5. It is straightforward to show that the axioms hold by constructing a truth table.

The *Modus Ponens (MP)* rule says that if some premise (X) holds and if some conclusion (Y) follows from it, then that conclusion holds.

We will also use the simplification

$$\neg\neg X = X$$

to cancel out repeated negations.

| | | | |
|------------------------|-----------------------------------|--|----------|
| C | P | $\neg C$ | $\neg P$ |
| false | false | true | true |
| false | true | true | false |
| true | false | false | true |
| true | true | false | false |
| $C \Rightarrow \neg P$ | $(C \Rightarrow \neg P) \wedge P$ | $((C \Rightarrow \neg P) \wedge P) \Rightarrow \neg C$ | |
| true | false | true | |
| true | true | true | |
| true | false | true | |
| false | false | true | |

Fig. 4.4 The truth table for an argument.

| | |
|---|-------------------|
| $X \Rightarrow (Y \Rightarrow X)$ | AX1 |
| $((X \Rightarrow Y) \Rightarrow Z) \Rightarrow ((X \Rightarrow Y) \Rightarrow (X \Rightarrow Z))$ | AX2 |
| $(X \Rightarrow Y) \Rightarrow (\neg Y \Rightarrow \neg X)$ | AX3 |
| given X and $X \Rightarrow Y$, conclude Y | Modus Ponens (MP) |
| $(X, Y$ and Z stand for any expressions) | |

Fig. 4.5 Axioms and the rule of inference for propositional logic.

So, to prove our theorem, with the assumptions

$$\begin{array}{ll} C \Rightarrow \neg P & \text{AS1} \\ P & \text{AS2} \end{array}$$

we proceed as follows:

1. $(C \Rightarrow \neg P) \Rightarrow (\neg\neg P \Rightarrow \neg C)$ AX3
2. $\neg\neg P \Rightarrow \neg C$ AS1, 1., and MP
3. $P \Rightarrow \neg C$ 2. and simplification
4. $\neg C$ AS2, 3., and MP

The ‘magic’ trick here is to be able to construct the right initial expression(s) from which to reason via the assumptions to the conclusion. It is, of course, debatable whether it is in any sense easier to construct a proof or a truth table.

4.3 Set theory

Propositional logic is restricted to reasoning about simple truthful propositions. In order to be able to reason about more general entities than truth values, we need some formalism for characterizing and manipulating them, both individually and severally. One of the simplest such systems is *set theory* (Halmos, 1960), which is concerned with properties of collections of things.

At their simplest, sets are made up of unitary *elements* with distinct symbolic representations. Sets are usually written as comma-separated sequences of elements within the {and} brackets:

$$\{element_1, element_2, \dots, element_N\}$$

For example, we might have:

$$\{\text{papaya, peach, pear, persimmon, pineapple, plantain, plum}\}$$

Note that each unique element may only occur in any set once.

The empty set is written {}.

Note that the order of elements is not important, so two sets are *equal* if they have the same elements:

$$\{\text{apple, banana, cherry}\} = \{\text{cherry, apple, banana}\} \rightarrow \text{true}$$

It may be tested whether some value is a *member* of a set:

$$\text{banana} \in \{\text{apple, banana, cherry}\} \rightarrow \text{true}$$

or if some set is a *subset* of another; that is, whether every member of the first is a member of the second:

$$\{\text{apple, cherry}\} \subset \{\text{banana, cherry, date}\} \rightarrow \text{false}$$

Two sets may be joined together through *union*:

$$\{\text{banana, apple, cherry}\} \cup \{\text{elderberry, cherry, date}\} \rightarrow \\ \{\text{banana, apple, cherry, elderberry, date}\}$$

Note that **cherry** only appears once in the resultant set.

The elements common to two sets may be found through *intersection*:

$$\{\text{apple, banana, cherry}\} \cap \{\text{cherry, date, elderberry}\} \rightarrow \{\text{cherry}\}$$

The elements of one set may be removed from another through *difference*:

$$\{\text{apple, banana, cherry, date}\} \setminus \{\text{date, banana}\} \rightarrow \{\text{apple, cherry}\}$$

As with propositional logic, there are also axiomatic systems for set theory. Cantor is attributed with developing the first. Contemporary set theories are developments of the more recent systems by Zermelo and Fraenkel. We will not consider these further here, but use set notation as a useful tool in subsequent sections.

4.4 Predicate logic

In the above discussion of propositional logic, we treated statements such as ‘Chris eats bananas’ and ‘Pat has a beard’ as unitary entities that can simply be true or false. In *predicate logic* (Hodges, 1977), we may further decompose such statements to separate out *predicates* that may qualify appropriate *values*. For example, we might use a truth table to identify the values *Chris* and *Pat*, which are both instances of *person*, and the predicates *eatsBananas* and *hasBeard*, which apply to a *person*, to return truth values:

| P | eatsBanana(P) | hasBeard(P) |
|-------|---------------|-------------|
| Chris | true | false |
| Pat | false | true |

Alternatively, we could identify *bananas* and *beard* as instances of *things*, and the binary *relations* *eats* and *has*, which both apply to a *person* and a *thing* to return a truth value:

| P | T | eats(P,T) | has(P,T) |
|-------|--------|-----------|----------|
| Chris | banana | true | false |
| Chris | beard | false | false |
| Pat | banana | false | false |
| Pat | beard | false | true |

We could formulate *person* and *thing* more precisely as *sets* by listing their members:

```
person = {Chris, Pat}
thing = {bananas, beard}
```

We can also formulate the predicates as sets:

```
eatsBananas = {Chris}
hasBeard = {Pat}
```

Now we can test whether a predicate holds for a value by checking if the value is a member of the set for the predicate:

```
hasBeard(Chris) == Chris ∈ hasBeard → false
```

For the binary relations *eats* and *has*, we can use sets of bracketed *pairs* to associate the operands:

```
eats = {(Chris, bananas)}
has = {(Pat, beard)}
```

and test if a relation holds between two values by checking if the corresponding pair is a member of the set for the relation:

```
eats(Chris, bananas) == (Chris, bananas) ∈ eats → true
```

Note that we may use sets to represent both values and properties of values.

Predicate logic enables us to ask questions as to whether or not some predicate holds for *all* members of a set (*universal quantification*):

$$\forall var \in set : predicate(var)$$

or whether there *exists* some member of a set for which some predicate holds (*existential quantification*):

$$\exists var \in set : predicate(var)$$

To explore this, let's consider a marginally richer world:

```
person = {Chris, Jo, Les, Pat}
likesFruit = {Chris, Jo, Pat}
fruit = {peach, pear, plum}
likes = {(Chris, peach), (Chris, plum)(Jo, pear),
        (Pat, peach), (Pat, plum)}
```

So we might ask if someone doesn't like fruit:

$$\exists p \in \text{person} : \neg \text{likesFruit}(p)$$

or if every fruit is liked by someone:

$$\forall f \in \text{fruit} : (\exists p \in \text{people} : \text{likes}(p, f))$$

Universal quantification is equivalent to a sequence of conjunctions:

$$\forall var \in \{e_1, e_2, \dots\} : \text{predicate}(var) == \\ \text{predicate}(e_1) \wedge \text{predicate}(e_2) \wedge \dots$$

and existential quantification to a sequence of disjunctions:

$$\exists var \in \{e_1, e_2, \dots\} : \text{predicate}(var) == \\ \text{predicate}(e_1) \vee \text{predicate}(e_2) \vee \dots$$

So, we can test the truth or falsity of a quantified expression by constructing a truth table and systematically evaluating the expanded form. For example, for 'someone doesn't like fruit':

| P | likesFruit(P) | \neg likesFruit(P) |
|-----------|---------------|----------------------|
| Chris | true | false |
| Jo | true | false |
| Les | false | true |
| Pat | true | false |
| \exists | | true |

we check `likesFruit` for each `person` and then \vee the results together. And for 'every fruit is liked by someone':

| P | F | likes(P,F) |
|-------------------|-------|------------|
| Chris | peach | true |
| Jo | peach | false |
| Les | peach | false |
| Pat | peach | true |
| \exists | | true |
| Chris | pear | false |
| Jo | pear | true |
| Les | pear | false |
| Pat | pear | true |
| \exists | | true |
| Chris | plum | true |
| Jo | plum | false |
| Les | plum | false |
| Pat | plum | true |
| \exists | | true |
| $\forall \exists$ | | true |

we test for \exists for each `fruit` in turn and then \wedge the results together.

As with propositional truth tables, predicate truth tables grow very quickly with the number of variables and the sizes of the associated sets. An alternative is to extend propositional logic with new axioms and rules of inference for predicate logic, to enable proof. We will not explore doing so here.

4.5 Recursion

Recursion (Peta, 1967) is a central technique in meta-mathematics and involves defining something in terms of properties of itself. Typically, recursion is used to explore properties of a collection of things like a set, by inspecting each item in the collection in turn until there are none left. Thus, recursive definitions usually have two components:

- the *base case*, where the collection is empty so some final value is returned
- the *recursion case*, where the collection is not empty so the property is checked for one item, and then the rest of the collection is checked

We can define the universal and existential quantifiers by recursion. \forall is equivalent to applying the predicate to the first member and \wedge ing the result to that from checking the rest of the set. And, in the base case, any predicate is true for all of an empty set:

$$\begin{aligned} \forall var \in \{e_1, e_2, \dots\} : predicate(var) &== \\ predicate(e_1) \wedge predicate(e_2) \wedge \dots &== \\ predicate(e_1) \wedge \forall var \in \{e_2, \dots\} : predicate(var) & \end{aligned}$$

$$\forall var \in \{\} : predicate(var) == \text{true}$$

Similarly, \exists is equivalent to applying the predicate to the first member and \vee ing the result to that from checking the rest of the set. And an empty set has no members for which a predicate can be true:

$$\begin{aligned} \exists var \in \{e_1, e_2, \dots\} : predicate(var) &== \\ predicate(e_1) \vee predicate(e_2) \vee \dots &== \\ predicate(e_1) \vee \exists var \in \{e_2, \dots\} : predicate(var) & \end{aligned}$$

$$\exists var \in \{\} : predicate(var) == \text{false}$$

4.6 Peano arithmetic

4.6.1 Natural numbers

At the end of the nineteenth century, the Italian mathematician Giuseppe Peano sought to formalize arithmetic over the *natural numbers*; that is, the positive integers. *Peano arithmetic* (Kneebone, 1963)

is based on the recursive notion that an arbitrary positive integer is a finite number of successors of zero. That is, all positive integers can be constructed by starting with zero and adding one a finite number of times. So, if we write the successor of a number N as $SUCC(N)$, then:

| decimal | Peano |
|---------|-----------------------|
| 0 | 0 |
| 1 = 1+0 | $SUCC(0)$ |
| 2 = 1+1 | $SUCC(SUCC(0))$ |
| 3 = 1+2 | $SUCC(SUCC(SUCC(0)))$ |
| ... | ... |

4.6.2 Arithmetic

It is then possible to elaborate many arithmetic operations by a *recursive definition* for an operand, say N , with:

- a *base case*, when N is 0
- a *recursion case*, where N is not 0

Recursive functions are often characterized by separate equations for the base case where the defining parameter is 0 and a recursion case where the defining parameter is $SUCC(N)$.

For example, we may define addition ($X + Y$) with cases where Y is 0 or $SUCC(Y)$:

$$\begin{aligned} X + 0 &= 0 \\ X + SUCC(Y) &= SUCC(X + Y) \end{aligned}$$

Any number plus zero is that number (base case). And the sum of any number (X) and the successor of some number (Y), is one more than the sum of those numbers (recursion case); that is, for X plus Y , increment X Y times.

For example, '2 + 2' is

$$\begin{aligned} SUCC(SUCC(0)) + SUCC(SUCC(0)) & \text{ (Recursion case)} \Rightarrow \\ SUCC(SUCC(SUCC(0)) + SUCC(0)) & \text{ (Recursion case)} \Rightarrow \\ SUCC(SUCC(SUCC(SUCC(0)) + 0)) & \text{ (Base case)} \Rightarrow \\ SUCC(SUCC(SUCC(SUCC(0)))) & \Rightarrow 4 \end{aligned}$$

Having defined addition purely in terms of recursion with zero and successor, we can now define multiplication:

$$\begin{aligned} X * 0 &= 0 \\ X * SUCC(Y) &= X + X * Y \end{aligned}$$

so the product of any number and zero is zero. And the product of any number and the successor of some number is the first number added to

the product of those two numbers. That is, for X times Y , add X to zero Y times.

Similarly, we can define subtraction, division, and the taking of the remainder (modulo):

$$\begin{aligned} X - 0 &= X \\ \text{SUCC}(X) - \text{SUCC}(Y) &= X - Y \end{aligned}$$

$$\begin{aligned} X/0 &= 0 \\ X/Y &= \text{SUCC}((X - Y)/Y) \end{aligned}$$

$$X \bmod Y = X - X * (X/Y)$$

This style of recursive definition and evaluation is now used in *declarative* programming languages such as Prolog and Haskell.

4.6.3 Inductive proof

Peano arithmetic may be formalized in a surprisingly small number of axioms. First:

1. 0 is a natural number.
2. If N is a natural number, then $\text{SUCC}(N)$ is a natural number.
3. For any natural number N , $\text{NOT}(\text{SUCC}(N) = 0)$.
4. For any natural numbers M and N , $M = N$ if $\text{SUCC}(M) = \text{SUCC}(N)$.

The first two axioms tell us how to make natural numbers, and the third and fourth establish when natural numbers are equal. Finally, Peano introduced an axiom for *inductive proof*, where some property P of all natural numbers may be proved true by:

- *base case*—proving $P(0)$ true
- *induction case*—assuming $P(N)$ true and proving $P(\text{SUCC}(N))$ true

Thus, inductive proof corresponds closely to recursive definition in terms of 0 and SUCC .

For example, suppose we want to prove that $\text{SUCC}(X + Y) = X + \text{SUCC}(Y)$. For the base case, we need to prove that $\text{SUCC}(X + 0) = X + \text{SUCC}(0)$:

$$1. \quad \begin{array}{l} \text{SUCC}(X + 0) \Rightarrow (+ \text{ base}) \\ \text{SUCC}(X) \end{array} \quad \left| \quad \begin{array}{l} 1. \quad X + \text{SUCC}(0) \Rightarrow (+ \text{ recursion}) \\ 2. \quad \text{SUCC}(X + 0) \Rightarrow (+ \text{ base}) \\ \text{SUCC}(X) \end{array} \right.$$

And for the recursion case, we assume that $\text{SUCC}(X + Y) = X + \text{SUCC}(Y)$ and try to show that $\text{SUCC}(X + \text{SUCC}(Y)) = X + \text{SUCC}(\text{SUCC}(Y))$:

$$\begin{array}{l|l}
1. \quad \begin{array}{l} \text{SUCC}(X + \text{SUCC}(Y)) \Rightarrow (\text{Assum}) \\ \text{SUCC}(\text{SUCC}(X + Y)) \end{array} & \begin{array}{l} 1. \quad X + \text{SUCC}(\text{SUCC}(Y)) \\ \quad \Rightarrow (\text{Assum}) \\ 2. \quad \text{SUCC}(X + \text{SUCC}(Y)) \\ \quad \Rightarrow (\text{Assum}) \\ \quad \text{SUCC}(\text{SUCC}(X + Y)) \end{array}
\end{array}$$

The significance of induction here is that it establishes some property for *any* number, not just some finite set of instances of numbers.

4.6.4 Numbers and sets

Note that we can now use Peano numbers to quantify sets—that is, to determine how many elements they have—by putting distinct set elements into *one-to-one correspondences* with successive Peano numbers. Indeed, we did this implicitly by characterizing sets as:

$$\{element_1, element_2, \dots, element_N\}$$

as we are using the successive numbers 1, 2, ..., N to distinguish the elements. Thus, we will now use the operator *size* to return the number of elements in a finite set; that is, N in the above characterization. Suppose that the operator *take* returns an arbitrary element of a set. Then, using induction on sets, we can define *size* as:

$$\begin{aligned}
size(\{\}) &= 0 \\
size(S) &= \text{SUCC}(size(S \setminus \{take(S)\}))
\end{aligned}$$

The size of an empty set is zero. And the size of a non-empty set is one more than the size of the set less an arbitrary element.

This also gives us a rather different way of looking at numbers as properties of sets. Hence, we might talk about a set whose elements can be put into one-to-one correspondence with the first N integers as having the property of ‘ N -ness’. For example, we can show that the set

$$\{\text{cabbage, carrot, celery, cauliflower, cucumber}\}$$

has five elements, or ‘fiveness’:

| | | | | | |
|---------|---------|--------|--------|-------------|----------|
| integer | 1 | 2 | 3 | 4 | 5 |
| element | cabbage | carrot | celery | cauliflower | cucumber |

Indeed, as discussed above, numbers may well have first arisen as abstractions from putting different collections of things into one-to-one correspondences with each other; for example, people and items of food.

4.7 Paradoxes

We have seen informally that there are strong links between logical argument and manipulating sets. There was considerable interest at

the start of the twentieth century into whether all mathematics could be developed systematically from logic and set theory. The German mathematician David Hilbert enunciated what has come to be known as *Hilbert's programme* (Kleene, 1952) for mathematics, for establishing whether it could be formally demonstrated that foundational systems were *consistent*, *complete*, and *decidable* (Hunter, 1971).

In a consistent system, it is not possible to prove a *contradiction*; that is, that some expression and its negation are both theorems. In a complete system, there are no expressions that can be shown to be theorems by arguments outside the system—say, by constructing truth tables—but that cannot be proved to be theorems. And in a decidable system, there is a *mechanical procedure* for establishing whether or not an arbitrary expression is a theorem. As we shall see, there are very strong correspondences between mechanical procedures and *computations*, and between decidability and the *limits to computation*.

Now, it might seem promising that formal axiomatic systems for *pure* propositional and predicate logic have these highly desirable properties. However, as we shall see, these start to break down as logical formality is further extended with set theory and Peano-style arithmetic.

Returning to set theory, a natural development from sets composed of atomic elements is to allow sets with other sets as members. For example, given sets of cats, dogs, and whales:

$$\{\text{jaguar, cougar, panther}\}\{\text{jackal, dingo, coyote}\}$$

$$\{\text{blue, sperm, right}\}$$

we might construct a set of animals:

$$\{\{\text{jaguar, cougar, panther}\}, \{\text{jackal, dingo, coyote}\},$$

$$\{\text{blue, sperm, right}\}\}$$

For example, given a set of people at a funeral, we might generate the set of pairs of all people who have to shake hands with each other:

$$\{\text{Jo, Chris, Pat}\} \rightarrow \{\{\text{Jo, Chris}\}, \{\text{Jo, Pat}\}, \{\text{Chris, Pat}\}\}$$

For example, given an arbitrary set, it is often useful to generate the *power set*; that is, the set of all subsets including the empty set. For example:

$$\{\text{Jo, Chris, Pat}\} \rightarrow \{\{\}, \{\text{Jo}\}, \{\text{Chris}\}, \{\text{Pat}\}, \{\text{Jo, Chris}\}, \{\text{Jo, Pat}\},$$

$$\{\text{Chris, Pat}\}, \{\text{Jo, Chris, Pat}\}\}$$

Now, an interesting question to ask is whether or not a set can be a member of itself. This is not as odd as it sounds. Consider a library that has separate sections for different subjects and a catalogue for each subject. Then, on the front desk, there might be a central catalogue of all the other catalogues. So, it would not seem unreasonable to include the central catalogue in itself, for consistency and completeness.²

This sounds like a trick, as catalogues contain names of books rather than books themselves, whereas sets seem to have lists of actual

²Tee hee...

elements. However, in the catalogues, the names of books are identifying actual books. So if sets can include the names of other sets, and names uniquely identify sets, then it seems natural to allow a set to include its own name. That is, we allow sets to be *self-referential*.

Here, it might be objected that if we subsequently tried to replace the names of sets by the sets that they named, then we might end up in an infinite regress. That is, when we replaced the name of a self-referential set with its members, we would introduce the self-reference again.

However, within a formal system we can legitimately define a set by specifying its properties using logic rather than by exhaustively listing all its members. That is, we may write:

$$\{\forall var \in S | \text{property}(var)\}$$

that is, the set of elements of S , called var in general, such that $property$ holds for var . For example, we might characterize a potentially infinite list of squares as

$$\{\forall sq \in \text{Nat} | (\exists i \in \text{Nat} | sq = i * i)\}$$

so our new set is all the numbers sq from the natural numbers (Nat) such that sq is the square of some other natural number i .

Thus, if a set itself satisfies some property that characterizes its members, then it might well include itself. For example, the set of sets with at least two elements:

$$\{\forall s | \text{size}(s) > 2\}$$

has at least two members and so could be included in itself.

Now, most sets are not members of themselves. So, following the British philosopher Bertrand Russell, perhaps we could make a set of sets that are not members of themselves:

$$\{\forall S | S \notin S\}$$

We can then ask if this set is or isn't a member of itself. So:

- if it is a member of itself then, by definition, it can't be a member of itself
- if it isn't a member of itself then, by definition, it must be a member of itself

Either way, we have a special sort of contradiction: a *paradox*; that is, a statement that asserts its own negation.

One of Hilbert's requirements for a full formalization of mathematics was that it should be consistent; that is, it should be free from contradictions. *Russell's Paradox* (Kleene, 1952) suggests that this will be problematic if mathematics is to be its own meta-language; that is, if mathematics is to be used to formally characterize itself. As we shall see in later sections, such self-characterization leads ineluctably to paradoxical self-referential constructs that fatally undermine decidability; that is, our ability to determine mechanically whether or not mathematical constructs, including computers and their programs, have seemingly innocuous properties.

4.8 Arithmetizing mathematics and incompleteness

Hilbert’s programme also sought to establish that formalized mathematics is complete; that is, there are no mathematical theorems that cannot be proved to be so. However, in a seminal piece of work in the early 1930s, the Austrian mathematician Kurt Gödel showed that a system that is powerful enough to represent itself must be incomplete; that is, there will be expressions that are clearly theorems but that are not provable in the system.

Gödel’s incompleteness theorems (Nagel and Newman, 1959; Gödel, 1962) are rooted in the curious property that an arbitrary, finite sequence of symbols can be represented as a unique integer, now referred to as *Gödelization*. Gödel used a technique based on recursive functions manipulating multiples of powers of prime numbers; that is, numbers with no divisors apart from themselves and 1. Rather than following the full formality of Gödel’s approach, we will use a simpler representation, and sketch the main features of his result.

Consider again our theorem of Peano arithmetic:

$$SUCC(X + Y) = X + SUCC(Y)$$

We can explode it into a sequence of individual symbols:

$$SUCC(X + Y) = X + SUCC(Y)$$

Suppose that we give distinct values to all the symbols we can use in expressions:

| | | | | | | | | | | | |
|--------|---|-----|----|----|----|------|----|----|----|----|-----|
| symbol | A | ... | X | Y | Z | SUCC | (|) | = | + | ... |
| value | 1 | ... | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | ... |

Then we can represent our expression as a sequence of values:

27 28 24 31 25 29 30 24 31 27 28 25 29

Next, we can turn this into a single number. We start with 0, and repeatedly multiply the number so far by 100 and add in the next symbol value, in reverse order:

| symbol values | number |
|--|----------------------------|
| 27 28 24 31 25 29 30 24 31 27 28 25 29 | 0 |
| 27 28 24 31 25 29 30 24 31 27 28 25 | 29 |
| 27 28 24 31 25 29 30 24 31 27 28 | 2925 |
| 27 28 24 31 25 29 30 24 31 27 | 292528 |
| ... | ... |
| | 29252827312430292531242827 |

This astonishingly large number has the pleasing property that we can extract each symbol value by reversing the process and repeatedly dividing by 100 and taking the remainder. We call this a *Gödel* number.

So, we have seen that we can encode expressions of Peano arithmetic as single Gödel numbers. Such numbers can themselves be expressed as Peano numbers, albeit with unimaginably large numbers of *SUCC* in them. So encoding and decoding expressions, by multiplying by 100 and adding, and dividing by 100 and taking the remainder, can both be expressed as operations in Peano arithmetic itself.

We will now write $\langle\langle e \rangle\rangle$ for the Gödel number for expression e , and not be further concerned with how such numbers are constructed.

In general, such *expressions* are of the form:

- 0, or ...
- a *variable*, say A to Z , or ...
- a *bracketed expression*
- *SUCC* of a *bracketed expression*, or ...
- two *expressions* separated by an *operator* such as $+$, $=$, and so on

Note that this is a recursive definition of *expressions* in terms of their sub-*expressions*. We can use this definition to structure the *syntactic analysis* of *expressions*.

For example, $SUCC(X + Y) = X + SUCC(Y)$ is an *expression* made up of:

1. An *expression*, $SUCC(X + Y)$, which is ...
 - a A *SUCC* of a *bracketed expression* $(X + Y)$, made up of ...
 - (i) An *expression*, which is a *variable* X , followed by ...
 - (ii) An *operator* $+$, followed by ...
 - (iii) *expression*, which is a *variable* Y , all followed by ...
2. An *operator* $=$, followed by ...
3. An *expression* $X + SUCC(Y)$, made up of:
 - (a) an *expression*, which is a *variable* X , followed by ...
 - (b) an *operator* $+$, followed by ...
 - (c) An *expression* $SUCC(Y)$, made up of:
 - (i) A *SUCC* of a *bracketed expression* (Y) , made up of ...
 - (A) An *expression*, which is a *variable* Y .

We could extend this definition to encompass proofs in the style we have used above, as sequences of the expressions produced at each step with their proof justifications. Then, in the spirit of Gödel, we could encode entire proofs as numbers and write recursive functions to pull them apart and analyse them, in particular to check whether or not they really do constitute valid proofs.³ Alas, as we discuss below, coming up with a proof in the first place is far more problematic than checking a claimed proof.

Most importantly, Gödelization enables true mathematical self-reference. That is, it is possible to write arithmetic expressions that refer

³In a similar spirit, contemporary *proof carrying code* consists of a computer program along with a proof of its input/output correctness. This is used to give assurance of program integrity, as a doubtful recipient can check that the proof is correct.

to the properties of numbers representing other arithmetic expressions. And, as we shall shortly see, this way lies paradox. Our account will follow Nagel and Newman (1959).⁴

First, in performing proofs we often want to be able to *replace* a variable in an expression with some other expression. We will write

$$\text{replace}(v, e1, e2)$$

to mean the Gödel number of the new expression after replacing all occurrences of the variable with number v , with the expression with number $e1$, in the original expression $e2$.

For example, in

$$\text{replace}(\langle\langle C \rangle\rangle, \langle\langle A * B \rangle\rangle, \langle\langle C + D \rangle\rangle \Rightarrow \langle\langle A * B + D \rangle\rangle)$$

we have replaced C in $C + D$ with $A * B$, all by manipulating Gödel numbers.

Next, suppose that x is the Gödel number encoding a proof sequence that proves that the logical assertion with number z is a theorem. We represent this as follows:

$$\text{proves}(x, z)$$

Note that *replace* makes a new number for a new expression, where *proves* is a logical assertion that is either true or false.

Now, consider the logical assertion

$$A. \forall x : \text{NOTproves}(x, z)$$

that is, for all numbers x , it is not the case that the sequence with number x is a proof of the logical assertion with number z , for some arbitrary z .

As a special case, consider

$$B. \forall x : \text{NOTproves}(x, \text{replace}(\langle\langle Y \rangle\rangle, Y, Y))$$

Remembering that $\langle\langle Y \rangle\rangle$ is the number for variable Y , this asserts that for all Gödel numbers x , it is not the case that the sequence with number x is a proof of the logical assertion resulting from replacing all occurrences of variable Y , with the number Y , in the expression with number Y .

It is really important to note that Y is a meta-variable denoting the concrete Gödel number for some expression, where $\langle\langle Y \rangle\rangle$ is the specific Gödel number, 25 in our approach, for the expression consisting solely of the specific variable Y .

Suppose that logical assertion B has number N . Now, consider the assertion

$$C. \forall x : \text{NOTproves}(x, \text{replace}(\langle\langle Y \rangle\rangle, N, N))$$

which states that for all Gödel numbers x , it is not the case that the sequence with number x is a proof of the logical assertion resulting from

⁴Without in any way subscribing to their claim that Gödel's results shows that human intelligence is necessarily more powerful than artificial intelligence.

replacing all occurrences of variable Y with number N in the expression with number N .

C asserts that there is no number x that is a proof sequence for the assertion with number $replace(\langle\langle Y \rangle\rangle, N, N)$. Now, compare B and C. To get C, we replaced all the occurrences of Y in B with N . And we said that B has Gödel number N . So C's Gödel number must be $replace(\langle\langle Y \rangle\rangle, N, N)$. Thus, C asserts that itself has no proof!

C is a perfectly decent assertion of mathematical logic, encoded entirely in mathematical logic. And C is patently true. So C is a theorem of mathematical logic that cannot be proved within mathematical logic. So we have to conclude that mathematical logic is incomplete.

To put this another way, for any given claimed proof sequence, we can check whether or not it actually proves some assertion, but there are true assertions for which we can't ever find proof sequences. This is a strong hint that formalized mathematics is undecidable; that is, that there is no mechanical way to determine whether or not an arbitrary assertion is a theorem. As we shall see when we explore Alan Turing's work below, this has profound implications in establishing the limits to computation.

4.9 Infinities

Peano's characterization of integers as arbitrary successors of zero is unbounded; that is, for *any* integer I we can form an immediately next integer $SUCC(I)$ without limit. Thus, there seems to be an *infinite* number of integers.

This itself is a strange assertion. It assumes that:

- in some sense, there are such things as integers
- integers themselves are things that can be counted
- we can give denote a quantity that is not finite

The first assumption does not require deep consideration of the ontological status of numbers; that is, whether or not numbers are in any sense more real than symbols with conventional meanings. That is, we can focus on numbers purely as symbolic constructs and mathematics as rules for manipulating symbols, which additionally capture the essential properties of real-world countability.

Then, if we are treating numbers as symbols, the second assumption follows directly. Each distinct number has a distinct symbolic representation as a finite number of successors of zero, and so the Peano sequence can be put into one-to-one correspondence with itself.

Informally, infinity is about quantities whose counting never terminates. And we are used to using the symbol ∞ to represent infinity. However, it seems reasonable to ask whether there are different sorts of infinity; that is, whether or not different infinities have different sizes. The German mathematician Gregor Cantor made a pioneering study of infinities: we will next survey his results (Kleene, 1952).

It is usual to distinguish between the *ordinal* numbers, that is, the usual numbers for counting, and the *cardinal* numbers, that is, the numbers for denoting the sizes of sets. Conventionally, while finite sets have finite cardinals from the natural numbers, transfinite sets have named cardinals. For example, Cantor started by naming the *cardinality* of the natural numbers as \aleph_0 (aleph null).

Intuitively, it seems that there must be many different-sized infinities, depending on where a sequence starts and how it is generated. For example, it might appear that there are half as many natural numbers divisible by two than natural numbers: in the sequence 1, 2, 3, and so on, only every second number is divisible by two. Curiously, however, Cantor's work shows that any infinity whose members have distinct finite representations can be put into one-to-one correspondence with the natural numbers, and so must have the same cardinality \aleph_0 .

For example, consider again the numbers that are divisible by two:

| | | | | | | | | | |
|--------|---|---|---|---|----|----|----|----|-----|
| number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
| double | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | ... |

The double of any natural number is divisible by two. Hence, there must be as many numbers divisible by two as natural numbers.

This process of systematically pairing up the members of a infinite set with the natural numbers is called *enumeration*; the members are then said to be *enumerable* or *denumerable* or *countable*.

Now consider the *rational* numbers;⁵ that is, numbers made up of the ratio of two natural numbers—for example, 27/159. Now, if we regard a rational number as being an arithmetic division that we can evaluate, then most do not have finite representations when fully worked out; for example, 4/3 is 1.333333... . Nonetheless, we can still put the rational numbers into one-to-one correspondence with the natural numbers if we use the ratio representation, by systematically listing them. A ratio n/d consists of a *numerator* (n) over a *denominator* (d), so we can construct a table:

⁵Decimal fractions.

| | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | d | 1 | 2 | 3 | 4 | 5 | 6 | ... |
| n | 1 | 1/1 | 1/2 | 1/3 | 1/4 | 1/5 | 1/6 | ... |
| | 2 | 2/1 | 2/2 | 2/3 | 2/4 | 2/5 | 2/6 | ... |
| | 3 | 3/1 | 3/2 | 3/3 | 3/4 | 3/5 | 3/6 | ... |
| | 4 | 4/1 | 4/2 | 4/3 | 4/4 | 4/5 | 4/6 | ... |
| | 5 | 5/1 | 5/2 | 5/3 | 5/4 | 5/5 | 5/6 | ... |
| | 6 | 6/1 | 6/2 | 6/3 | 6/4 | 6/5 | 6/6 | ... |

We can then read off the table entries against the natural numbers, starting in the top left-hand corner and systematically traversing each lower left to upper right diagonal:

| | | | | | | | |
|--------|-----|-----|-----|-----|-----|-----|-----|
| number | 1 | 2 | 3 | 4 | 5 | 6 | ... |
| ratio | 1/1 | 2/1 | 1/2 | 3/1 | 2/2 | 1/3 | ... |

Note that the ratios include many representations of the same values; so, for example, 1 appears as $1/1$, $2/2$, $3/3$, and so on. But we have shown that there are as many ratios as natural numbers, and that the rationals are a subset of the ratios, so once again, there are as many rational numbers as natural numbers: that is, they have the same cardinality \aleph_0 .

Quite contrary to our familiar understanding that all operations on infinity give infinity, and hence infinity is unique, Cantor reasoned that there are infinite cardinalities beyond \aleph_0 . For example, a *power set* is the set of all subsets of some original set. Figure 4.6 shows the power sets for the sets $\{1\}$, $\{1,2\}$, $\{1,2,3\}$, and $\{1,2,3,4\}$. Note that the power set includes the empty set and the original set itself.

If the set has size n , then the power set has size 2^N .

Thus the cardinality of the power set of the natural numbers of size \aleph_0 must be 2^{\aleph_0} , called \aleph_1 (aleph one), which is strictly bigger than \aleph_0 .

4.10 Real numbers and Cantor diagonalization

It seems that there are also numbers that are not rational; in other words, that cannot be expressed as simple ratios of natural numbers. Such numbers are called *irrational* or *real*, and they frequently crop up in everyday arithmetic.

For example, properties of circles are expressed in terms of π , the ratio of the circumference to the diameter. Then, in terms of a circle's radius R , the area is πR^2 .

Like all real numbers, π can only be given a finite characterization by an equation capturing an infinite computation that will generate an infinite number of decimal place digits. One of the earliest computations for π , by Madhava of Sangamagrama, from 1400 BC, is as follows:

$$4/1 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 \dots$$

Usually, for calculations involving π , we use the crude approximation $22/7$, or the finite expansion of some equation giving the familiar 3.14159.. to some useful number of places.

Another common source of real numbers is in taking square roots; for example, to find the length between diagonally opposite corners of a

| Set | Power set | Set size | Power set size |
|---------------|--|----------|----------------|
| $\{1\}$ | $\{\{\},\{1\}\}$ | 1 | 2 |
| $\{1,2\}$ | $\{\{\},\{1\},\{2\},\{1,2\}\}$ | 2 | 4 |
| $\{1,2,3\}$ | $\{\{\},\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$ | 3 | 8 |
| $\{1,2,3,4\}$ | $\{\{\},\{1\},\{2\},\{3\},\{4\},\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\},\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\},\{1,2,3,4\}\}$ | 4 | 16 |

Fig. 4.6 Power sets.

room. As shown in Chapter 7, the square root of 2 cannot be expressed as a ratio of rational numbers.

We will critically discuss the ‘reality’ of the real numbers in more detail in Chapter 7. For now, if we consider numbers as lying on some notional line, in ascending numerical order, then it seems that there might be an arbitrary number of real numbers in between two adjacent rational numbers. Nonetheless, perhaps the reals also have cardinality \aleph_0 ; that is, maybe they could be put into one-to-one correspondence with the natural numbers, just as the rationals could.

In an elegant demonstration, Cantor showed that, in fact, there must necessarily be more reals than natural numbers. Suppose that there are as many reals as natural numbers, and we can characterize them all such that we can generate their expansions.

A real number consists of a whole integer part and a decimal fraction part following the decimal point. Here, we will just focus on reals between 0 and 1, so that they all start with 0. Then we can write down their digits as follows, where $d_{i,j}$ is the j th digit of the i th real:

$$\begin{array}{rcccccccc}
 0 & . & d_{1,1} & d_{1,2} & d_{1,3} & d_{1,4} & d_{1,5} & \dots \\
 0 & . & d_{2,1} & d_{2,2} & d_{2,3} & d_{2,4} & d_{2,5} & \dots \\
 0 & . & d_{3,1} & d_{3,2} & d_{3,3} & d_{3,4} & d_{3,5} & \dots \\
 0 & . & d_{4,1} & d_{4,2} & d_{4,3} & d_{4,4} & d_{4,5} & \dots \\
 0 & . & d_{5,1} & d_{5,2} & d_{5,3} & d_{5,4} & d_{5,5} & \dots \\
 0 & . & d_{6,1} & d_{6,2} & d_{6,3} & d_{6,4} & d_{6,5} & \dots \\
 \dots & & & & & & &
 \end{array}$$

Then, we can construct a new number D , with digits:

$$0 . D_1 D_2 D_3 D_4 D_5 D_6 \dots$$

where the j th digit of D is formed by, say, adding 1 to the j th digit of number $d_{j,j}$, and replacing 10 with 0. Then, D_1 is not the same as $d_{1,1}$, and D_2 is not the same as $d_{2,2}$, and D_3 is not the same as $d_{3,3}$, and so on, so there is no digit d_i that is the same as D . But we had assumed that we could write down all the reals in correspondence with the natural numbers. So there must be more reals than natural numbers, and than rational numbers.

The cardinality of the reals may be shown to be \aleph^1 (Kleene, 1952).

As we shall discuss in a later chapter, Cantor diagonalization is not without controversy, as it involves assumptions about being able to characterize and hence generate expansions of arbitrary reals to arbitrary places. Nonetheless, we shall see that Turing deployed diagonalization in a somewhat different context to establish fundamental limits to computation.

4.11 Turing Machines

As we mentioned above, the third requirement of Hilbert’s programme was to try to establish if formalized mathematics was decidable; that

is, whether or not there was a mechanical procedure for determining whether or not an arbitrary assertion was a theorem. Mechanical procedures, also known as *algorithms* or *effective procedures*, consist of sets of *rules* that return definitive results after a finite number of *applications* (see Fig. 4.7; see also Swinbourne, 1875).

Many mathematicians concentrated on trying to find rules to underpin ‘procedures’, without reference to their ‘mechanical’ application. In contrast, the British mathematician Alan Turing took the ‘mechanical’ aspect as his primary focus, and elaborated the foundational idea of what are now called *Turing Machines (TMs)* (Turing, 1937).

Turing started by considering what happens when a human being ‘does mathematics’, in particular arithmetic. Typically, some initial symbols, say a sum, are written on a piece of paper, and then systematically manipulated according to fixed rules. Usually this involves crossing or rubbing out some or all of the symbols, and writing down new symbols, until a final answer is reached. During this process, humans will work backwards and forwards through the symbols, and, if they run out of paper, they may continue on a new piece.

Turing then came up with the idea of a very simple machine that could behave in an analogous manner. Rather than working with symbols on

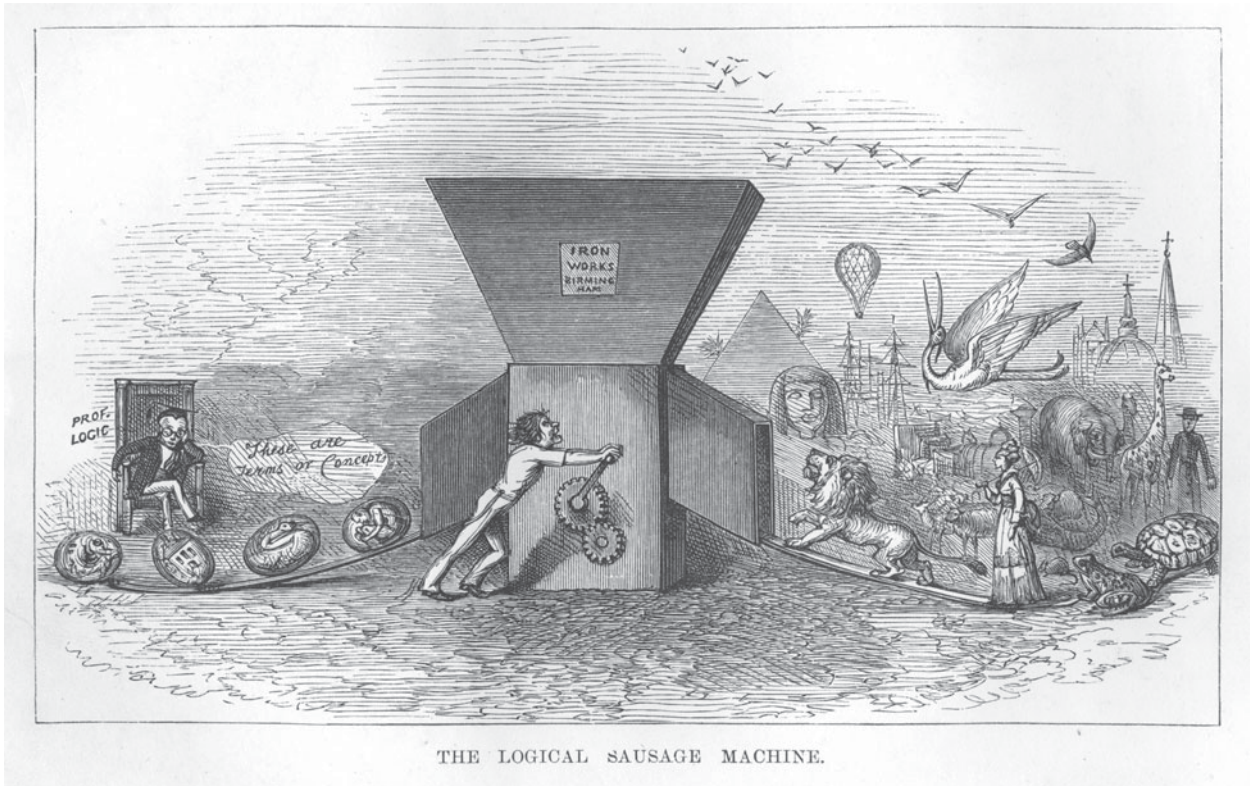


Fig. 4.7 The logical sausage machine.

pieces of paper, TMs process a linear *tape* made up of a sequence of *cells*, where each cell can hold just one symbol. The machine also has a *tape head* that can inspect and change the symbol in the *current cell*.

A TM includes a set of rules for solving some specific problem. At each stage of execution, only a subset of the rules are likely to be relevant; and, after a rule has been chosen and applied, it may be necessary to consider a different subset of rules for the next stage of execution. In a TM, rules are grouped into subsets corresponding to what are called *states*. Each rule is then of the following form:

$$(oldstate, oldsymbold) \rightarrow (newstate, newsymbol, direction)$$

This may be read as follows:

If the current state is *oldstate* and the head is inspecting *oldsymbol*, then change the symbol to *newsymbol*, move the head according to *direction*, and change the state to *newstate*.

So, we might envisage a TM as shown in Fig. 4.8. An instance of the problem is written down, symbol by symbol, on to the cells of a tape of the right size, and the current state is set to identify an initial subset of rules. The execution cycle is then as follows:

- find a rule corresponding to the current state (a) and symbol (b)
- change the current state to the new state (c)
- change the current symbol to the new symbol (d)
- move the tape head either left or right (e)

When either end of the tape is reached, a new empty cell is joined on.

To get a better feel for how TMs work, consider making a copy of a sequence of 1s with *s at either end. Suppose that we:

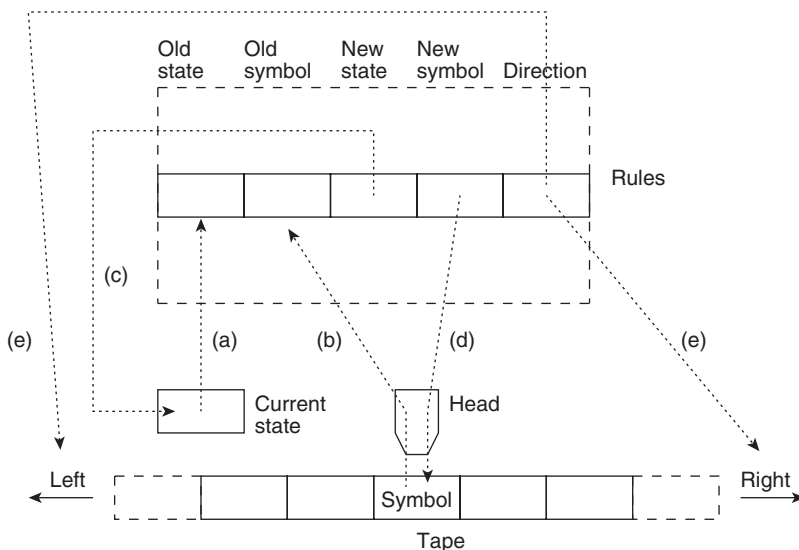


Fig. 4.8 The Turing Machine.

1. Start at the left-hand * and move right to the first 1.
2. Replace the 1 with an X to show that we've copied it, and move right.
3. Move repeatedly right to the next blank cell, write a 1, and move left.
4. Move repeatedly left back to the X, replace it with a 1, and move right to the next 1.

and repeat the process from (2) until there are no more 1s to copy. Then, (5) we move repeatedly to the right to a blank cell, write a *, and halt.

We can visualize this activity as the *state transition diagram* shown in Fig. 4.9. Note that an empty cell is denoted by a .. Then, we can encode the machine as follows:

| | |
|---------------------------------|--|
| $(1, *) \rightarrow (2, *, R)$ | $(4, 1) \rightarrow (4, 1, L)$ |
| $(2, 1) \rightarrow (3, X, R)$ | $(4, *) \rightarrow (4, *, L)$ |
| $(2, *) \rightarrow (5, *, R)$ | $(4, X) \rightarrow (2, 1, R)$ |
| $(3, 1) \rightarrow (3, 1, R)$ | $(5, 1) \rightarrow (5, 1, R)$ |
| $(3, *) \rightarrow (3, *, R)$ | $(5, _) \rightarrow (_, *, \text{halt})$ |
| $(3, _) \rightarrow (4, 1, L)$ | |

Running the machine on an initial tape with *11* proceeds as follows, with the current head position shown between < and >:

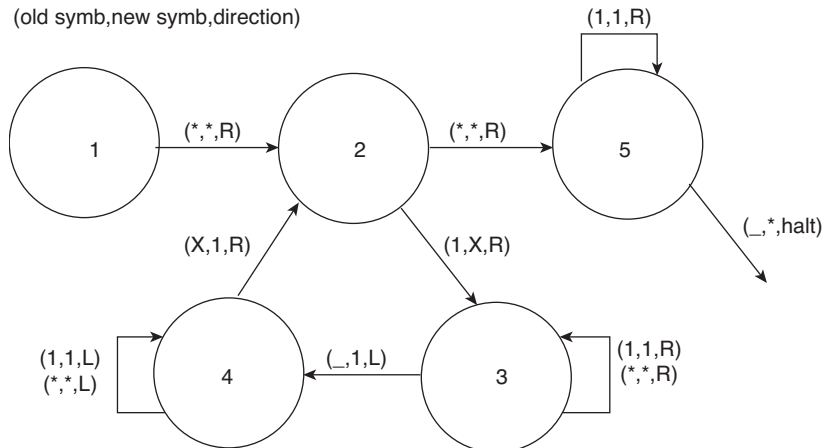


Fig. 4.9 The state transition diagram for copy.

| state | tape | state | tape |
|-------|--------------------------|-------|----------------------------|
| 1. | $\langle * \rangle 11*$ | 3. | $*1X*\langle 1 \rangle$ |
| 2. | $*\langle 1 \rangle 1*$ | 3. | $*1X*1\langle _ \rangle$ |
| 3. | $*X\langle 1 \rangle *$ | 4. | $*1X*\langle 1 \rangle 1$ |
| 3. | $*X1\langle * \rangle$ | 4. | $*1X\langle * \rangle 11$ |
| 3. | $*X1*\langle _ \rangle$ | 4. | $*1\langle X \rangle *11$ |
| 4. | $*X1\langle * \rangle 1$ | 4. | $*11\langle * \rangle 11$ |
| 4. | $*X\langle 1 \rangle *1$ | 5. | $*11*\langle 1 \rangle 1$ |
| 4. | $*\langle X \rangle 1*1$ | 5. | $*11*1\langle 1 \rangle$ |
| 2. | $*1\langle 1 \rangle *1$ | 5. | $*11*11\langle _ \rangle$ |
| 3. | $*1X\langle * \rangle 1$ | halt | $*11*11\langle * \rangle$ |

Now, copying sequences of symbols may seem to be somewhat removed from meta-mathematics. However, we can represent natural numbers in *unary*—that is, as sequences of 1s—just like the Peano representation as sequences of *SUCC*s of 0. Then, we can sketch how to construct Peano arithmetic operations for *SUCC*, $+$, and $*$ as follows, where N_i is some Peano number and UN_i is that number in unary:

- $SUCC(N)$:
 - tape is $*UN*$
 - TM copies UN with additional 1 at the end
- $N_1 + N_2$:
 - tape is $*UN_1 * UN_2*$
 - TM copies UN_2 immediately after copy of UN_1
- $N_1 * N_2$:
 - tape is $*UN_1 * UN_2*$
 - TM copies UN_2 once for each 1 in UN_1

In principle, we could manipulate mathematical assertions by constructing TM equivalents of recursive Peano arithmetic with Gödelization. However, we can instead take the much simpler approach of representing assertions directly on a TM tape, symbol by symbol. We can then craft TMs to manipulate such assertions by bolting together rule sets for finding, copying, and modifying symbol sequences.

For example, let's consider how to implement $replace(v, e1, e2)$, which replaces variable v with expression $e1$ in expression $e2$. First, we can construct a set of TM instructions to find an occurrence of one symbol sequence in another symbol sequence, though we won't give the details here. Then, if the tape is $*v * e1 * e2$, we repeatedly copy symbols from $e2$ until we find a v when we copy $e1$ instead.

4.12 Universal TM and undecidability

As we have seen, executing a TM one step, from a current symbol and state, involves finding an instruction with a matching old symbol and

old state, replacing the current symbol on the tape with the new symbol, changing the current state to the new state, and moving the tape head left or right.

Now, TM instructions are themselves made up of finite sequences of symbols, so they can be held, symbol by symbol, on a TM tape. Thus, with Turing, we could construct a *Universal TM (UTM)* that will behave like an arbitrary TM, given a tape containing that TM's instructions, tape start state, and initial head position. A UTM is itself defined by a set of TM instructions that once again involve finding, copying, and changing symbol sequences.

We will not further discuss the construction of a UTM here: Minsky (1972) provides an accessible description. However, we should note that the representation and manipulation of TMs on TM tapes is yet another instance of self-reference in meta-mathematics, and again anticipates paradox.

Turing was particularly interested in *computable* numbers; that is, real numbers that can be calculated as decimal fractions by what he termed *finite means*: 'a number is computable if its decimal can be written down by a machine'.

As we've seen above, at each step a TM tape may contain both symbols that are part of the final result and symbols that are used to record intermediate stages. For example, in the copy machine, we marked a 1 that we were about to copy with an *X*. Turing distinguished between *circular* TMs, which generate only a finite number of result symbols, and *circle-free* machines that never stop generating result symbols. Thus, a computable real must be generated by a circle-free TM, as it has an infinite number of digits after the decimal point.

Now, TMs are composed of finite numbers of instructions, each with a finite number of states and symbols, so it is possible to systematically generate and hence enumerate them. The process is very like generating the rational numbers as ratios of natural numbers. We start with the single instruction from one possible state and one possible symbol. Then we produce the single instructions with one possible state and two possible symbols. Then we produce all the pairs of instructions from combinations of these single instructions. Then we produce all the single instructions with two states and one symbol, the new pairs of single instructions, and the triples of single instructions. And so on.

Turing also characterized TMs as *description numbers*, from instructions defined with a small set of symbols, using what is effectively Gödelization, by repeatedly multiplying by 10 and adding. So, another approach (Chaitin, 1987) is to generate successive integers and keep those that are description numbers of correctly formed TM instruction sequences.

Either way, the TMs for computable reals form a subset of all the TMs. On the dubious assumption that we can tell which TMs generate computable reals, we could set them running one at a time, generating successive digits:

| | | | | | | | | |
|--------|---|---|-----------|-----------|-----------|-----------|-----------|-----|
| TM_1 | 0 | . | $d_{1,1}$ | $d_{1,2}$ | $d_{1,3}$ | $d_{1,4}$ | $d_{1,5}$ | ... |
| TM_2 | 0 | . | $d_{2,1}$ | $d_{2,2}$ | $d_{2,3}$ | $d_{2,4}$ | $d_{2,5}$ | ... |
| TM_3 | 0 | . | $d_{3,1}$ | $d_{3,2}$ | $d_{3,3}$ | $d_{3,4}$ | $d_{3,5}$ | ... |
| TM_4 | 0 | . | $d_{4,1}$ | $d_{4,2}$ | $d_{4,3}$ | $d_{4,4}$ | $d_{4,5}$ | ... |
| TM_5 | 0 | . | $d_{5,1}$ | $d_{5,2}$ | $d_{5,3}$ | $d_{5,4}$ | $d_{5,5}$ | ... |
| TM_6 | 0 | . | $d_{6,1}$ | $d_{6,2}$ | $d_{6,3}$ | $d_{6,4}$ | $d_{6,5}$ | ... |
| ... | | | | | | | | |

Now, it might seem that we could use diagonalization to construct a new real whose i th digit is different to digit d_{ii} of TM_i . Thus, the TM for the new real can't be in the enumeration, so the new real can't be computable. But the process of constructing this uncomputable real from computable reals is itself computable by a simple TM that picks up appropriate digits and modifies them. So this argument seems to disprove the claim that we can enumerate all the TMs that generate computable reals.

However, we have made the fundamental assumption that each enumerated TM is circle free; that is, it goes on generating digits forever without halting. Suppose, then, that we can build a variant of the UTM—say, the *halting UTM (HUTM)*—which can examine an arbitrary TM and work out whether or not that TM halts. Note that this HUTM must itself halt, say, printing a Y if the TM halts and N if the TM doesn't halt: that is, the HUTM is circular.

Now, we can modify this HUTM to make it circle free if the TM halts. Suppose that the HUTM has an instruction:

$$(State_i, Symbol_i) \rightarrow (halt, Y, -)$$

Here, in $State_i$, the HUTM has decided that the TM doesn't halt, so it prints a Y and halts.

Suppose that we modify this to the following:

$$\begin{aligned} (State_i, Symbol_i) &\rightarrow (State_{i+1}, Y, R) \\ (State_{i+1}, -) &\rightarrow (State_{i+1}, -, R) \end{aligned}$$

Now, the HUTM writes a Y and changes to state $State_{i+1}$, where it loops endlessly, moving right over empty tape cells.

If we apply this non-halting modified HUTM (nHUTM) to a copy of itself, then:

- if the nHUTM copy halts, then the nHUTM writes a Y and doesn't halt
- if the nHUTM copy doesn't halt, then the nHUTM writes a N and halts

So, we have a paradox and we must conclude that we can't build the HUTM.

First, this shows that it is impossible to tell if an arbitrary TM halts. In particular, we can't tell if an arbitrary TM generates a computable

real, because we can't tell whether or not it generates an infinite sequence of digits. That is, if we've had no output for a while from a TM, we can't tell if it has halted or is just thinking.

More profoundly, this confounds the third requirement of Hilbert's programme for an effective procedure to decide whether or not an arbitrary assertion is a theorem; we have no way of telling whether or not a TM embodying an alleged decision procedure will ever halt and confirm or deny that the assertion is a theorem.

During the 1930s, several other mathematicians were exploring effective decision procedures. The Frenchman Emile Post (1936) also focused on the 'mechanical' aspect of effectivity, and, quite independently, developed a formulation very similar to Turing's, now known as the *Post machine (PM)*.⁶

⁶In the period before the Internet, when international phone calls were both expensive and hard to organize, researchers who were not already in contact typically found out about each other's work only after it had been presented at a conference or published in a journal.

One important difference between a TM and PM is that Post required that there be an infinite tape available from the start; whereas, in the TM, the tape may grow indefinitely but at any given instance it is finite in length. In Aristotelian terms (Aristotle, 1983), Post sought an *actualized* infinity, where Turing only needed a *potential* infinity. We shall return to this distinction in Chapter 7 when discussing schemes for transcending the limits to computing that Turing established.

4.13 Computational procedures

In contrast to Turing and Post, mathematicians such as the Americans Stephen Kleene and Alonzo Church focused on the 'procedural' aspect of 'effective procedure', seeking to develop logical systems that could capture decidability.

4.13.1 Recursive function theory

Kleene's theory of recursive functions (Kleene, 1952) is a direct descendant of Peano's work. Whereas Peano's formulation permits recursive functions of arbitrary form, Kleene distinguishes explicitly between *primitive* and *general* recursive functions.

A primitive recursion function is based on a schema that will always terminate. That is, every recursion must make progress towards some termination property. The examples that we considered above of addition and multiplication in Peano arithmetic are primitive recursive, as the base case terminates with 0 and the recursion case on $SUCC(N)$ is defined in terms of some function of the strictly smaller N . Primitive recursion is also termed *bounded minimization*.

In contrast, a general recursive function is based on a schema that recurses so long as some arbitrary property of the argument holds, typically seeking a least value satisfying some property. Unlike primitive recursion, general recursion may continue for an arbitrary number of times. Thus, general recursion is also termed *unbounded minimization*.

In this formulation, decidable problems are those characterized by general recursive functions that always halt, whereas semi-decidable problems are those characterized by *partial* recursive functions that may not halt. Note that partial recursive functions have equivalent TMs, but these TMs may not embody effective computations: an effective computation is known to halt, but the TM for a partial recursive function may not.

4.13.2 λ calculus

Church's λ calculus (Church, 1936), which we will now quickly explore, took a markedly different approach to that of Kleene.

The λ calculus may be thought of as a notation for describing *substitutions*, a bit like the *replace* function above. λ *expressions* are made up of bracketed sequences of *variables* that form loci for replacement by other *expressions*. More formally, a λ *expression* may be:

1. A variable—*var*.
2. A *function*— λ *var*.*expression*.
3. An *application*—*expression*₁ *expression*₂.
4. A bracketed *expression*—(*expression*).

Note that this astonishingly simple calculus does not have any of the familiar numerical forms or operators. Nonetheless, it is every bit as powerful as number-theoretic predicate logic, and as TMs.

To go into a bit more detail, in a function

$$\lambda \text{ var.expression}$$

the *var* (after the λ) is called the *bound variable* and identifies places in the *body expression* (after the $.$) at which substitutions may be made.

For example, consider the function

$$\lambda \mathbf{x}.\mathbf{x} \text{ (identity)}$$

This has bound variable \mathbf{x} and body \mathbf{x} . So, if this function is applied to any expression, it just returns that expression.

Next, consider the function

$$\lambda \mathbf{f}.\lambda \mathbf{a}.\mathbf{f} \ \mathbf{a} \text{ (apply)}$$

This has bound variable \mathbf{f} and body $\lambda \mathbf{a}.\mathbf{f} \ \mathbf{a}$. This function applies one expression from \mathbf{f} to another from \mathbf{a} .

Now consider the function

$$\lambda \mathbf{s}.\mathbf{s} \ \mathbf{s} \text{ (self apply)}$$

This has bound variable \mathbf{s} and body $\mathbf{s} \ \mathbf{s}$. So, this function applies an expression to itself.⁷

⁷Alarm bells ...

⁸Here, we just provide a brief overview of β reduction. For the full subtleties of free and bound variables, and α renaming to avoid free variable capture, see (Church, 1936).

Expressions are evaluated by a process of substitution termed β reduction.⁸ We will indicate one β reduction step with \Rightarrow_{β} .

Then, for an application

$$expression_1 \ expression_2$$

if the first expression $expression_1$ can be evaluated to a function of the form $\lambda \text{ var} . expression_3$, then the second expression $expression_2$ should be substituted in the body $expression_3$ for the bound variable var .

Given an expression, applications are successively *beta* reduced from the outside in, until none are left. The expression is then said to be in *normal form*.

For example:

1. $((\lambda f . \lambda a . f \ a) (\lambda s . s \ s)) (\lambda x . x) \Rightarrow_{\beta}$
2. $(\lambda a . (\lambda s . s \ s) \ a) (\lambda x . x) \Rightarrow_{\beta}$
3. $(\lambda s . s \ s) (\lambda x . x) \Rightarrow_{\beta}$
4. $(\lambda x . x) (\lambda x . x) \Rightarrow_{\beta}$
5. $\lambda x . x$

At step 1, f is replaced with $\lambda s . s \ s$. At step 2, a is replaced with $\lambda x . x$. At step 3, s is replaced twice with $\lambda x . x$. And at step 4, x is replaced with $\lambda x . x$.

In other words, if we apply the ‘apply’ function to the ‘self apply’ and the ‘identity’ functions (1), we will apply the ‘self apply’ function to the ‘identity’ function (2,3), and finally apply the ‘identity’ function to itself (4), giving the ‘identity’ function (5).

Not every expression has a normal form; that is, there are expressions whose evaluation never halts. For example, consider applying the ‘self apply’ function to itself:

1. $(\lambda s . s \ s) (\lambda s . s \ s) \Rightarrow_{\beta}$
2. $(\lambda s . s \ s) (\lambda s . s \ s) \Rightarrow_{\beta}$
3. $(\lambda s . s \ s) (\lambda s . s \ s) \Rightarrow_{\beta}$
4. ...

Here, both occurrences of s are replaced by $\lambda s . s \ s$, and we’re back where we started.

4.13.3 Meta-mathematics in λ calculus

It is straightforward to represent propositional logic in λ calculus. To begin with, consider extending logic with the form

$$\text{if } condition \text{ then } expression_1 \text{ else } expression_2$$

where:

- if the *condition* is **true**, then $expression_1$ is evaluated
- if the *condition* is **false**, then $expression_2$ is evaluated

if `if` behaves like a function of three arguments:

$$\lambda e1.\lambda e2.\lambda c.(c\ e1)\ e2\ \text{if}$$

where the third (`c`) somehow selects the first (`e1`) or second (`e2`). Then, `true` and `false` should behave like functions of two arguments, where `true` selects the first and `false` selects the second:

$$\begin{aligned} &\lambda x.\lambda y.x\ \text{true} \\ &\lambda x.\lambda y.y\ \text{false} \end{aligned}$$

Now, consider the truth table for negation:

| | |
|--------------------|--------------------|
| <code>X</code> | <code>¬ X</code> |
| <code>false</code> | <code>true</code> |
| <code>true</code> | <code>false</code> |

This is like

$$\text{if } X \text{ then false else true}$$

or

$$\begin{aligned} &((\lambda e1.\lambda e2.\lambda c.(c\ e1)\ e2)\ \text{false})\ \text{true} \Rightarrow_{\beta} \\ &(\lambda e2.\lambda c.(c\ \text{false})\ e2)\ \text{true} \Rightarrow_{\beta} \\ &\lambda c.(c\ \text{false})\ \text{true}\ (\neg) \end{aligned}$$

Thus:

$$\begin{aligned} &\neg\ \text{false} \Rightarrow \\ &(\lambda c.(c\ \text{false})\ \text{true})\ \text{false} \Rightarrow_{\beta} \\ &(\text{false}\ \text{false})\ \text{true} \Rightarrow \\ &((\lambda x.\lambda y.y)\ \text{false})\ \text{true} \Rightarrow_{\beta} \\ &(\lambda y.y)\ \text{true} \Rightarrow \\ &\text{true} \end{aligned}$$

Now, again consider the truth tables for conjunction and disjunction:

| | | | | | |
|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| <code>X</code> | <code>Y</code> | <code>X ∧ Y</code> | <code>X</code> | <code>Y</code> | <code>X ∨ Y</code> |
| <code>false</code> | <code>false</code> | <code>false</code> | <code>false</code> | <code>false</code> | <code>false</code> |
| <code>false</code> | <code>true</code> | <code>false</code> | <code>false</code> | <code>true</code> | <code>true</code> |
| <code>true</code> | <code>false</code> | <code>false</code> | <code>true</code> | <code>false</code> | <code>true</code> |
| <code>true</code> | <code>true</code> | <code>true</code> | <code>true</code> | <code>true</code> | <code>true</code> |

For *AND*, whenever `X` is `true` the result is the same as `Y`, and otherwise the result is `false`:

$$\text{if } X \text{ then } Y \text{ else false} \equiv \lambda x.\lambda y.(x\ y)\ \text{false}\ (\wedge)$$

For *OR*, whenever `X` is `true` the result is `true` and otherwise the result is the same as `Y`:

$$\text{if } X \text{ then true else } Y \equiv \lambda x.\lambda y.(x\ \text{true})\ y\ (\vee)$$

We can then systematically use this `if ... then ... else ...` style of construct to represent sets, numbers, Peano arithmetic, and predicate logic.

Recursion is enabled by the pleasingly named *paradoxical combinator*, also known as Y :

$$Y = \lambda f.(\lambda s.f(ss))(\lambda s.f(ss))$$

Y has the property that, for any lambda expression F , then

$$YF = F(YF)$$

as

$$\begin{aligned} YF &\Rightarrow_{\beta} \\ &(\lambda f.(\lambda s.f(ss))(\lambda s.f(ss)))F \Rightarrow_{\beta} \\ &(\lambda s.F(ss))(\lambda s.F(ss)) \Rightarrow_{\beta} \\ &F(\lambda s.F(ss))(\lambda s.F(ss)) \Rightarrow_{\beta} \\ &F(YF) \end{aligned}$$

That is, Y passes a copy of an expression F with a copy of Y into itself, so this self-copying can further continue. It will, therefore, come as no surprise that self-referential paradoxes leading to undecidability can be formulated in λ calculus. In particular, it can be shown that there is no mechanical procedure for determining whether or not an arbitrary λ expression has a normal form. In other words, there is no way to tell if β reduction of an arbitrary λ expression ever terminates.

4.14 The Church–Turing thesis

Church and Turing both argued that their abstractions captured the essential features of mechanical procedures/algorithms, which Turing termed ‘computability’ and Church ‘effective calculability’.

Kleene’s recursive function theory gave precise characterization to the blend of predicate logic and Peano arithmetic that Gödel has used to establish his incompleteness results. At the time, Church thought that λ calculus and recursive function theory both captured the same notion of effective calculability. Indeed, just as TMs and λ calculus can be used to represent what is effectively recursive function theory, so:

- TMs can represent λ calculus
- λ calculus can represent TMs
- recursive function theory can represent TMs and λ calculus

This suggests that these formalisms really are strongly *equivalent* in two important senses. First, all have the same *expressive power*. That is, there are no computations that can be captured by some of these formalisms but not by others, as an arbitrary computation can be translated from formalism to formalism. Secondly, a result established directly in any one formalism must hold through translation in all the

others. In particular, undecidability results for one formalism will apply to the other formalisms.

In what is now called the *Church–Turing thesis*, it is widely held that all possible characterizations of computability will prove to be equivalent. Of course, this is a ‘thesis’ rather than a ‘theorem’, as it cannot be proved: there is no way to know all possible computability formalisms. However, every new computability formalism since this pioneering work has been shown to be equivalent to at least one of the classic formalisms.

In particular, contemporary CPUs can easily be programmed to behave like TMs, showing that both CPUs and programming languages satisfy the Church–Turing thesis. Thus, all the classic decidability results apply to contemporary computers and their programs, with deep practical implications for what we can hope to know about them. For example:

- if it is impossible to tell beforehand if an arbitrary program stops, it is impossible to tell:
 - how long it will take to run
 - how much memory it will require
 - how much power it will consume
- if it is impossible to tell whether or not an arbitrary assertion is a theorem, then it is impossible to prove whether or not:
 - an arbitrary program is *correct*—that is, satisfies some input/output behaviour characterized in logical assertions
 - two arbitrary programs do the same thing
 - one arbitrary program has been plagiarized from another program

And in more recent results developed directly for programming languages, the American computer scientist Gregory Chaitin (1998) has shown that it is impossible to tell if:

- a program is the most *elegant*—that is, the smallest possible for some problem
- a sequence of digits generated by a program really is *random*—that is, each subsequence is equally likely

Of course, there may still be *heuristics* for checking aspects of these properties; that is, procedures that can often give useful answers for many specific problem instances, even if they are not effective in always being guaranteed to halt with a definitive answer.

4.15 Machines, programs, and expressions

We started this chapter by introducing the idea of meta-mathematics; that is, the mathematical formalization of mathematics itself. And we have latterly focused on the undecidability of mechanical procedures for checking whether or not mathematical assertions are theorems, noting that different mathematicians have placed differing emphases on ‘procedure’ and ‘mechanical’.

This corresponds to our familiar distinction between a ‘program’ as a sequence of instructions and a ‘computer’ as a machine for executing programs by carrying out the instruction sequences. Curiously, while it seems obvious that human mathematicians ‘do’ maths, the notion of human programmers ‘doing’ programs appears eccentric: computers do programs. However, programmers really do ‘do’ programs when they develop and debug them; that is, they quite explicitly try to trace through their programs as if they were computers executing them. Indeed, originally the term ‘computer’ meant a person who carried out a rule-based activity, typically solving or checking the results of hard sums. And, as we’ve seen above, Turing drew out his characterization of TMs from a simplified model of humans following rules. Nonetheless, without something to understand it, be it human or machine, a program is just an inert set of symbols.

We can clarify this by again considering the difference between a TM and a Universal TM. First, a TM is a specific machine that embodies some specific mechanical procedure. A TM cannot be reprogrammed. If a TM is needed for a different procedure, then it must be built afresh.

Now, if we separate out the head and tape mechanisms, which are common to all TMs, then the differences between TMs lie solely in the fixed sets of instructions they are built to follow. Even a Universal TM has a specific, fixed set of instructions. However, those instructions enable a UTM to behave like any other TM, provided that it is given an appropriate TM description on its tape. That is, the UTM is an *interpreter* of TM descriptions. A TM description itself consists of a sequence of transitions and an initial tape, written in a precise notation. Thus, we could view the TM description notation as a *programming language*, with the meanings of programs—that is, TM descriptions—defined by the UTM.

This is directly analogous to our common-or-garden computers. Inside a computer is a *Central Processing Unit (CPU)*, which is an electronic machine for executing sequences of *machine code* instructions. The CPU is physically connected to *main memory*, which may contain electronic representations of machine code programs—that is, sequences of instructions—and of data values. The CPU can then systematically access the memory, pick up instructions, and execute them to manipulate data, also from memory.

In TM terms, a CPU is a universal machine code machine; that is, a CPU is an interpreter for programs written in the machine code language. Thus we can equate:

| Computer | UTM |
|----------------------|-----------------|
| CPU | UTM control |
| Machine code | TM instructions |
| Main memory | UTM tape |
| Machine code program | TM description |
| Data | TM tape |

Machine code is a very simple notation and it is hard for humans to construct substantial machine code programs. Instead, we have developed *high-level* programming languages, such as Java and HTML, and high-level data descriptions, such as XML. Nonetheless, for a computer to execute a program in a high-level language, it must be *compiled*—that is, translated—into machine code instructions. Similarly, data in a high-level form such as XML must be converted into a low-level form suitable for manipulation in memory.

These days, rather than compiling high-level programs into machine code, it is increasingly common to deploy an *abstract* or *virtual machine*; that is, a program that interprets programs written in some other⁹ language. Then further virtual machines can be described in languages for virtual machines, so computer systems may consist of hierarchies of virtual machines. For example, a PC may process a stream of XML data using a Java virtual machine running on the .NET virtual machine that underpins Microsoft Windows systems. Nonetheless, all these virtual machines finally boil down to the behaviour of a physical CPU interpreting instruction sequences in its specific machine code.

⁹Or the same ...

In contrast to TMs, which embody specific TM descriptions, and UTMs, which can interpret arbitrary TM descriptions, λ calculus and recursive function theory expressions do not describe specific machines. Rather, they are akin to TM descriptions, waiting for a UTM to animate them. Hence, in a very deep sense, being able to represent expressions in λ calculus or recursive function theory as TMs is akin to compiling them into TM descriptions for execution on a UTM.

Of course, it is possible to build both physical and virtual machines to execute λ calculus or recursive function theory expressions. Indeed, the very high-level *functional* programming languages, like *Haskell* (Marlow, 2010) and *F#* (McNamara, 2010), are directly based on these formalisms, and are often implemented via virtual machines, of which Peter Landin's SECD machine (Landin, 1964) is the classic example.

It is also possible to build specific machines directly from programs. Thus *Field Programmable Gate Arrays (FPGAs)* are highly flexible, very low-level hardware devices that can be configured for specific 'mechanical procedures' from specific programs. Once again, however, FPGAs are also configured as *soft cores*; that is, as interpreters for machine codes for extant CPUs.

4.15.1 Von Neumann machines

While TMs provide foundational underpinnings for the theory of computing, practical computers are based on a very different design that evolved from work in the United Kingdom, Germany, and the United States before and during the Second World War. The *von Neumann machine* is named after the Hungarian-American mathematician John von Neumann (1945), whose report gave the first formal characterization of the *stored program computer* that is so ubiquitous today. von Neumann machines are conceptually close to Babbage's Analytical

Engine, in distinguishing processor, memory, and input–output, and to the Universal Turing Machine in holding both programs and data in unitary memory, and in providing a general-purpose processor that interprets machine instructions.

In contrast, *Harvard machines*, while retaining the UTM’s general-purpose processor, are closer to the Analytical Engine in holding programs and data in separate memories. Contemporary designs—for example, the ARM—may conflate this useful but simplistic distinction.

Heat, information, and geometry

5

5.1 The triumph of digital computation

Although Turing had advanced the idea of the universal digital computer during the 1930s, right up to the end of the 1960s there was an active tradition of analogue computing that was only gradually replaced by digital techniques. The tradition of analogue gun control computers continued, as we have mentioned (p. 40), well beyond the 1940s. Through the 1950s and 1960s, analogue electronic computers were serious rivals to digital ones for many tasks. The analogue approach ultimately lost out because of the twin problems of accuracy and programmability.

Analogue machines were significantly less accurate than digital ones. Wass (1955) stated that

Errors as small as 0.1% are hard to achieve; errors of 1% are not unusual, and they are sometimes as large as 5%–10%. This is in striking contrast to digital machines.

(p. 3).

This meant that analogue computers had to be applied in areas where high accuracy was not as important as speed of operation. They were used in aerodynamic control applications such as missile guidance, where their errors could be tolerated. In these applications, the basic aerodynamic coefficients of the equations being solved were known to only about 10% accuracy, whereas the real-time response needed could not be attained by then-existing digital machines.

Analogue machines were programmed by plug boards. Units to carry out operations of adding, subtracting, multiplying, differentiating, and so on were interconnected by removable wires in the same way as in a manual telephone exchange. Output was typically in the form of real-time CRT traces of the graph of the function being computed. Storage of analogue quantities remained a constant problem. MacKay and Fisher (1962) describes the use of modified Williams (1948) tubes as analogue stores. (Bergman, 1955) found that over a process of eight read/write cycles with such tubes the cumulative error amounted to 10%.

| | | |
|------------|---|------------|
| 5.1 | The triumph of digital computation | 83 |
| 5.2 | Analogue computing with real numbers | 84 |
| 5.3 | What memories are made of | 86 |
| 5.4 | Power consumption as a limit | 91 |
| 5.5 | Entropy | 95 |
| 5.6 | Shannon's information theory | 111 |
| 5.7 | Landauer's limit | 114 |
| 5.8 | Non-entropic computation | 117 |
| 5.9 | Interconnection | 122 |

The sources of error in such machines were as follows:

1. Systematic scaling errors due to uncertainty in the parameters of the resistors and other passive components being used.
2. Distortions due to non-linearities in the amplifiers used.
3. Random fluctuations due to environmental factors. These include thermal noise in the amplifiers, and in the case of Williams tubes the effects of stray magnetic fields on the trajectories of the electrons.

These problems are representative of the difficulties that plague any attempt to carry out accurate analogue computation.

5.2 Analogue computing with real numbers

In principle, an analogue device can be thought of as computing with real numbers. These real numbers can be divided into two classes, the parameters supplied as inputs to the calculation and the variables that take on time-varying values as the computation proceeds. Whilst in a digital machine the parameters and the variables are uniformly represented by strings of bits in some digital store, in an analogue machine they are typically of different types. The variables in an electronic analogue computer, for example, are instantaneous voltages, whilst the parameters are provided by the physical setting of resistors, capacitors, and so on. These components are subject to manufacturing errors and the provision of higher-specification components becomes exponentially expensive:

That the machine is digital however has a more subtle significance. It means firstly that numbers can be represented by strings of digits that can be as long as one wishes. One can therefore work to any desired degree of accuracy. This accuracy is not obtained by more careful machining of parts, control of temperature variations, and such means, but by a slight increase in the amount of equipment in the machine. To double the number of significant figures, would involve increasing the amount of the equipment by a factor definitely less than two, and would also have some effect in increasing the time taken over each job. This is in sharp contrast with analogue machines, and continuous variable machines such as the differential analyser, where each additional decimal digit required necessitates a complete redesign of the machine, and an increase in the cost by as much as a factor of 10.

(Turing, 2004).

The parameters and variables are, in the mathematical abstraction, transformed by operators representing addition/multiplication, and so on. In an actual analogue computer, these operators have to be implemented by some apparatus the effect of which is analogous to that of the mathematical operator. The analogy is never exact. Multipliers turn out to be only approximately linear, adders show some losses, and so on. All of these mean that even if the variables were perfect representations of

the abstract concept of real numbers, the entire apparatus would only perform to bounded accuracy. But no physically measurable attribute of an apparatus will be a perfect representation of a real number.

Voltage, for example, is subject to thermal and ultimately quantum noise. We can never set up an apparatus that is completely isolated from the environment. The stray magnetic fields that troubled Bergman will never totally vanish. It may be objected that what we call digital computers are built out of transistors working with continuously varying currents. Since digital machines seem to stand on analogue foundations, what then privileges the digital over the analogue?

The analogue character of, for instance, voltage is itself a simplification. The charge on the gate of a transistor is, at a deeper level, derived from an integral number of electrons. The arrival of these electrons on the gate will be subject to shot noise. The noise will follow a Poisson distribution, whose standard deviation is $\approx \sqrt{n}$, with n the mean number of electrons on the gate. It is clear that by increasing n , making the device larger, we control the signal-to-noise level. For large enough transistors, this noise can be reduced to such an extent that the probability of switching errors becomes negligible during the normal operation of the computer. It is only if we make devices too small that we have to bear the cost of lower reliability. We look at this in more detail in Section 5.7.1.

Suppose that we have a device, either electronic or photonic, that measures in the range 0–1 and that we treat any value above 0.6 as a boolean TRUE and any value below 0.4 as a boolean FALSE, and say that in between the results are undefined. Similar coding schemes in terms of voltage are used by all logic families. For simplicity, we will assume that our measurements are in the range 0–1 V (volts).

Now suppose that our switching device is designed to be fed with a mean of 100 quanta when we input a TRUE to it. Following a Poisson distribution, we have $\sigma = 10$, so we need to know the probability that the reading will be indeterminate, below 0.6 V, or how likely it is that only 60 quanta will arrive given the shot noise; in other words, a deviation of 4σ from the mean. Using tabulations of the normal distribution, we find that this probability is 0.0000317.

Consider a computer with a million gates, each using 100 electrons. Then 31 of the gates would yield indeterminate results each clock cycle. This is unacceptable.

Assuming the million gates and a 1 GHz clock, and that we will tolerate only one indeterminate calculation a day, we want to push this down to a failure probability per gate per cycle of about 10^{-19} or 9σ . This implies $\sigma = 0.044$ V, which can be achieved when our capacitor is sufficiently large that about 500 electrons will generate a swing of 1.0 V. The figures are merely illustrative, and posed at a level of abstraction that would apply to both electronic and optical computers, but they illustrate the way in which reliability and the number of quanta used to represent TRUE and FALSE trade off against one another.

Our digital computers are reliable because they use a large number of degrees of freedom—for example, a large number of quanta of charge—to

represent one bit. The number can be chosen so that the probability of a read error is very low, and then, following a read, the signal can be regenerated. This prevents the propagation of errors that are the bane of analogue computing.

5.3 What memories are made of

Turing's initial idea of a universal computer was, as described in Chapter 4, one that used a tape memory. During the Second World War, he had experience with the code-breaking computer's code named Colossus (Hodges, 1983; Gannon, 2006), which was built to break into the German 'Fish' code. These machines used very high-speed paper tape readers as a read only store. Most were destroyed after the war, but a few were retained for the generation of one-time pads punched tape for use in British diplomatic ciphers. But the use of tape was not an essential part of Turing's conception of a digital computer. His Automatic Computing Engine, designed in 1945, replaced tapes with mercury delay lines (Turing, 2004). The essential point was that the memory could be used to hold either instructions or data. This had been implicit in his original paper (Turing, 1937). Based on his experience since then, he was by 1950 giving a very general definition of a digital computer:

The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail. We may suppose that these rules are supplied in a book, which is altered whenever he is put on to a new job. . . . A digital computer can usually be regarded as consisting of three parts:

- (i) Store.
- (ii) Executive unit.
- (iii) Control.

The store is a store of information, and corresponds to the human computer's paper, whether this is the paper on which he does his calculations or that on which his book of rules is printed.

(Turing, 1950, p. 436).

His mature conception of the memory was that it addressed randomly rather than sequentially:

The information in the store is usually broken up into packets of moderately small size. In one machine, for instance, a packet might consist of ten decimal digits. Numbers are assigned to the parts of the store in which the various packets of information are stored, in some systematic manner. A typical instruction might say:

'Add the number stored in position 6809 to that in 4302 and put the result back into the latter storage position.'

Needless to say it would not occur in the machine expressed in English. It would more likely be coded in a form such as 6809430217.

(Turing, 1950, p. 437).

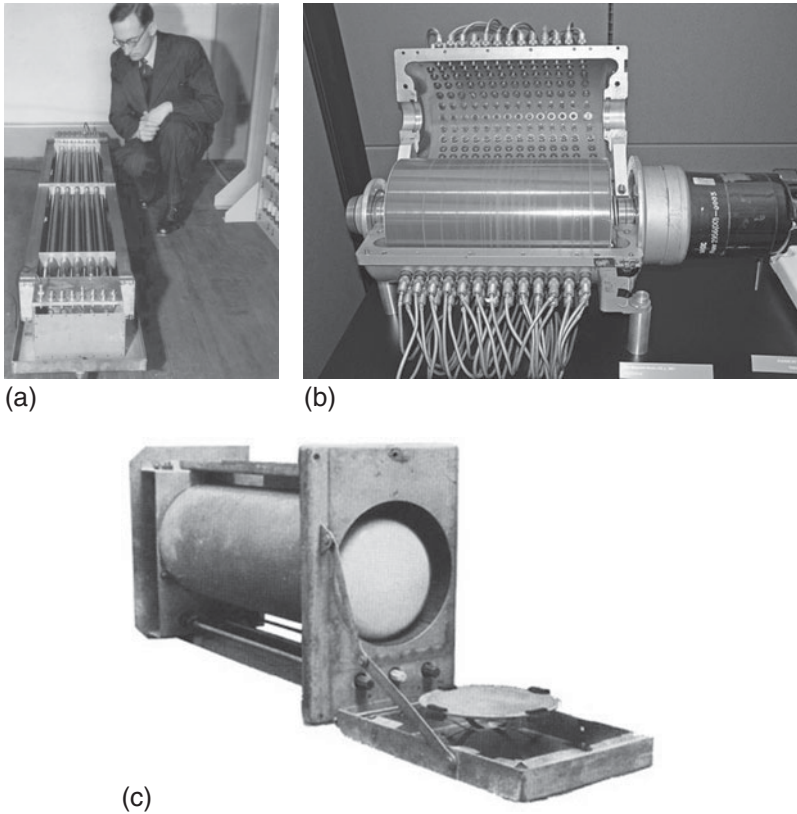


Fig. 5.1 Memories available to Turing in the 1940s. (a) Mercury delay lines, shown with Maurice Wilkes, the designer of EDSAC. (b) The magnetic drum. (c) The Williams tube. Photos (a) and (c) Wikimedia images; (b) photo by Nial Kennedy.

At the time at which Turing was designing the ACE or working at Manchester, designing a suitable random access store was still a considerable problem. Although the conceptual design was for a store that could be accessed as a series of numbered slots, the actual implementations available (Fig. 5.1) were all cyclical.

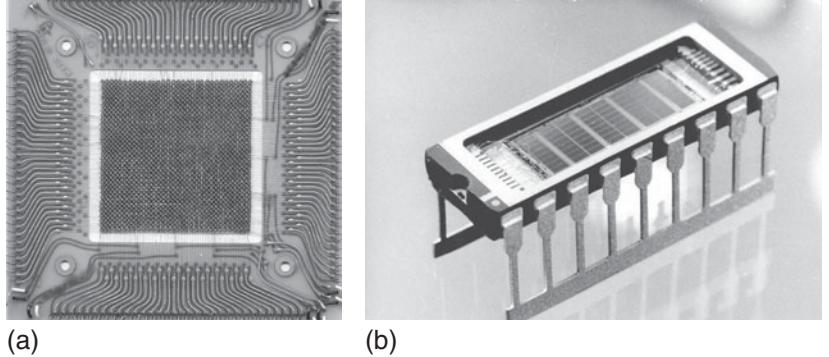
In the Williams tube (Williams, 1948; Lavington, 1975), a cathode ray tube was raster scanned to read out bits deposited as electric charge on the surface, an inherently cyclical/serial process.

In the drum store, the data was written to a set of tracks on the drum and read by fixed heads at slightly offset positions. Data from any head could be randomly selected, but the user had to wait for a particular word to come back round as the drum rotated.

In a delay line, the sound waves passed down a mercury-filled tube and were picked up at the other end to be regenerated (Lavington, 1980). In both cases, the limitation on the computers performance was the cycle time at which data came back round. On average, we would expect memory access for a random word to take approximately half the memory cycle time.

The first genuinely random access memory did not come until the invention of magnetic core planes such as the one shown in Fig. 5.2.

Fig. 5.2 (a) A magnetic core memory storing 1024 bits of information, dating from 1961, made by the Control Data Corporation. (b) The first German 1 Mb Dynamic Random Access Memory chip of 1989, made by VEB Carl Zeiss Jena.



With these, the access time could be much faster than in delay lines or drums. The time to access a core plane of n bits could be modelled as

$$t_m = a \log(n) + b\sqrt{n} + c$$

where the logarithmic term $a \log(n)$ derives from the decode logic that selects row and column lines, the $b\sqrt{n}$ term derives from the time it takes for the signal to propagate along the row and column lines, and the constant term c is the time taken for a single core itself to respond to being selected. Why does it take this form?

Well, in the photograph the plane has 32 rows and 32 columns. When an access is performed, one of the rows is selected. To select one row thus takes 5 bits, since $32 = 2^5$. So the signal that selects the row must have had to pass through five AND gates. Activation of a row will cause all 32 bits on that row to be read out along the column lines. The bit that is wanted must be selected from these 32 using another five address lines. Therefore, the total number of AND gates involved in the critical path will be ten. There will be five to select the row and five for the column. There are 1024 bits in the memory, and $\log_2(1024) = 10$.

Next, consider the time to propagate along the rows and down the columns. For a randomly chosen memory cell, the row signal has to propagate halfway across and the column signal has to propagate halfway down the column. So the mean propagation distance in our example will be $\frac{32}{2} + \frac{32}{2} = 32$ cells. Therefore, in Fig. 5.2 what we have is

$$t_m = 10a + 32b + c$$

with a being the delay through an AND gate, and b the propagation time across a distance of one cell, and c the time for the selected core to switch.

Clearly, for large arrays, the square root term will dominate. In consequence, there will be an optimal size of plane. Suppose that with the technology of 1961, the optimal size was 1024 bits. Rather than creating a plane of 4096 bits whose propagation time would have been twice as great, it would have been better to fit the machine with four planes of 1024 bits at a cost of only two more AND gate delays. The optimal

size depended on the relative sizes of the constants a and b , both of which would change over time as AND gates improved, and as physically smaller cores meant that the signal propagation distances fell.

During the 1970s, the preferred technology for computer memories changed from induction to capacitance. Capacitive memories became commercially viable with the Mostek 4096 bit dynamic RAM chip in 1973. These quickly displaced core memory because they were faster, cheaper, and more compact. If you look at the DRAM shown in Fig. 5.2(b), you can see that it was made of 64 smaller arrays, illustrating our previous argument that there is a sweet point for a given two-dimensional memory array technology, an array size that minimizes both decode and propagation delays. For larger memories, one creates multiples of these optimally sized arrays. It is no accident that both the core memory and the DRAM were organized as two-dimensional arrays. A combination of logic, economy, and thermodynamics have forced this approach on to computer designers.

It is a logical approach because the organization of memory as a grid allows the decode logic to be shared across many memory cells. In the old 1024 bit core memory, each row or column wire would have been fed by an output of a five-level decode tree, a deeper version of Fig. 5.3. A p -level decode tree requires $2^p - 1$ demultiplexers, so our core matrix will have required 31 demux's for the rows and 31 for the columns; in general, this approximates to $2\sqrt{n}$ components for an array of n bits. If each memory bit was fully decoded, then n demultiplexers would be used, so the grid arrangement uses much less resource in gates.

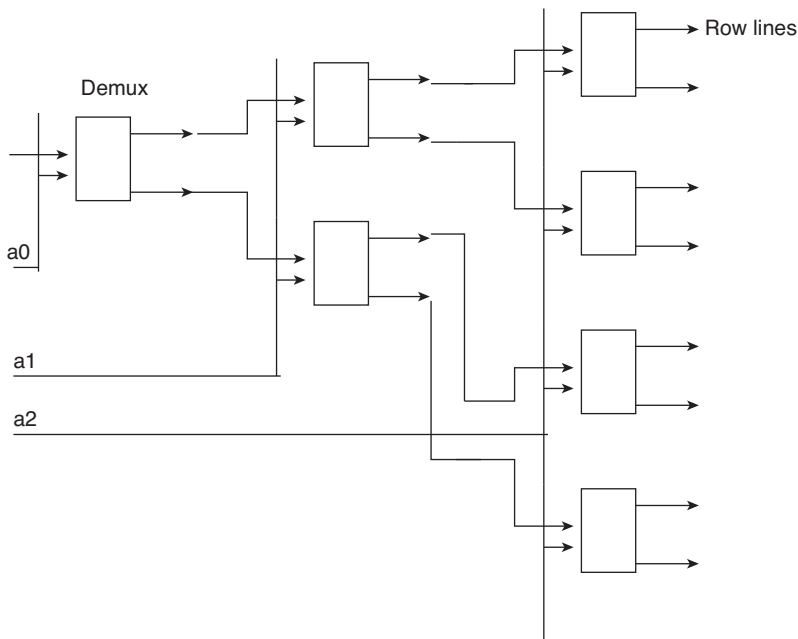


Fig. 5.3 A three-level decode tree.

There is another reason why two-dimensional arrays are advantageous. We should remember that computers are labour-saving devices. They are only worth using if the total work set free by using the machine exceeds the total work that goes into making it. If we look at the calculator shown in Fig. 3.16, its cost in the 1950s was around two month's average earnings. It would only be worthwhile for somebody to buy one if they were an accountant or had some other job that involved them in a great deal of calculation over a long period of time.

By 1972, when the first mass-produced electronic calculator, the Sinclair Executive, was launched in the United Kingdom, its price was the equivalent of two weeks' wages. Today, a basic four-function electronic calculator costs the equivalent of about 20 minutes' average wages. At today's price, the time saved on even occasional calculations will make buying a calculator worthwhile for most families. The fall in price—which is most obvious with calculators, but has affected all computing equipment—is a consequence of the adoption of what is basically a printing technique to produce electronic logic. The chip layout is transferred to the chip by a parallel photographic process. This allows all transistors on the chip to be simultaneously manufactured. Successive generations of chips have small and smaller features on them. Improvements in imaging technology—the ability to use and focus ever shorter waves of electromagnetic radiation—have allowed repeated halvings of feature sizes. These halvings of feature sizes translate into an exponential growth in the number of transistors per hour that a state-of-the-art plant can make. Despite the growth in capital costs of chip plants (Polcari, 2005), this has still led to an exponential cheapening of transistors.

But for this to work it has been essential that there exists an axis normal to the circuit, along which its constitutive information can be transmitted in parallel. In a three-dimensional universe, this constrains us to making two-dimensional printed products.

Prior to the adoption of printing techniques, some three-dimensional core arrays were built. But core memories were manufactured sequentially, threading wires through the cores one at a time. Indeed, although machines were built to automate construction, in the end it was found more cost-effective to use oriental cheap labour to make them by hand (Pugh *et al.*, 1991). For cores, there was not the same imperative to adopt a purely planar organization.

Suppose that we did try to mass produce chips with a 3D structure. Since the manufacturing process is basically a printing process, at least one printing step would be required for each layer of our 3D structure. If we wanted to improve our gate count on a 2D chip by a factor of four, we could do so provided that we could print each feature at half the size. Using the third dimension, we could also increase the number of gates by a factor of four by having four times as many manufacturing steps to lay down 4 times as many layers. If we grow gates by shrinking feature sizes, the number of manufacturing steps remains constant. If we do it by adding layers, the cost increases at least in proportion to

the additional layers. In fact, the cost metric will be worse than this for a multi-layer chip, since manufacturing errors will grow exponentially with the number of layers used. For these reasons, 3D chip manufacture has never been economically competitive.

Another factor is the problem of heat dissipation. A three-dimensional structure can only conduct or radiate away heat from its surface. If heat is to be removed from the interior, it has to be made porous and liquid or gas blown through to cool it. This was feasible with cores since they were porous, but it is much more of a problem with silicon chips. These have to dissipate their heat from their surfaces, and as circuit technology has advanced, the problem of heat dissipation has become more severe.

5.4 Power consumption as a limit

The first generations of electronic computers used vacuum tubes—or ‘valves’, as they were called—to perform binary switching. In a valve, the cathode has to be heated by a red-hot element; in consequence, a valve computer used huge amounts of power. Lavington recounts how amazed people were by the amount of power used by the Manchester Mk1 in 1948 (Lavington, 1975). Power consumption was substantially reduced in the second and third generations of mainframe computers, which used discrete transistors or small-scale integrated circuits (Bashe *et al.*, 1986; Pugh *et al.*, 1991). The key to the inexpensive mass production of electronic computers was the development of the microprocessor, initially as a device for cheapening the production of desktop calculators (Ceruzzi, 2003).

5.4.1 CMOS

The first generation of microprocessors like the Intel 4004 shown in Fig. 5.4(a), used what are called PMOS transistors. These were followed shortly after by NMOS chips like the 8080 and then, from the mid-1980s, CMOS chips have dominated. The aim of successive changes in circuit technology has been to combine circuit speed with power economy.

To understand what is involved, look at the NMOS gate shown in Fig. 5.5. The diagram shows a cross-section through a gate with air above and silicon below. The silicon is doped in two ways: N-type doping, which imparts a negative charge; and P-type doping, which imparts a positive charge. The two N-type regions are labelled S and D, for ‘source’ and ‘drain’. In the middle, labelled G, is the gate, a capacitor plate that is isolated from the silicon by a layer of insulating silicon dioxide. The gate acts as a switch that can be opened or closed by applying a voltage to it.

In the off situation shown in Fig. 5.5(a), current cannot flow from the source to the drain as it is opposed by the bias voltage between the N-type silicon of the source and the P-type silicon under the gate. If a positive charge is applied to the plate G, however, it induces an

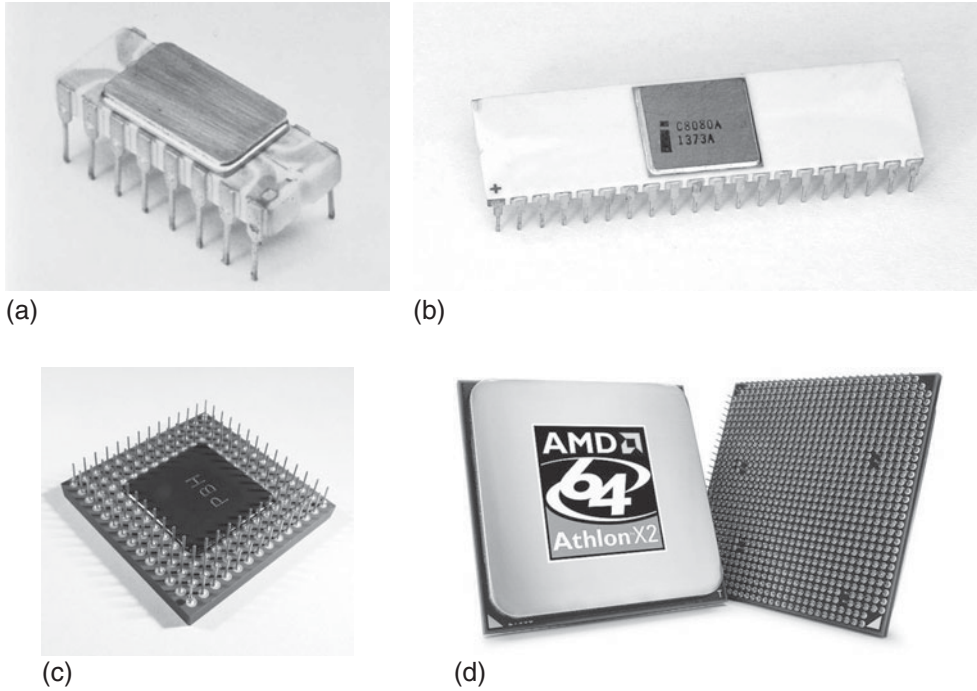
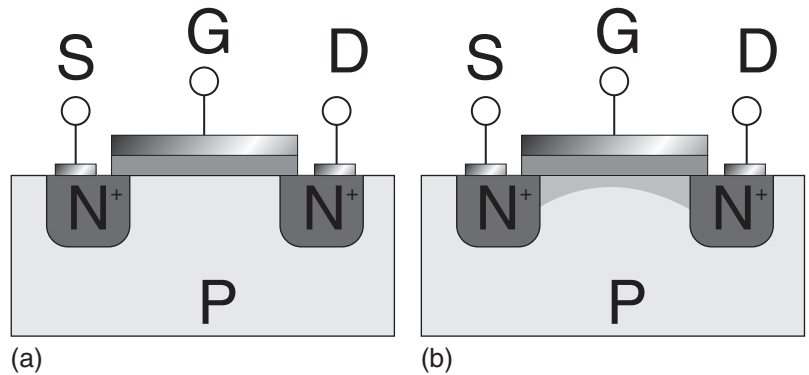


Fig. 5.4 Intel and AMD microprocessors. (a) A 4004 PMOS 4-bit chip, from 1971. This was the first microprocessor and was designed to be used in Busicom desk calculators. (b) An 8080 NMOS 8-bit chip, from 1974. This was used in the first-generation personal computers, such as the Altair 8800 or the IMSAI 8080. (c) An Intel 80386 CMOS 32-bit chip. (d) An AMD Athlon-64 dual core CMOS 64-bit chip, from 2005.

Fig. 5.5 An NMOS gate: (a) in the off state; and (b) in the on state.



N-type depletion zone immediately under the gate. Consequently, there is a continuous N-type region from the source to the drain and current flows. In a PMOS gate, the source and drain are built of P-type silicon and in the off mode the channel is N. In this case, a negative charge on the gate induces a P-type depletion zone between the source and drain.

CMOS chips combine the use of both NMOS and PMOS gates, the motivation being to reduce power consumption. If we look at the NMOS

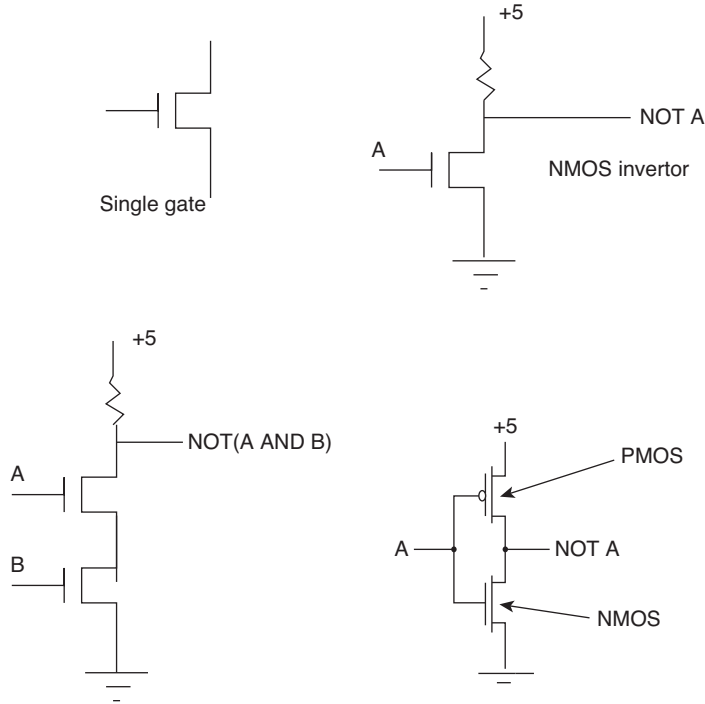


Fig. 5.6 Left to right: a basic NMOS transistor symbol, an NMOS NOT gate, an NMOS NAND gate, and a CMOS NOT gate. Note the pull-up resistors on the NMOS NOT gate and. When the output of the NMOS NOT gate is low, a current will flow through this resistor to ground through the transistor. In the CMOS case, the output is connected either to high or to ground, with no leakage current.

gates in Fig. 5.6, we see that when the output is low a leakage current will flow through the pull-up resistor to ground. The CMOS gate avoids this leakage current by connection of the output either to high or to ground. Thus there is no static current flow for a CMOS gate. The only current flow occurs during switching, in order to overcome the capacitance of the circuit attached to the output of the gate. This led to a considerable economy in the use of power when the initial transfer from NMOS to CMOS technology occurred.

As clock speeds have risen, however, the currents required to overcome capacitance have increased so much that power consumption has again become a pressing problem. Let us construct an ultra-simple model of the power consumption of a CMOS processor. We can approximate a CMOS chip by a collection of capacitors, a portion of which charge and discharge on each clock cycle. This charging and discharging dissipates power. For our simple model, we will consider only the capacitance of the gates themselves, not of the wires between them. The capacitance of a parallel plate capacitor is given by

$$C = \kappa 8.854 \times 10^{-12} \frac{A}{d}$$

where C is the capacitance in farads (F), A is the area in square metres, κ is the dielectric constant of the material between the plates, and d is their separation in metres. For the silicon dioxide insulator used in most chips, κ is 3.9. Let us initially consider only the capacitance of the CMOS

gates themselves, ignoring the wires for now. Since the capacitance of the gates is inversely proportional to the gate insulation thickness, as feature sizes are made smaller, the capacitance tends to increase as the thickness of the gate insulation decreases. Since the insulation under the gates is thinner than under the wires, the gates contribute a disproportionate amount to the total capacitance.

Consider a gate that is 65 nm square and has an oxide thickness of 1.2 nm (realistic figures for about 2008):

| | |
|-------------------------|--------------------------------------|
| gate dimensions | 65×10^{-9} m |
| gate area | 4.2×10^{-15} m ² |
| distance between plates | 1.2×10^{-9} m |
| capacitance | 1.2×10^{-16} F |

The charge on the gate will be the product of the working voltage and the capacitance $= vC$, and the power used will be $P = fv^2C$, where f is the clock frequency. If we operated the gate at 3.3 V and had a clock speed of 1 GHz, then each gate would use a power of about 1.3×10^{-6} W (watts). This does not seem much until you take into account how many gates there are in a modern chip. An AMD K10, for example, has of the order of 5×10^8 gates. The power used will depend on the fraction of the gates that switch each cycle; if 10% switch each cycle, the total power consumption of the chip would be about 65 W. Clearly, the above model is very approximate. The fraction of gates switching each cycle is a guess, and we have made the simplifying assumption that gates are squares of edge the minimum feature size. We have ignored capacitance on wires, since the thickness of insulation here is much larger, and we have also neglected resistive losses.

Despite the simplifications, our estimated power consumption agrees reasonably well with the figures given by AMD for the Agena version of the K10, which consumes between 65 and 140 W for clock speeds between 1.8 GHz and 2.6 GHz.

As clock speeds increase, so does power consumption. As feature sizes shrink, and thus the insulator thickness falls, this too increases power consumption per square centimetre of chip surface. The result is shown in Fig. 5.7. The processor becomes physically dominated by the equipment required to remove excess heat. The rate of heat flow away from the surface of the chip is already more than can be handled by simple conduction. To augment this, heat pipes (Dunn and Reay, 1973) have to be used (see Fig. 5.7(c)). We are approaching the limit of heat dissipation that can be practically handled using low-cost techniques. In the past, mainframe computers used liquid cooling (Cray, 1986), but this is expensive and not suitable for mass-produced machines.

Our account of computing has led from primitive arithmetical aids to devices more like little steam engine boilers. This seems strange.

But there turns out to be a deep relationship between computing and thermodynamics, and it is thermodynamics that, on the basis of current understanding, fundamentally limits the power of our computing.

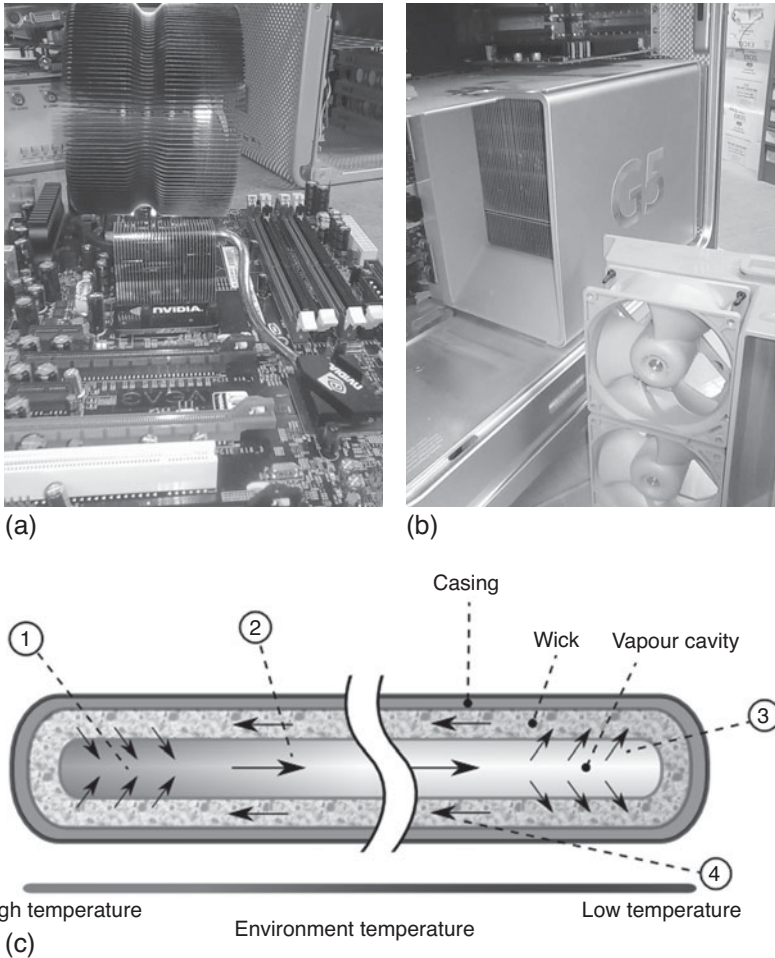


Fig. 5.7 Heat exchangers used in modern desktop computers. (a) The heat exchanger used for a 4 core Intel processor. The large finned heat exchanger is cooled by an axial fan and heat is led up to it from the processor and graphics chips by heat pipes. (b) The heat exchanger system used for the G5 processor on an Apple computer. (c) The principle of the heat pipe: 1, the vapour evaporates, absorbing heat; 2, the vapour flows to the cold end; 3, the vapour condenses, releasing heat; 4, the wick conducts the liquid back.

5.5 Entropy

Classical thermodynamics is a true experimental science and a foundation stone of classical physics. The theory was fully developed during the nineteenth century but its roots are earlier, in the study of *heat engines*, machines that harness heat to perform work and whose key role in the future of industrial civilization was already apparent by the early 1800s. Although the subject had extremely practical motivations, the principles uncovered by its founding fathers are so general that they can be applied to any finite macroscopic system and are now considered to constitute universal laws across this domain. These principles have been extensively confirmed through empirical confirmation of predictions.

Thermodynamics is concerned with macroscopic properties of matter and requires no hypotheses concerning microscopic structure. Indeed, the key to its universal applicability lies in the extraordinary simplicity and generality of the assumptions that it makes. However, the

corpuscular view of matter had become widely accepted by the late nineteenth century, and a number of scientists strove to develop a particle-based theory from which the key derived quantities and laws of thermodynamics could be deduced. The key protagonists of this enterprise were James Clerk Maxwell (1831–79) and, especially, Ludwig Boltzmann (1844–1906), who showed that consideration of the statistical behaviour of large numbers of particles, behaving probabilistically, could yield the same predictions as the high-level macroscopic treatment of thermodynamics. This probabilistic corpuscular theory came to be known as *statistical mechanics* (sometimes also called *statistical thermodynamics*). It was extended with enormous success in the twentieth century when, with the advent of quantum theory, a proper quantum-mechanical treatment could be applied to the statistical assemblies under consideration. With this refinement, the predictions of statistical mechanics have been verified for a wide range of systems, including those based on matter in extreme and exotic conditions.

Even in its classical form, statistical mechanics can give a considerable insight into the nature of thermodynamic properties and laws, but the philosophical status of this insight is not perhaps quite as simple as it first appears. It is often assumed that thermodynamics must be, in some sense, a less fundamental theory than statistical mechanics; but this is not strictly the case, as argued powerfully by, for example, Primas (1998). Such deep considerations aside, however, in what follows we will focus on the power of the classical thermodynamic theory, but use a corpuscular narrative to help anchor the ideas.

The base unit of study in classical thermodynamics is that of a *thermodynamic system*, any finite macroscopic region of the universe that is under study as a coherent whole for some reason. The qualifier ‘macroscopic’ may be interpreted to mean that all matter and energy within the system and entering or leaving it can be treated as continuous in nature. The universe other than the system is said to constitute the *surroundings* or *environment*, and the interface between system and environment is the *boundary*. Thermodynamics considers the ways in which energy can drive the behaviour of the system, whether or not that energy can be transferred to and from the environment. A system may be *simple*, consisting for example of a volume of a single substance, or *compound*, with several distinct and possibly interacting parts.

Any thermodynamic system is characterized by a number of *properties* that describe it. Some of these properties are measurable physical values such as mass, volume, or pressure, whereas others are derived quantities such as density. Some properties, such as pressure, field strength, or temperature, can vary from point to point within the system and are called *intensive*; others, such as volume or mass, are properties of the system as whole and are called *extensive*. If the surroundings of the system change, these properties will also change over time, until eventually no further change occurs. At this point, the system is said to be in *thermodynamic equilibrium*. When a system is in equilibrium, it can be characterized by specifying fixed values for all its properties and it is

then said to be in a definite *thermodynamic state*. If after being slightly displaced the system will return to its equilibrium state, the equilibrium is said to be *stable*; if it is stable for only very small displacements but unstable for larger ones, it is said to be *meta-stable*. Finally, a system that can be displaced by a small disturbance, but will remain in the displaced state when released, is in *neutral equilibrium*. Systems in equilibrium are of central importance in thermodynamics, but much that is of interest also concerns systems that are not in equilibrium.

One of the simplest systems, often used to illustrate thermodynamic ideas, is a fixed mass of a so-called ‘ideal’ gas held at equilibrium in a known volume, V , at some fixed pressure, p , and absolute temperature, T . The reason for the popularity of this particular system is because its behaviour is well understood and summarized in the *ideal gas law*:

$$pV = nRT \quad (5.1)$$

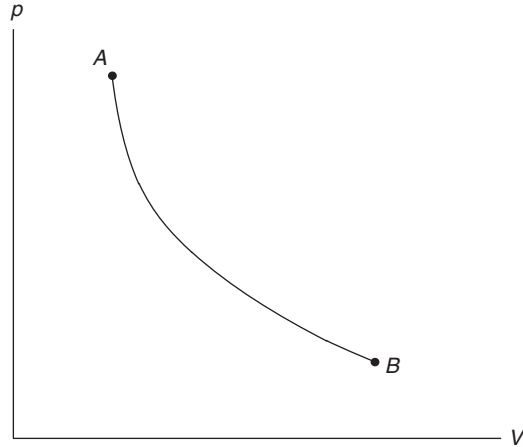
where n is the amount of the substance in moles and R is the *gas constant* ($\sim 8.314 \text{ J K}^{-1} \text{ mol}^{-1}$).

The properties of a system in equilibrium are typically subject to relationships that place constraints on them. For example, if the mass of the system is fixed at one mole ($n = 1$) of the constituent gas, the volume, pressure, and temperature measured will always obey the ideal gas law (Eq. 5.1), so that if any two of the quantities p , V , and T are known, the third is automatically determined. The law is an example of what is called an *equation of state* for the system and it shows that if a minimum number of properties are known, the others are all determined. This number of independent parameters required to describe a system is a characteristic of that system and is referred to as the number of *degrees of freedom* that it possesses. The properties used to describe the system state are sometimes called *thermodynamic coordinates*.

In the case of the ideal gas, the system has two degrees of freedom. The state of the system can be fully described by specifying, say, its volume and pressure as thermodynamic coordinates (although volume and temperature, or temperature and pressure, would do equally well). It is then possible to represent the state of the system as a point on a two-dimensional coordinate plane with axes labelled p and V (or p and T , or V and T). A graph of this kind is called an *indicator diagram* for the system (see Fig. 5.8).

It is important to recognize that the concept of state only applies to a system at equilibrium, and only such a system can be represented as a point on an indicator diagram. A system in an initial state of equilibrium will not undergo further change unless it is perturbed. When this happens, however, the system will typically change, resulting in it eventually moving to a new equilibrium state. The change can be partially described by showing the initial and final state as two points on an indicator diagram but, while this details the start and end of the change, it does not say anything about what happens in the middle.

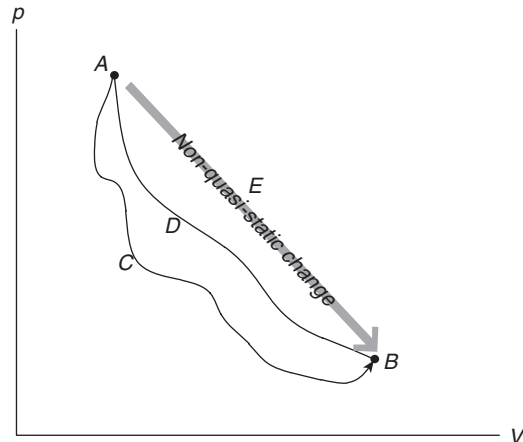
Fig. 5.8 The state of a thermodynamic system is represented by a point on a pressure–volume (p – V) indicator diagram. A quasi-static change from an initial state, A, to a final state, B, is then a continuous path joining the two points.



In certain circumstances, if a change occurs slowly enough, the system may be thought of as being close to a definite state at every instant between its initial and final positions. In this case, the system's progress between initial and final states can be plotted as a path on the indicator diagram, with each point on the path denoting an intermediate state through which the system has passed on the way. A change of this type is called *quasi-static*. Between any two points, many paths are possible, so a system in a certain initial state may follow many different quasi-static paths to the same final state (see Fig. 5.9). Often, however, a system will alter state so quickly that some of its intensive properties have no defined values during the change.

For example, during an explosion, pressure may vary so widely across a system that was previously in equilibrium that no meaningful value can be assigned to the property. In this case, while the system is between its initial and final states, it cannot be represented on an indicator diagram and no path can be assigned to the change of state.

Fig. 5.9 All changes from state A to state B are associated with the same change in internal energy, $\Delta U = U_B - U_A$, regardless of mechanism. This applies not only to all quasi-static paths (e.g. C and D) but also non-quasi-static changes with no indicator diagram path, such as E.



For a thermodynamic system in equilibrium to undergo change, energy must either be delivered to it or taken from it by its surroundings. The concept of energy is extremely fundamental in physics. Without attempting a general definition, we will note that a system may possess energy by virtue of its macroscopic motion (*kinetic energy*) or position in an external field of force (*potential energy*) but also internally, stored in its physical structure. Here, we will ignore the kinetic and potential energy and address only that stored internally in the system. In terms of thermodynamics, it is relatively straightforward to show that a system's *internal energy*, U , is a function of its state. Thus, for example, for a system consisting of a fixed mass of gas at equilibrium with volume, V , and at pressure, p , we can write: $U = U(p, V)$.

In general, when a system moves from an initial state— A , say—to a final state, B , its internal energy changes by a fixed amount: $\Delta U = U_B - U_A$ and this is the same regardless of how the change of state occurs. Whether it is quasi-static or not and, if so, whatever path it follows, the internal energy difference between initial and final state is always the same. Indeed, it is a fundamental principle of physics that energy in such scenarios is conserved, so internal energy gained or lost comes from or goes to somewhere. Internal energy can be gained either by performing work on the system or by transferring heat into it from outside. Similarly, internal energy can be lost by having the system perform work on its surroundings or by allowing it to deliver heat to them. From this, it should be clear that the terms 'work' and 'heat' are both descriptions of energy in transit and are not, in any way, properties of any system's state.

Given this insight, the difference between energy in the form of work and in the form of heat is interesting to dwell on for a moment. It is most easily understood if we look at the system from a microscopic point of view. From this viewpoint, the internal energy of a system is just the sum of all the energies of the individual particles within it. When energy is delivered to the system in the form of work, the delivery is 'ordered', for example, by a wall of the system (say, a piston) moving to compress it. This ordered delivery imparts a non-random motion component to all system particles encountering the wall. When energy is delivered in the form of heat, on the other hand, the particles in the wall move randomly (or *thermally*) and increase the energy of the system particles in a similarly random fashion. A transfer of energy will only happen in this latter circumstance if the average random or thermal energy of the surroundings is higher than that of the system. However, the distinction is not always clear cut: both work done on a system and heat entering it will cause its internal energy to increase.

A key question posed by thermodynamics concerns the extent to which an increase in a system's internal energy can then be reversed by the system doing work on or giving heat to its surroundings, and whether such a reversal is associated with a penalty of any kind. A change in a system is said to be *reversible* if and only if both the system and the environment can be restored to their respective conditions before the change.

The concept of reversibility is an idealization and approximating to it relies on allowing changes to occur very slowly, so that the system is always close to a state of equilibrium. For reasons that will become clearer later, a change can only be reversible if its direction can be reversed by an infinitesimal change in the conditions of the surroundings. All reversible changes are necessarily quasi-static, but some quasi-static changes are not reversible. For example, changes involving dissipative work, as discussed below, can sometimes be quasi-static but are never reversible.

When two systems (or a system and its surroundings) are placed in contact, heat energy will flow from the system whose particles have higher average thermal energy to the other system. This will occur until the systems reach a new equilibrium where the average thermal energy is the same in both. The systems are then said to be in *thermal equilibrium*. From observation, it is found that if we have three systems, A , B , and C , then if A is in thermal equilibrium with B and also in thermal equilibrium with C , then B and C are in thermal equilibrium with each other. This universal observation is now known as *The Zeroth Law of Thermodynamics*. It is this law that allows the concept of temperature to be put on a firm footing. Essentially, all systems in thermal equilibrium are at the same temperature. A standard system with some thermometric property (for example, the volume of a liquid) that changes with temperature can then be used as a *thermometer*. In microscopic terms, temperature can be interpreted as the average thermal energy of the particles of which a system is made up.

The most straightforward way in which work can be done on a system is to supply energy via some change in the external forces acting on it so as to alter one of its extensive properties. A simple example of this is the compression of a gas, which—by definition—reduces its volume (Fig. 5.10a). Work of this type is called *configurative* and has the property that, if applied sufficiently slowly, the system can be kept quasi-statically on a path on its indicator diagram. Assuming that all energy supplied goes into the configurative change, the system and the environment can be restored to their original states by reversing the path and allowing the system to do work. So, for example, by reducing the pressure slowly, the compressed gas can be allowed to expand again, so doing work on the surroundings. However, not all work is configurative. Sometimes when work is done on a system no change in configuration occurs. An example of this is where work done on a system is dissipated via friction; another where external electrical energy is supplied to a resistance within the system (Fig. 5.10b). As is well known, the effect of such *dissipative work* is to raise the internal energy of the system in exactly the same way as if energy had been delivered to the system as heat. A change due to dissipative work is not reversible for reasons that we shall see shortly.

Heat is exchanged between a system and its surroundings when there is a temperature difference between them, but only if the boundary between the systems is *diathermal*—in other words, it allows heat to pass. All real boundaries allow some heat to pass, but we can observe

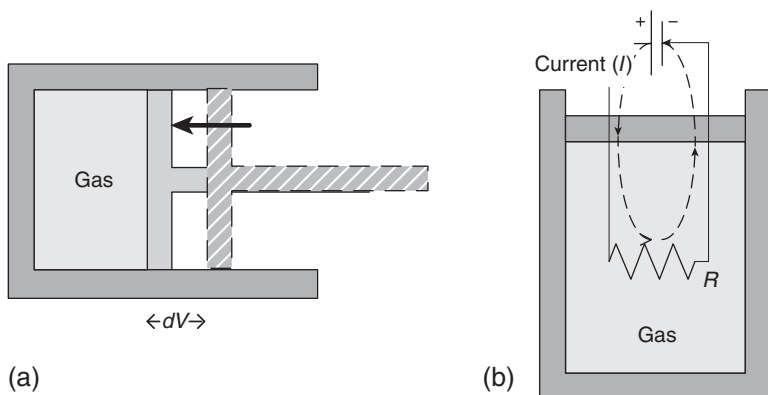


Fig. 5.10 (a) When a gas is compressed without frictional losses, all the work goes into reversibly reducing the volume. This is an example of configurative work. (b) When a current, I , is passed through a resistance, R , the work done on the system, I^2R , is dissipative and irreversible. If there is no change in volume, the temperature and pressure will increase.

that certain types greatly inhibit it and we could conceive of an idealized insulating boundary that would be impervious to heat: such a boundary and a system isolated by it are said to be *adiabatic*.

An adiabatic system cannot exchange heat with its surroundings and can only change its internal energy through work. In such a system, it is observed that the amount of work required to change the system from an initial state, A , to a final state, B , is always the same, regardless of how the change is made. This is the most basic form of the famous *First Law of Thermodynamics*. If the adiabatic constraint is relaxed, then any change in internal energy is the sum of the work, W , and the heat, Q , transferred between the system and its surroundings. Conventionally, we take work done on the system and heat transferred into it to be positive, while work done by the system and heat transferred out of it are negative. The First Law then takes the form:¹

$$\Delta U = Q + W \quad (5.2)$$

For an infinitesimal change, the law is expressed thus:

$$dU = d'Q + d'W \quad (5.3)$$

It is important to note that, for any given change, although the difference in internal energy is fixed, the values of Q and W depend on the path taken, and not only on the endpoints. This is entirely in line with the remarks above about energy transfer: Q and W are not functions of state, a fact recognized explicitly by marking the differentials with a d' to show that they cannot be integrated except along a known path. Differentials of this type are called *inexact*.

In essence, the First Law says that in any change of state in a thermodynamic system, energy is always conserved. This is not surprising, but it is only part of the story. Many changes that the First Law would allow never actually occur. For example, if two systems at different temperatures are placed in thermal contact, heat will flow from the hotter to the colder until the temperature equalizes. However, heat is never observed to flow from a colder to a hotter body or even between

¹In fact, logically this is just the definition of heat. The First Law is actually the above statement about work in adiabatic systems. However, the equation here is often loosely also referred to as the First Law and we shall follow this convention.

two systems at the same temperature. Yet such transfers do not violate the First Law.

The physicists who systematized thermodynamics wanted to unearth a principle that might explain why some thermodynamic transfers allowed by the First Law never occur and to do this they turned to the subject that motivated thermodynamics in the first place: the study of engines. An engine is a system capable of doing useful work; however, in order to do useful work, it must cycle from an initial state of higher internal energy— A , say—to one of lower energy and back to A again. If an engine does not execute a cycle, it cannot keep delivering work, but to return to its initial state it must be fed energy from its surroundings. The engines of interest to the founders of thermodynamics were *heat engines*, characterized by using a flow of heat from the surroundings to supply this required energy. Other options are possible, of course: an engine might be driven by an external source of electrical energy, for example, but heat engines are the starting point and it turns out that the principles discovered from their study are universally applicable.

The key generalization that emerged concerning which changes are allowable and which are not, is *The Second Law of Thermodynamics*. The first formulation of the Second Law, in 1850, is attributed to Rudolf Clausius (1822–88), a German physicist and mathematician who was one of the most important figures in the development of the subject. The Clausius statement of the Second Law says simply that no process is possible that has the sole result of transferring heat from a colder to a hotter body.

Shortly afterwards, William Thomson (1824–1907), who became Lord Kelvin in 1892, showed that this statement is entirely equivalent to asserting that no process is possible by which heat can be completely converted into work. This is now sometimes called the *Kelvin–Planck statement* of the Second Law. If an engine takes in heat and does work, some heat will be wasted and must be returned to the surroundings. This means that the engine needs some part of the surroundings to be hotter than it is (a so-called *hot reservoir*) and another part to be colder (a *cold reservoir*). If the Clausius statement were false, such an engine could be accompanied by another process that simply took the waste heat from the cold reservoir and restored it to the hot at no cost whatsoever. Similarly, it can also be shown that if the Kelvin–Planck statement were false, the Clausius version would also fail. To see this, imagine an ‘anti-Kelvin’ engine driven from a hot reservoir, driving a refrigerator that uses the work delivered by the engine to move heat from the cold reservoir to the hot, thus violating the Clausius statement. In short, the two formulations, although distinct at first sight, are in fact expressing exactly the same physical principle.

If an engine has a cycle consisting of reversible changes throughout, the engine itself is said to be reversible. Suppose that an engine, whether reversible or not, operates between two reservoirs. It absorbs a quantity of heat Q_1 from the hot reservoir at temperature T_1 , does work, W , and

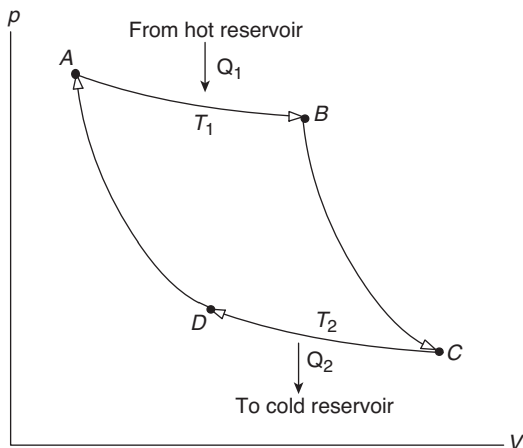


Fig. 5.11 In order to do useful work, an engine must cycle from an initial state of higher energy to one of lower energy and back again. In the *Carnot engine*, all paths are reversible. During AB (isothermal expansion), the engine takes in heat Q_1 at a constant temperature T_1 but, by expanding, it does some work on its environment. During BC (adiabatic expansion), it purely delivers work but cools. During CD (isothermal compression), it rejects heat Q_2 to its cold reservoir at temperature T_2 and also has some work done on it. During DA (adiabatic compression), work is done on it to return it to its original state, heating it up in the process.

then rejects heat Q_2 to the cold reservoir at temperature T_2 (Fig. 5.11). The *efficiency* of the engine is defined in the following way:

$$\eta = \frac{W}{Q_1} = \frac{Q_1 - Q_2}{Q_1} \quad (5.4)$$

In 1824, another of the key figures in the history of thermodynamics, Sadi Carnot (1796–1832), showed that all reversible engines operating between two reservoirs are of equal efficiency, and that no non-reversible engine can attain this efficiency. If all reversible engines operating between the same two reservoirs have the same efficiency, this (maximal) value must be a function of the reservoirs themselves and so of their only defining parameter, their temperatures. Lord Kelvin showed that one can then define an *absolute temperature scale*, based upon which, for any reversible engine:

$$\frac{Q_{r2}}{Q_{r1}} = \frac{T_2}{T_1} \quad (5.5)$$

where Q_{r1} is the heat reversibly absorbed by the engine from the hot reservoir at temperature T_1 and Q_{r2} is the heat reversibly rejected to the cold reservoir at temperature T_2 . For a general (non-reversible) engine, since the efficiency must be less than or equal to its reversible equivalent between the same two reservoirs, we must have

$$\frac{Q_1}{T_1} \leq \frac{Q_2}{T_2} \quad (5.6)$$

If we now take heat entering a system (e.g. Q_1) as positive and heat leaving it as negative, it is easy to show directly from the Second Law (see, e.g., [Adkins, 1983]) that this result can be generalized around a cycle where heat is exchanged with the surroundings at arbitrarily varying temperatures:

$$\oint \frac{dQ}{T} \leq 0 \quad (5.7)$$

with equality holding if and only if the cycle is reversible. In other words if, for each infinitesimal step around the cycle, we sum the amount of

heat exchanged with the surroundings (+ve or -ve) divided by the temperature at which the exchange takes place, the result will be less than or equal to zero. If the path is reversible, equality will definitely hold. Note that in a reversible change, the temperature of the system and the surroundings are always approximately equal; if this were not the case, changes to the system would be irreversible since no infinitesimal change in the surroundings could reverse the direction. In the irreversible case, T in Eq. 5.7 is the temperature at which heat is transferred across the system boundary. The result (Eq. 5.7) is now known as Clausius' Theorem.

Clausius now defined a new quantity that he associated with any reversible change and that he called *entropy*, usually denoted by the letter S . For any infinitesimal reversible change that involves a heat transfer $d'Q_r$ at temperature T , the entropy of the system changes by an amount

$$dS = \frac{d'Q_r}{T} \quad (5.8)$$

It immediately follows from Clausius' Theorem that, in completing any reversible cycle, entropy does not change and so, like internal energy, it must be a property of state. A system in a given state always has the same entropy, just as it always has the same internal energy, and the change in entropy associated with a given change of state is the same regardless of how the change occurs. This is an interesting mathematical point. In moving a system reversibly between two states, if we sum the heat exchanges between it and its environment, the final answer depends entirely on the path followed; however, if each element of heat is divided by the temperature at which it is transferred, the sum is always the same, independent of path. The differential form of the First Law can be usefully recast for reversible changes where work is configurative and heat transfer reversible. This is often stated in a form where the configurative work is achieved by using pressure to change volume reversibly, so that we have $d'W = -pdV$ (the minus sign is because doing work on the system corresponds to decreasing its volume). Substituting in the differential form of the First Law:

$$dU = TdS - pdV \quad (5.9)$$

This casts the law in a form where all the terms are composed of functions of state or state variables. The $p dV$ term can be generalized to include a sum of any intensive-extensive pair of quantities relevant to a system which together perform configurative work (for example, a force F , extending an elastic rod by length dL). That aside, from the form of the equation, it is clear that entropy is an extensive variable like volume or internal energy. Thus if two systems are taken together, the entropy of the composite system is just the sum of the entropies of the components.

Turning now to changes that are not reversible, from Clausius' Theorem, since the quantity is less than that in an equivalent reversible

change, it follows that

$$dS > \frac{d'Q}{T} \quad (5.10)$$

Now consider an adiabatic system. Here, $d'Q$ is always zero by definition, so for any change in such a system we must have

$$\Delta S \geq 0 \quad (5.11)$$

This is sometimes called the *Law of the Increase of Entropy*. It is sometimes stated simply as follows: ‘the entropy of an isolated system cannot decrease’. In an isolated system in a given equilibrium state, it is only possible for changes to occur that carry the system to states of equal or higher entropy. Irreversible changes are those that increase entropy. Changes that reduce entropy do not occur in an isolated system. It is, incidentally, possible to show that the earlier statements of the Second Law can be proven from this assertion and so it is itself an alternative formulation of the latter.

When a system is not isolated, it is possible for its entropy to decrease, but only if this is accompanied by a greater or equal increase in the entropy of the environment. So, for example, when dissipative work is done on a system the system can be returned to its original state by transferring heat to the environment, but this heat raises the entropy of the environment and so the universe as a whole cannot be returned to its previous state. Note that it is possible for entropy to increase with no transfer of heat or work into the system if there is a state of equal internal energy that happens to possess higher entropy than the initial one.

As defined, the quantities ‘internal energy’ and ‘entropy’ always appear as differences between states. It has been observed that they are functions of state variables, but no explicit form of these functions has been given. In fact, it is possible to deduce such forms, but they do vary for different types of system. The simplest system to analyse is again the familiar one consisting of a fixed mass (say, one mole) of an ideal gas. In this case, it is easy to show that

$$U = c_v T + c_1 \quad (5.12)$$

and

$$S = c_v \ln T + R \ln V + c_2 \quad (5.13)$$

where c_v is the *heat capacity* of a mole at constant volume, R is the *universal gas constant* and c_1 , c_2 are arbitrary constants that can be fixed at an appropriate origin. Of course, the concept of an ideal gas is an abstract conceptualization, but real gases at low pressures can approximate to the behaviour. However, at low temperatures no real substance behaves like an ideal gas and the formulae given here break down: thus although the second equation predicts that at $T = 0$, entropy

would be $-\infty$, this is absurd. The breakdown of classical idealizations at low temperatures is due to the fact that quantum effects begin to dominate close to absolute zero. However, at temperatures where the approximation to real behaviour is tenable, it can be shown that c_v for an ideal gas is constant. Thus the entropy of an ideal gas does depend logarithmically on temperature, exactly as the formula suggests.

Mathematically speaking, thermodynamic entropy is a well-defined and precise concept, but it often presents intuitive difficulties to new students of the subject. It is here that a microscopic interpretation is valuable. In the statistical mechanical picture, the parameters that classical thermodynamics uses to describe a system are bulk manifestations of the properties of enormous numbers of individual particles averaged out. To get a sense of just how big the numbers are, we need only note that the count of molecules in a gram mole of any substance (say, 12 g of carbon) is given by the *Avogadro constant*, which is about 6×10^{23} . Since the particles in such a microscopic picture would be quantum in nature, the determination of their properties is a quantum *N-body problem*, which is quite intractable. Given the number of entities involved, any solution must rely on statistical argument.

Assemblies of quantum particles are well understood. In a mass of a single substance, the constituent particles are indistinguishable in a fundamental quantum-mechanical sense that has no analogue in classical physics. Such identical particles do not, in fact, have an entirely independent existence, but are entangled in a quantum superposition. What this means is that the properties of individual particles in a quantum assembly are, in general, undefined, in complete contrast to the equivalent classical picture. However, it can be shown that if we can approximate the assembly by assuming no direct interaction between the particles (an independence premise), individual particle properties reappear, at least in the sense that it is logically consistent to talk about them.

If a system is thermally isolated (adiabatic), then so long as work is not done on it or by it, it will have a fixed internal energy and may be in equilibrium at a different temperature to its surroundings. If, on the other hand, the system is in thermal contact with its environment, it is the internal energy of the system plus that of the environment that is fixed. The environment is usually modelled as being sufficiently extensive that its temperature is not significantly affected even when heat is exchanged with the system, and is traditionally referred to as a *heat bath*: at equilibrium, the system must be at the same temperature as the bath in which it is immersed. In statistical mechanics, an adiabatic system is said to exhibit *microcanonical* behaviour, while a system in isothermal contact with a heat bath exhibits *canonical* behaviour. In both situations, the number of particles in the system is fixed: however, if this constraint is relaxed and both heat and particles may be exchanged with the environment, the system is described as *grand canonical*.

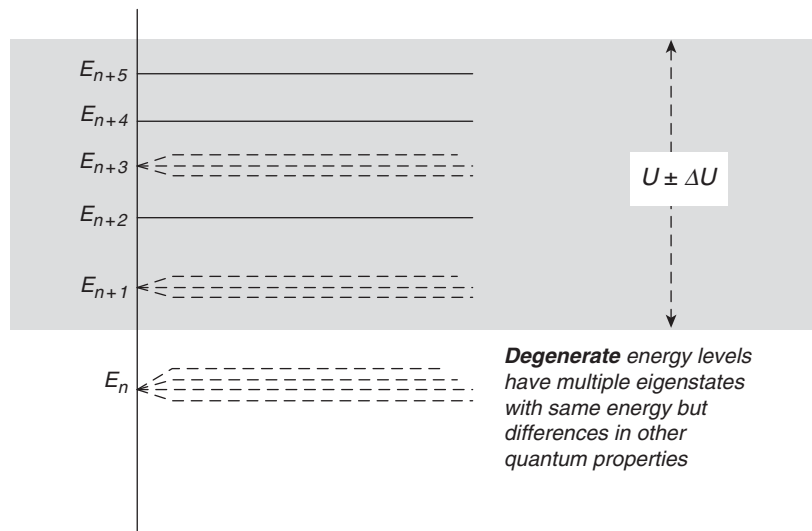


Fig. 5.12 A multi-body quantum system has energy levels for the system as a whole. In principle, the system may be in one such state if its energy has been accurately determined; in practice, it will be possible only to measure the mean energy with some accuracy ΔU and the state will be a quantum mixture of energy states within that range.

To take the simplest example, consider an isolated system (micro-canonical) with fixed internal energy— U , say—measured to accuracy ΔU . If we were able to solve the Schrödinger equation for the entire system, we would recover the so-called *stationary states* of the system, $\Psi_{n\{j\}}$ and their associated energy *levels* E_n (Fig. 5.12). A given energy level may be common to many different stationary states that differ in some other quantum properties, as indicated by the index j . When there are multiple states with the same energy level, the level is said to be *degenerate*. In principle, when a (quantum) energy measurement is conducted on the system as postulated, the result should be one of the E_n , and the exact *microstate* of the system (one of the $\Psi_{n\{j\}}$) is then determined. Further measurements of other compatible properties may be necessary to settle the degeneracy.

In practice, however, such precise determination is impossible. The reason is that the energy levels of any macroscopic system are very densely packed, with a separation that can be shown to decrease exponentially with the number of particles. As a result, in any realistic system, random fluctuations in energy caused by interaction with the environment will be much larger than the distance between neighbouring energy levels and consequently the system will never remain in one energy microstate for long. This is true even if the system is thermodynamically isolated, because no macroscopic system can be isolated at the quantum level. Consequently, any macroscopic energy measure will always be no more than a mean value with accuracy much coarser than the inter-level spacing. For example, if it is determined that the total energy lies in the range, the system may occupy any microstate with an energy in this range and will continually move between such microstates.

Once we have an energy value for the system, we may make further macroscopic measurements and thus determine a macroscopic configuration or *macrostate*. We may elect to describe the macrostate

with parameters and associated accuracies, determinable via macroscopic measurement, but not included in the classical thermodynamic set—such as, for example, the number of particles in the system. There will typically be enormously many microstates compatible with any given macrostate, but we cannot ever practically determine which one the system might occupy at a given instant of time. The number of microstates corresponding to a given macrostate is called the *weight* of the macrostate.

In quantum theory, a system that is not well enough described to allow its *pure* time-dependent quantum state to be determined is said to be in a *mixed* state, and this concept is at the core of statistical mechanics. The technical approach is to allocate classical probabilities to the different microstates that the system could possibly be in (see, e.g., Isham, 1995). A mixture is quite different from a quantum superposition of states: a system in a quantum superposition of several states is in a distinct state different from all the constituents; a system in a mixed state, however, is always in one or other of its constituents and moves between them randomly according to some probability distribution. To say that a system is in a mixed state is to acknowledge that there is not sufficient information to allow a pure state to be determined. It can be shown that a thermodynamic system at equilibrium in the mixed state can be characterized as consisting of *stationary states* alone and, consequently, the system can be characterized as moving continuously between microstates of this type. In principle, a probability distribution can be defined across these microstates but there is, at first sight, insufficient information to allow such a distribution to be determined.

The way around this apparent impasse is to postulate that all stationary microstates are occupied with equal likelihood, so that the probability of a system being in a given macrostate is proportional to its weight. In general, for macroscopic systems, the maxima of the weight function are very strongly peaked and it is these peaks that define the equilibrium states. A state of a system is a state of equilibrium, when there are so many microstates corresponding to it that the system will never stray far from it for long. Thus a macrostate has high entropy simply because it has a high weight of microstates. Boltzmann was able to establish that the entropy of a system is directly proportional to the logarithm of the weight, W , of the equilibrium state:

$$S = k \ln W \tag{5.14}$$

The constant of proportionality, $k = 1.38 \times 10^{-23} \text{ J K}^{-1}$, is the famous *Boltzmann's constant*.

If all microstates are assumed to be equally probable, working out probability distributions for macrostates reduces to a counting exercise. To count stationary microstates, we make another approximation and assume that the individual particles in the system are effectively independent. In this case, it can be shown that the microstates of the system as a whole are products of the energy eigenstates of the N individual particles. We can thus enumerate the microstates by counting

the number of ways in which N particles can occupy allowable *single particle energy levels* so as to give a total energy sum in the required range. This greatly simplifies matters, since single-particle stationary states can be obtained by solving a single-particle Schrödinger equation, a much more tractable task than tackling an N -body problem

A particle can be thought of as having energy by virtue of its velocity (dominant in gases), its rotation, or its vibration, but the principle is the same in each case. The actual values of the particle's allowable energy levels are found by solving the Schrödinger equation for it under suitable boundary conditions. For example, for a (non-rotating, non-vibrating) particle of mass m moving freely in a cubical box of volume V , it is easy to show (Sears, 1975) that the energy levels are given by

$$E_j = (n_x^2 + n_y^2 + n_z^2) \frac{h^2 V^{-\frac{2}{3}}}{8m} \quad (5.15)$$

where n_x , n_y , and n_z can take any positive integer values, representing the quantized momentum along each of the coordinate axes. Note first that the energy levels are proportional to $V^{-2/3}$ so that, if the volume of the system increases, the energy levels decrease in value. Each triplet (n_x, n_y, n_z) defines a different state and we can quickly see that many energy levels are degenerate: for example, the states defined by $(1, 1, 2)$, $(1, 2, 1)$, and $(2, 1, 1)$ all have the same value of E . It is easy to verify that there is only one state for the lowest energy level, E_1 , (the *ground state*), but there are three states with E_2 , and so on. Subject to the total amount of energy available, individual particles may occupy any set of single-particle states at any energy level and a system microstate can be specified simply by indicating which single-particle state each component particle is in. Any such microstate must, of course, generate the correct macroscopic measurements for the system: in particular, the occupation numbers, n_s , of the various energy levels, E_s , must be such that

$$U - \Delta U < \sum_s E_s n_s < U + \Delta U \quad (5.16)$$

By counting in this way, we can enumerate, in principle at least, the number of microstates compatible with the macrostate. Where multiple macrostates are possible, then, as observed above, the macrostate with the maximal weight is the one in which the system is most likely to be found. Further, since the most probable macrostates are overwhelmingly more probable than alternatives, these are the states of maximum entropy and the ones towards which the system inevitably moves—and, once there, stays. They are, in short, the equilibrium states of the system.

All this gives considerable insight into what the 'physical meaning' of Clausius' entropy really is. For example, when the temperature of a gas is raised, this increases its internal energy and, in so doing, raises the average energy of the particles. This allows access to higher energy levels and increases the number of microstates available, consequently raising the entropy. If, on the other hand, the volume is increased while holding U fixed, the available single-particle states reduce in energy level

and get closer together, allowing particles to access more levels and thus increasing the number of ways in which U can be made up. With the greater number of microstates available, entropy again rises.

It is sometimes said that entropy is a measure of disorder. What this means is that in high-entropy states there are very many microstates that are indistinguishable from a macroscopic point of view, and the macroscopic description gives little clue about which microstate is really in play at any given time, because there is a large amount of missing information. In a low-entropy state, the macroscopic description is much closer to the microscopic one. In the extreme example where every particle is in its ground state and there is only one such state, the system is entirely specified by the macrostate and the entropy is zero.

Another way of thinking about entropy is that it measures the extent to which a system is close to equilibrium. Equilibrium occurs at a local maximum of entropy and, once a system is in stable equilibrium, it requires external energy to displace it. A system in equilibrium is, by itself, incapable of converting energy to work; for this, a composite system is required that is not in equilibrium and that allows energy to flow between its parts. However, the composite system will itself attempt to reach equilibrium as heat flows from hotter to colder parts and so its entropy will increase. The best that can be done is to use this flow of energy to drive a process that does useful work, but the most that any such process can achieve, and then only if it is reversible, is to prevent entropy from increasing. In reality, no process is genuinely reversible as there are always dissipative losses, and so the inexorable approach to equilibrium and maximum entropy cannot be staved off. When its entropy does finally reach a local maximum, the composite system itself is no longer capable of doing useful work and, at equilibrium, is entirely resistant to change.

To close this section, we will rewrite Boltzmann's relation (Eq. 5.14) reasoning as follows. Suppose that there are N (distinguishable) particles in a system of total energy E and that the system can be analysed according to the occupancy of its single-particle states. Suppose further that the single-particle energy levels are given by E_i with $i \approx 1, \dots, m$ and that each energy level contains n_i particles. For simplicity, we also assume that the energy levels are non-degenerate. Clearly,

$$N = \sum_{i=1}^n n_i \quad (5.17)$$

and

$$E = \sum_{i=1}^m n_i E_i \quad (5.18)$$

The occupancy of the system energy levels can be taken to define the system macrostate. A microstate, on the other hand, must specify exactly which particles are at which energy level. From a well-known result in combinatorics, the number of possible arrangements of the N

particles with n_i of them at level E_i is given by

$$W = \frac{N!}{n_1!n_2!\dots n_m!} \quad (5.19)$$

which can be slotted into Eq. 5.14:

$$S = k \ln \left(\frac{N!}{n_1!n_2!\dots n_m!} \right) \quad (5.20)$$

So long as N is very large, we can use Stirling's formula for approximating the factorials of large numbers, and rewrite this as follows:

$$S = -kN \sum_{i=1}^m \frac{n_i}{N} \log \frac{n_i}{N} \quad (5.21)$$

Now observe that the probability that a particle has energy E_i is just

$$p_i = \frac{n_i}{N} \quad (5.22)$$

so Eq. 5.21 becomes

$$S = -kN \sum_{i=1}^m p_i \log p_i \quad (5.23)$$

This relates the Clausius conception of entropy, through Boltzmann, to Shannon's *information theory*, which is discussed in the following section. In fact, in information terms, Eq. 5.23 is, except for a constant, the missing information that would be required to specify exactly the microstate configuration of the system, given only the macrostate information. Although the example chosen here is simplified, the idea can be generalized to any physical system at equilibrium. Thus Clausius entropy is intimately associated with missing information in a strictly quantifiable sense.

In information theory, the concept of entropy is generalized to mean the average information associated with any probability distribution. It is important to be clear that the two concepts are categorically distinct. Shannon's information entropy is a mathematical property of any probability distribution. Clausius entropy, on the other hand, is a physical property of a thermodynamic system, defined only in equilibrium states. Nonetheless, the Shannon definition, which is essentially the same as Eq. 5.23, can be used to extend the classical thermodynamic conception of entropy to any system where a probability distribution can be defined across microstates, *whether or not that system is in thermodynamic equilibrium*. This will not, however, be pursued further here.

5.6 Shannon's information theory

The establishment of information theory as a science occurred in the middle of the last century and is closely associated with the name of

Claude Shannon. If anyone was father to the information revolution, it was him. Shannon's revolution came from asking new questions, and asking them in a very practical engineering context. Shannon was a telephone engineer working for Bell Laboratories, and he was concerned with determining the capacity of a telephone or telegraph line to transmit information. He formalized the concept of information through trying to measure the efficiency of communications equipment (Shannon, 1948).

To measure the transmission of information over a telephone line, some definite unit of measurement is needed, otherwise the capacity of lines of different quality cannot be meaningfully compared. We need to quantify information. According to Shannon, the information content of a message is a function of how surprised we are by it. The less probable a message, the more information it contains.

Suppose that each morning the news told us

There has been no major earthquake in the last 24 hours.

We would soon get fed up. That conveys almost no information. If instead we are told:

We have just heard that Seattle has been devastated by a magnitude 8 earthquake.

this is real news. It is surprising. It is unusual. It is information.

A daily bulletin telling us whether or not a quake had happened would usually tell us nothing, then one day it would give us some useful information. Leaving aside the details, if an announcement were to be made each morning, there would two possible messages

0 'No big earthquake.'

1 'There has been a big quake.'

If such messages were being sent by wire, we could encode them as the presence or absence of a short electrical pulse, as a binary digit or 'bit' in the widely understood sense of the word. We normally assume that a bit is the information required to discriminate between *two possible alternatives*.

Shannon defines a bit more subtly, as the amount of information required for the receiver of the message to decide between *two equally probable alternatives*.

For example, a sequence of tosses of a fair coin do contain one bit per toss, and can be efficiently encoded so that heads are 1 and tails 0.

Shannon's theorem says that if we are sending a stream of 0 or 1 messages affirming or denying some proposition, then unless the truth and falsity of the proposition are equally likely, these 0s and 1s contain less than one bit of information each—in which case, there will be a more economical way of sending the messages. The trick is to use a system where the more probable message-contents gets a shorter code.

For example, suppose that the messages are the answer to a question that we know, *a priori*, will be true one time in every three messages.

Table 5.1 A possible code for transmitting messages that are true $\frac{1}{3}$ of the time.

| Binary code | Length | Meaning | Probability |
|-------------|--------|--------------|---------------|
| 0 | 1 | False, False | $\frac{4}{9}$ |
| 10 | 2 | False, True | $\frac{2}{9}$ |
| 110 | 3 | True, False | $\frac{2}{9}$ |
| 111 | 3 | True, True | $\frac{1}{9}$ |

Since the two possibilities are not equally likely, Shannon says that there will be a more efficient way of encoding the stream of messages than simply sending a 0 if the answer is false and a 1 if the answer is true. Consider the code shown in Table 5.1.

Instead of sending each message individually, we package the messages into pairs, and use between one and three binary digits to encode the four possible pairs of messages.

The shortest code goes to the most probable message, the sequence *False False* with probability $\frac{2}{3} \times \frac{2}{3} = \frac{4}{9}$. The codes are such that they can be uniquely decoded at the receiving end.

For instance, suppose that the sequence '110100' is received: checking the table, we can see that this can only be parsed as 110, 10, 0, or True, False, False, True, False, False.

To find the mean number of digits required to encode two messages, we multiply the length of the codes for the message-pairs by their respective probabilities:

$$\frac{4}{9} + 2 \times \frac{2}{9} + 3 \times \frac{2}{9} + 3 \times \frac{1}{9} = 1\frac{8}{9} \approx 1.889 \quad (5.24)$$

which is less than two bits.

Shannon came up with a formula that gives the shortest possible encoding for a stream of distinct messages, given the probabilities of their individual occurrences:

$$H = - \sum_{i=1}^n p_i \log_2 p_i \quad (5.25)$$

The mean information content in a collection of messages comes by multiplying the log of the probability of each message by the probability of that message. He showed that no encoding of messages in 1s and 0s could be shorter than this.

The formula gave him an irreducible minimum of the number of bits needed to transmit a message stream: the real information content of the stream.

In his 1948 article, Shannon notes:

Quantities of the form $H = - \sum_{i=1}^n p_i \log p_i$ play a central role in information theory as measures of information, choice and uncertainty. The form of H will be recognized as that of entropy as defined in certain formulations of

statistical mechanics where p_i is the probability of a system being in cell i of its phase space. H is then, for example, the H in Boltzmann's famous H theorem. We shall call $H = -\sum p_i \log p_i$ the entropy of the set of probabilities p_1, \dots, p_n .

So here we get a critical result: information and entropy are the same.

5.7 Landauer's limit

Information is not a disembodied abstract entity; it is always tied to a physical representation. It is represented by engraving on a stone tablet, a spin, a charge, a hole in a punched card, a mark on paper, or some other equivalent. This ties the handling of information to all the possibilities and restrictions of our real physical world, its laws of physics and its storehouse of available parts.

(Landauer, 1996).

We discussed earlier how difficult it was to get rid of the heat generated by current CMOS circuits. That argument was based on currently existing techniques for building logic gates. Human ingenuity being what it is, we should expect that novel designs of gates will arise in the future that allow components to be even smaller, and to use less energy.

The decisive factor in cooling is the number of watts per square centimetre of heat released. Provided that the shrinkage in area proceeds as fast as, or faster than, the shrinkage in power consumption, we will be OK. The most obvious step is to reduce the voltage at which the chip operates—and this has indeed been done over time. But there is an interaction between gate voltage and reliability. As you shrink gate voltages, the reliability of the gate in the face of thermal noise declines.

5.7.1 Thermal noise

The thermal fluctuations of voltage on a capacitor in a CMOS chip give rise to a thermal noise voltage (Kish, 2004), which has the form

$$U_n = \sqrt{\frac{kT}{C}} \quad (5.26)$$

As the capacitance of the gate falls, the thermal noise rises. Note the similarity of this to the distribution discussed on page 85. The discrete nature of charge lies at the base of the thermal noise, as it does with shot noise. If we assume that gate oxide layers have hit an effective lower limit at about 1 nm, then capacitance is going to scale proportionately to λ^2 , where λ is the minimum feature size on a chip. If we reduce λ from 40 nm to 20 nm, the area of a gate capacitor will fall from 160 nm² to 40 nm². Thus, by Eq. 5.26, if we halve the feature size on a chip,

we double the thermal noise voltage. To provide adequate protection against the occurrence of errors, it is necessary for the operating voltage level of the chip to be scaled with the thermal noise such that the voltage difference between a 1 and 0 is about 11 or 12 times U_n . From this, it follows that if we keep shrinking our transistors, we will have to start increasing the operating voltage of the chips to provide a sufficient noise threshold.

This obviously has serious implications for power consumption per unit area, since the power consumption is proportional to the square of the voltage. It means that continual shrinkage of CMOS gates will hit a noise-limited floor to power consumption per square centimetre that will scale inversely with the square of the feature size. A halving of feature size will lead to a quadrupling of power consumption at a given clock speed. In effect, this sets a floor to the amount of energy that a CMOS gate can use and still perform reliably.²

²See also Eq. 6.67.

But this seems tied to the particular properties of CMOS and raises a more general question: Is there an ultimate lower limit to the amount of power that a computer must use?

This was asked by (Landauer, 1961, 1991, 2002) and the answer he gave was 'yes'. His argument was based on an ingenious combination of the theorems of Boltzmann and Shannon.

He first observed that two of the three basic gates from which we build our digital logic (AND, OR, NOT) are information destroying (Fig. 5.13). Whenever data goes into an AND gate, two bits go in but only one bit comes out. Since information and entropy are equivalent, this amounts to an entropy reduction operation. But entropy reduction is prohibited in any closed system by the Second Law of Thermodynamics.

What happens to the entropy of the missing bit?

It can't just vanish: it has to be turned into heat. If the entropy of the logic goes down, the entropy of the environment must rise to compensate. Using Boltzmann's constant again, he derives the minimum energy that is wasted by discarding a bit as $kT \ln(2)$ joules.³

³The $\ln(2)$ term arises because Boltzmann's constant is defined in terms of natural logarithms and information is defined in terms of \log_2 .

We can fit this into our previous calculation of the power used by a K10 processor and find out what would be the minimum power that it could use. We make the same assumption about the number

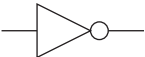
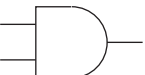

| | | Inputs | Outputs | Bits lost |
|----------|---|--------|---------|-----------|
| NOT gate |  | 1 | 1 | 0 |
| AND gate |  | 2 | 1 | 1 |
| OR gate |  | 2 | 1 | 1 |

Fig. 5.13 The basic gates used in modern computers. All but the NOT gate destroy information.

of active gates as before, and assume that it is working at room temperature:

| | |
|-----------------------------------|-------------------------------|
| k = Boltzmann's constant | 1.38×10^{-23} |
| T = temperature | 300 K |
| $\ln(2)$ | 0.693 |
| $e = kT \ln(2)$ = energy per gate | 2.87×10^{-21} joules |
| n = number of active gates | 50 000 000 |
| f = frequency | 1 GHz |
| $p = fne$ = power | 1.43×10^{-4} W |

This is a very small amount of power. The actual K10 uses about a million times as much power (p. 94). So there is plenty of room for improvement before we hit the ultimate limit of thermal efficiency for our computers!

So there is a limit. Why worry? Be happy! It is far away!

There are several reasons to worry.

We have to consider first that Landauer's limit is very much a lower bound for a gate. A gate dissipating energy at this rate so close to the thermal noise level would be too unreliable for practical use. But if we ignore that for now, and pretend that we really could have a computer using only the Landauer energy, the rate of improvement in computer technology has been such that we would soon reach even that limit. The first computers used triode valves, which used about 5 W for each gate. The Manchester Mk I machine on which Turing worked as a programmer was the first programmable general-purpose electronic computer. It had about 4000 valves and used 25 kW (Lavington, 1975, 1978, 1980). It operated at a clock speed of 85 kHz, so that to perform one logical operation with a triode it was using about 6×10^{-5} joules. The calculations on page 94 indicate that the gates on the K10 use about 2.6×10^{-15} joules per logic operation. So over a 60-year period, gate efficiency has improved by a factor of about 10^{10} .

Efficiency has been improving by tenfold every six years. If this trend continues, then by around 2045, gate efficiency will hit its ultimate thermodynamic limit.

But that only takes the question of efficiency into account. We also have to consider the growth in processor power. This can be bounded by the product of the clock speed and the number of logic operations performed each clock cycle. In practice, due to the increasing complexity of logic to achieve a given instruction mix, this upper bound is not met, but let us use the optimistic measure for now. Since between the 1949 Mk I and the 2007 K10 the gate count of a machine has risen from 4×10^3 to 5×10^8 , a factor of 10^5 , and the clock speed has risen by a factor of about 10^4 , the number of logic operations per second has been growing by a factor of 10 every 6.5 years. If this growth were to continue, how long would it be before even a machine working at

maximal Landauer efficiency would be generating too much heat for us to keep it cool?

The K10 generates about 200 W per square centimetre. It can be cooled, but the limit of what can be done by conductive plus air cooling is probably of the order of 1 kW per square centimetre. Suppose that we envisage a future 1 cm by 1 cm computer chip, operating at maximum Landauer efficiency and putting out 1 kW. Let us call this the Last Chip. How many logic operations per second could it do?

Clearly, the answer is

$$\frac{10^3 \text{joules}}{2.87 \times 10^{-21} \text{joules}} = 3.48 \times 10^{23} \quad (5.27)$$

Since we have estimated that our K10 is doing around 5×10^{16} logic operations per second, this allows our Last Chip to be about 10 million times more powerful than anything we have today. Given the current growth in processing power, this level of performance could be attained in about 40 years—say, in around 2050.

These very crude calculations imply that, if the historical trends were to continue, we would reach 100% Landauer efficiency by about 2045; and that about 5 years later, even chips running at this efficiency level would become thermally limited. So although the Landauer limit is some years away, it can be expected within the working careers of students reading this book.

5.8 Non-entropic computation

The discovery of the Landauer limit prompted research into whether it was possible, in principle, to build computers that would use less energy. One approach was that put forward by (Fredkin and Toffoli, 1982), who invented a new class of logic gates that they called conservative logic. They made the ambitious claim that:

The central result of conservative logic is that it is ideally possible to build sequential circuits with zero internal power dissipation.

(Fredkin and Toffoli, 1982, p. 3).

The key idea was to get round what they saw as the limit in conventional logic technology, that it was not reversible. We have seen how the AND gate takes in two bits and outputs only one. If one could devise a new family of logic gates, which met the conditions that:

- they had the same number of outputs as inputs
- their logic functions were reversible—that is, from the output of the gate you could determine what the inputs must have been

then there would be no loss of information as the gates operated. If there was no loss of internal information or internal entropy, then there would be no need to shunt entropy into the external environment as heat.

Hence, given a suitable reversible gate family, one could in principle build zero-power computers. They set out to answer four questions:

Question 1. Are there reversible systems capable of general-purpose computation?

Question 2. Are there any specific physical effects (rather than mere mathematical constructs) on which reversible computation can in principle be based?

Question 3. In Section 4, we have achieved reversibility of computation at the cost of keeping garbage signals within the system's mechanical modes. In a complex computation, won't garbage become unmanageable if we cannot dissipate it? And won't the need to dissipate garbage write off any energy savings that we may have achieved by organizing the computation in a reversible way?

Question 4. Finally, without damping and signal regeneration, won't the slightest residual noise either in the initial conditions or in the running environment be amplified by an enormous factor during a computation, and render the results meaningless?

(Fredkin and Toffoli, 1982, p. 13).

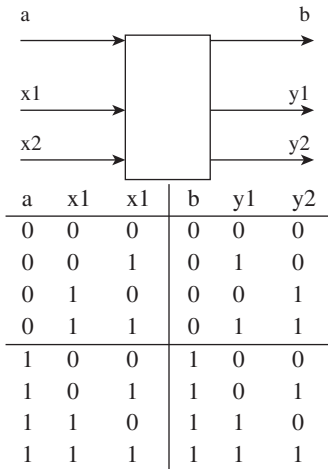


Fig. 5.14 The the Fredkin gate with its input output table.

The came up with adequate responses to the first three questions but, in our opinion, failed on the last one. Whether or not we consider their proposal to be plausible, it certainly remains instructive.

A number of such gates would be possible, but they gave as an example the so-called Fredkin gate, shown in Fig. 5.14. They showed that if one fed 0 or 1 into appropriate inputs, one could emulate the classical AND and NOT gates with Fredkin gates. The fact that no bits were being thrown away did mean that you would have to keep hold of a series of garbage bits generated in the calculation. This might seem to offer no advantage over old-style logic gates, since we have merely postponed discarding entropy by encoding it in the garbage bits. What happens to these bits at the end of the calculation?

If you just put them into a bit sink, you generate heat and have gained nothing by using Fredkin gates. Their ingenious solution was to do the calculation in three stages:

1. Put the input variables V and the required constants C into the gates and compute the boolean result R plus the garbage G .
2. Perform a copy of the result into a final register F . This operation is destructive and liberates $l_F kT$ of energy, where l_F is the length in bits of the result register.
3. Send the information in $\langle R, G \rangle$ back through the logic in the reverse direction, restoring the original V and C .

Since the same C is used every time you do the calculation, the only cost of the whole operation is pre-setting V and setting F , so the bit cost in terms of entropy is just the sum of the input variables plus the output variables. The energy cost no longer depends upon the number of logic steps performed, only on the size of the question and the size of the answer.

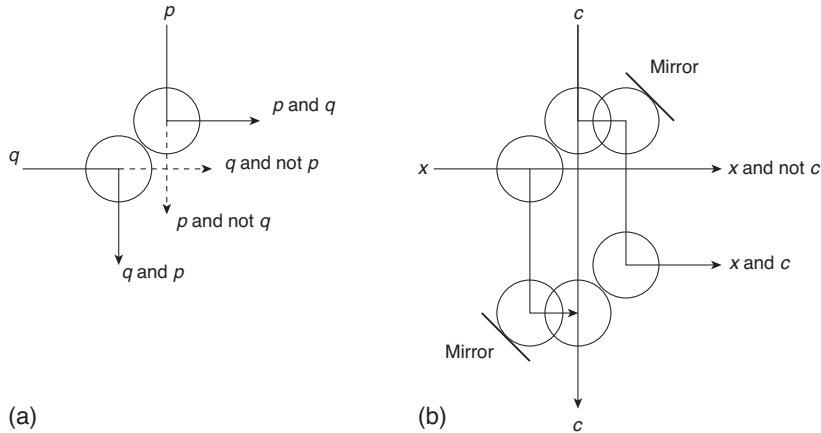


Fig. 5.15 Billiard ball interaction gates: (a) elementary collision; (b) the switch gate. The descending bit C switches the bit X between two possible paths and then continues on its way. Note that the logical output bit C may not be the original ball.

They had now answered their first and third questions, but what about physical realizability?

For this, they proposed billiard ball logic as shown in Fig. 5.15. They argued that spheres of radius $1/\sqrt{2}$ moving with unit velocity along a grid and then undergoing elastic collisions can be thought of as performing logic operations. By inserting reflectors at appropriate positions, it is possible to get the switch gate shown in Fig. 5.15(b). By appropriately routing four of these switch gates in sequence, they showed that one can logically realize the Fredkin gate.

Fredkin and Toffoli argued that they had good precedents for the model that they proposed:

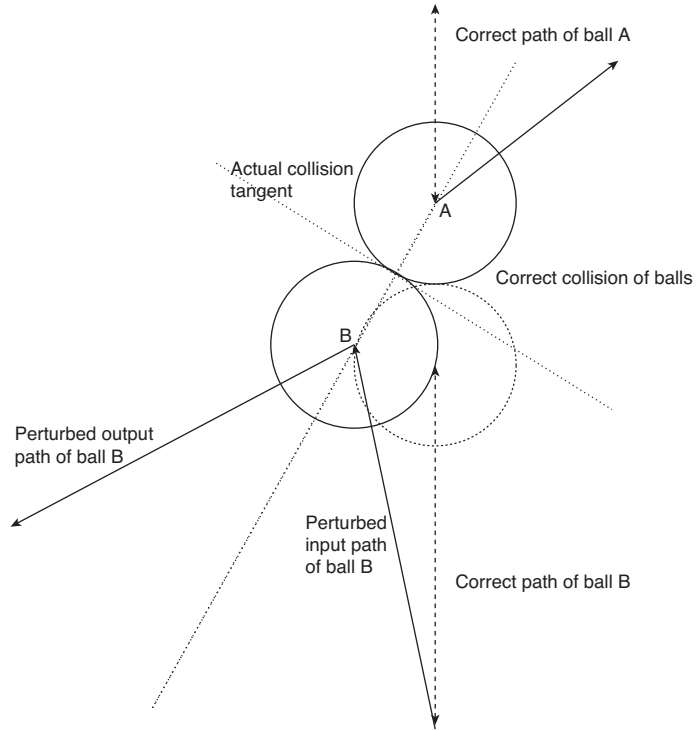
In the past century, a satisfactory explanation for the macroscopic behavior of perfect gases was arrived at by applying statistical mechanical arguments to kinematic models of a gas. The simplest such model represents molecules as spheres of finite diameter. Both molecules and container walls are perfectly hard and elastic, and collisions are instantaneous.

(Fredkin and Toffoli, 1982, p. 18).

This is true historically. In his *Lectures on Gas Theory*, Boltzmann (1995) did indeed assume that molecules were spheres of the sort used by Fredkin and Toffoli, but he also paid considerable attention to the scattering properties of collisions between molecules (Boltzmann, 1995, Chapter 1.4). If we look at Fredkin and Toffoli's proposals from this standpoint, we have to conclude that their gate model is non-viable even within the theoretical framework of rigid perfectly elastic spheres. The problem lies with the exponential instability of collision paths.

Consider Fig. 5.16. In this diagram, the two balls should meet in the middle, on the vertical ingoing and outgoing paths shown as dashed lines. In the Fredkin model, the balls should approach at 90 degrees rather than 180 degrees, but by choosing our frame of reference appropriately we can ignore all but the vertical component of the intended approach. The ingoing path of ball B is shown as being perturbed to the right,

Fig. 5.16 In the event of a perturbed collision between two balls, the deviation of the outgoing path from the correct path is greater than the deviation of the incoming paths.



resulting in an off-centre collision. The consequence of this is that the outgoing path is perturbed by a greater amount than the ingoing one. The perturbation shown is quite large, but even for a small initial perturbation a similar growth in perturbation occurs with each collision. A system of colliding spheres is chaotic; the behaviour of the whole system quickly becomes randomized.

Perturbations are unavoidable. Collisions with the reflectors will give a thermal component to the energies of the balls equivalent to kT , which will become amplified with each collision. After a few layers of logic, the output of the gates will become completely non-deterministic and you will be left with what you should have expected from the kinetic theory of gases, ‘molecules’ bouncing about in a maximally entropic fashion.

For the billiard ball logic to work, we have to postulate the complete isolation of the information encoding degrees of freedom from all thermal noise. For a mechanism operating classically, this is impossible. Fredkin’s model did, however, prepare the way for Feynman’s proposal to use quantum mechanics to implement reversible gates (Feynman 1986, 1999).

5.8.1 Adiabatic CMOS gates

We said when discussing power consumption in CMOS (pp. 93–94) that CMOS circuits used power because they were charging and discharging capacitances every cycle. Let us look at this a little more closely.

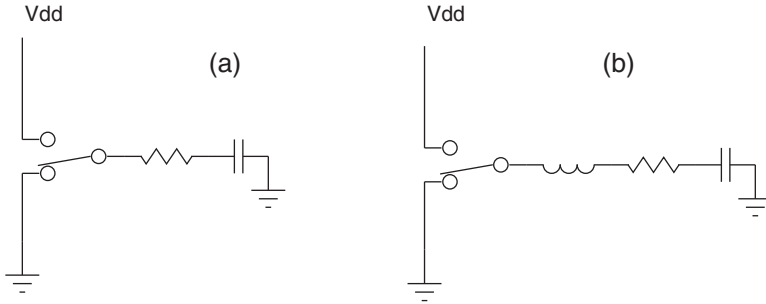


Fig. 5.17 (a) A CMOS gate uses energy charging and discharging its capacitor through the resistance of the circuit that connects it alternately to power and ground. (b) If we place an inductance in the path, we can reduce the potential across the resistance and thus reduce resistive losses at the cost of slower operation.

Figure 5.17 shows a conceptual model of a CMOS gate, which is alternatively charged and discharged through the resistance of the path leading between the gate and either the power supply or ground. In practice, the resistance to ground and to high would not be the same, but that is not important for what follows. Earlier, we estimated the energy loss used by the gate by considering the energy dissipated as the charge on the gate is allowed to fall in potential from V_{dd} (the supply voltage) to ground. What actually happens is that the potential energy on the capacitor is turned into heat overcoming the resistance of the resistance on the path to ground.

The power consumed by a resistor is the product of the voltage across the resistor and the current flowing. Initially, when the switch connecting the capacitor to ground is thrown, the voltage across the resistance is V_{dd} ; but as the capacitor discharges, the voltage falls, so the rate of power dissipation peaks and then falls off. Suppose that we could reduce the voltage across the resistor to less than the full power supply drop from the start. In principle, this can be done by placing an inductance in the path. The initial voltage drop will now be split between the inductor and the resistor, and in consequence the resistor will heat up less, leading to a more efficient overall operation. This mode of operation of CMOS gates is called *adiabatic*.

It is in principle possible to integrate inductors on to chips (Burghartz *et al.*, 1996), but the cost in area of doing this is relatively high. Researchers in the area have instead adopted the approach of globally ramping the power supply up and down using external inductors in the power supply path to the chip. The Charge Recovery Logic family designed by Knight and Younis (1994) works that way, and also requires a considerable growth in the number of transistors used for each logic gate in order to accommodate the fact that the operations are now done using AC rather than DC power. Younis (1994) gave a design for a NAND gate that involved 32 transistors. A normal CMOS NAND requires four transistors. There is thus an eightfold increase in the components required.

The other down side is that the speed at which we can discharge or charge the gates decreases, so we have to operate the circuit at a slower rate. This would seem to defeat the purpose of the exercise, since heat generation is only a problem in high-speed circuits. But it turns out

that if we slow the discharge down significantly, then the power losses through the resistor become proportional to the square of the frequency at which we operate the circuit. Suppose that as a result of using using adiabatic operation, we reduced the clock speed of a 100 W processor chip from 1 GHz to 100 MHz. Using a conventional circuit, the power would drop to 10 W. But if we used an adiabatic circuit, the power loss on each transistor would have fallen to only 1% of what it originally was. Allowing for having eight times as many transistors, we could expect the adiabatic circuit to be using 8 W at 100 MHz. But, of course, if we had simply reduced the clock speed of the original chip to 100 MHz, the power would also have fallen to 10 W without the extra manufacturing cost of using eight times as many gates.

But the number of transistors that you can fit on a chip is a function of your manufacturing process. At any given time, the maximum transistor budget is fixed. An adiabatic computer using eight times as many transistors for each primitive logic gate would have fewer logic gates and thus would need to be of a simpler architectural design. It might have to be a 32-bit processor instead of a 64-bit one; and it might have less internal parallelism, no dual-instruction issue, and so on. This means that our 100 MHz adiabatic chip would have less computing power than the conventional CMOS chip slowed down to 100 MHz.

If we essay more drastic cuts in frequency, the adiabatic technology begins to show some advantages. At 25 MHz the adiabatic chip is using 0.5 W. To cut the conventional chip's power down this low, we would have to run it at 5 MHz. It is plausible that the adiabatic chip would now be doing more computing work for the same energy as the conventional one. So for some applications—for example, where battery usage has to be kept to an absolute minimum, and where the computing requirements are modest—adiabatic circuits have arguable advantages.

It is harder to see them competing with mainstream processors. In principle, by throwing in enough slow adiabatic processors running in parallel we might be able to do the computing work that can be done with a current 3 GHz processor chip and still use less electrical power. But the cost of buying 50 CPUs instead of one would be prohibitive, to say nothing of the impossibility of getting the 50 CPUs to run software that is currently designed to run on one processor. It is true that levels of parallelism are expected to rise anyway, but this is parallelism accompanied by an increase in performance. It is unlikely that people would be willing to spend a lot of money and effort on parallelism at the same level of performance just to save modest amounts of electricity.

5.9 Interconnection

We argued above that there were strong economic and physical reasons why two-dimensional layouts have been favoured for computing hardware: circuit boards, chips, disk surfaces, and so on. We will now look at the implications of this for interconnection. Any electronic computer

is a mass of wires. These are necessary to transmit information between different parts of the machine. Real processors are dauntingly complex, so it is best to start with something simple, the finite state automaton. These were introduced these in Section 2.3.2, in the discussion of counting. They constitute the simplest abstract computing machine and all practical processor chips can be thought of as finite-state automata connected in some way to some sort of external memory.

Suppose that we have a finite state machine capable of entering at 12 distinct states. It is clear that we can't build a binary machine like this unless we have at least 4 bits of binary storage, since 4 bits would allow us to encode 16 states, and 3 bits would only encode eight states. So every finite state machine with n possible states requires a binary state vector of length l , where $2^{l-1} < n \leq 2^l$. We can illustrate this in tabular form:

| States | Bit vector |
|-----------|------------|
| 1 | 0 |
| 10 | 4 |
| 100 | 7 |
| 1 000 | 10 |
| 1 000 000 | 20 |

What does this imply when we come to wire up the machine? How many wires would we need for a ten-state machine, how many for a 100-state machine, and so on?

Since each the next state of any one bit is potentially affected by the current state of all of the other bits, we need, in principle, wires going between all of the bits.

It helps to look at some diagrams. Figure 5.18 shows that as the number of bits goes from one to four, the number of wires grows in the sequence 0, 2, 6, 12, which implies the formula $l(l - 1)$. This is approximately of order l^2 .

As the diagram shows, it becomes more difficult to fit the wiring in as the number of bits increases. Indeed, once we have 5 bits, the wires start to cross one another. Suppose that we were building this out of chips on a circuit board, with each chip storing one bit plus the logic gates to determine what the next state of the bit should be on the basis of inputs from all of the other chips. Up to four chips could have been laid out on a single-layered circuit board. Once you reach five full interconnected chips, you start having to use two layers in the circuit board, so as to allow wires to cross, Fig. 5.19.

The von Neumann computer architecture has been the classic way of getting over the tendency for wiring in a fully connected machine to increase with the square of the number of state bits. In von Neumann's approach, the state vector of the computer is divided into a small number of bits within the CPU that can alter each clock cycle and

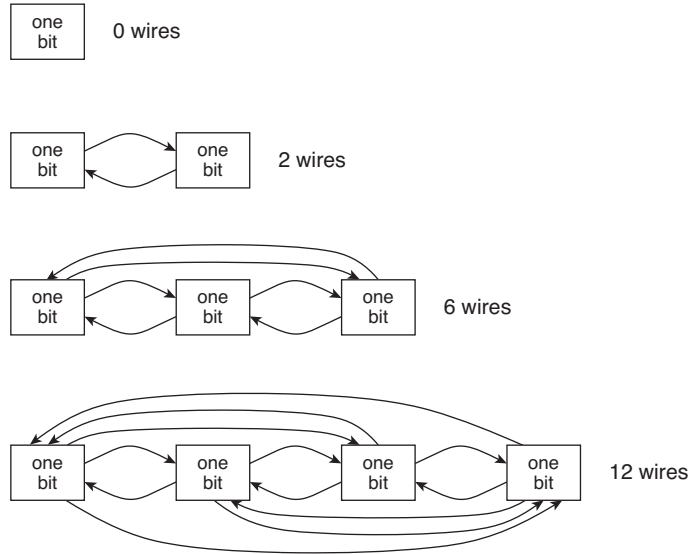


Fig. 5.18 As the number of bits in a finite state machine grows, the amount of wires required grows as $l(l-1)$ in the length of bit vector l .

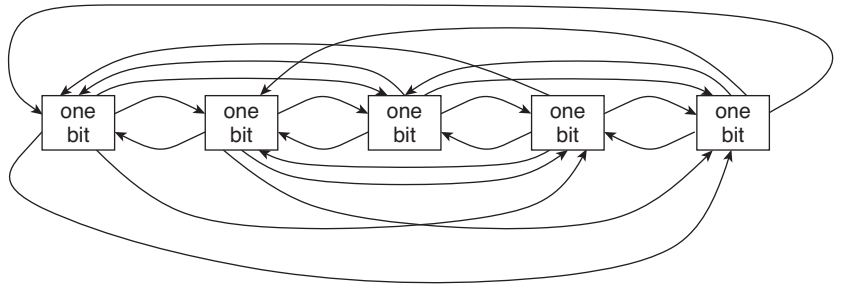


Fig. 5.19 Once we have 5 bits, all of which must be fully interconnected, the wires start to cross.

an external memory, only one word of which can alter each cycle. The number of wires required to connect the processor to an external memory will tend to grow only as the logarithm of the number of bits in that memory, so a 1 kb chip can use 12 wires (one data, one read/write, and 10 address), a 1 Mb chip would use 22 wires (20 address in this case). In practice, for larger memory chips fewer address wires are used, with the most significant address bits being sent first, followed by the least significant bits, to get a further economy. This economy in wiring has been the fundamental reason for the success of the von Neumann architecture. Without it, it would have been physically impossible to construct machines with large numbers of memory bits.

Before whole processors were placed on a chip, the problem of the increasing number of crossovers required to wire up a computer was solved either by increasing the number of layers on the backplane or using individual point-to-point wires, as in the Cray computer shown in Fig. 5.20. On silicon, this option of intruding wires into a third dimension is no longer available to designers. The number of wires used is no longer an important measure of cost; designers are more concerned with the

total silicon area being used. But on silicon the designers also have the ability to alter the shape of the logic blocks that they are connecting up. Instead of just squares or relatively compact rectangles, they can make them into much more elongated shapes.

Figure 5.21 shows a common layout paradigm. The logic circuits driving individual bits are made in the form of elongated columns, with the latched outputs going back up again. Horizontal crossbars, each connected to one of the risers, then route the output signals to all of the logic columns. This layout paradigm is well suited for arithmetic and logic units, and the area tends to grow as the square of the number of bits being operated on. But for fully general finite state machines, the idea that the area will grow as l^2 is an underestimate.

Suppose that we have a finite state machine that cycles between six states. Clearly, we will need at least 3 bits for the state vector. It is necessary for the machine to recognize each of the states that it can enter. For example, state 5 could be encoded in binary as 101. Suppose that we label the bits of the state vector as $B_{1..3}$. Then to recognize that we are in state 5 we have to compute the logical function B_1 AND NOT B_2 AND B_3 . For each state, there will be a corresponding AND term.

Thus the number of AND terms will be proportional to the number of states in the state machine. Figure 5.22 shows a schematic layout for this kind of AND logic arranged in array format. For some state machines, it is possible to optimize the logic and leave out some of the transistors in the matrix, but we can see from the layout that the area of the matrix will tend to grow as $n \log(n)$, where n is the number of states in the state machine. This is much worse than the l^2 suggested by Fig. 5.21, since $l \approx \log(n)$. This implies that for a densely coded finite state machine the silicon area will grow exponentially with the number of state bits. The power used will tend, to first approximation, to be



Fig. 5.20 For larger numbers of connected elements, crossovers can lead to the sort of rat's nest seen in the backwiring of a Cray computer of the 1970s. Photograph due to 'brewbooks' under creative commons license.

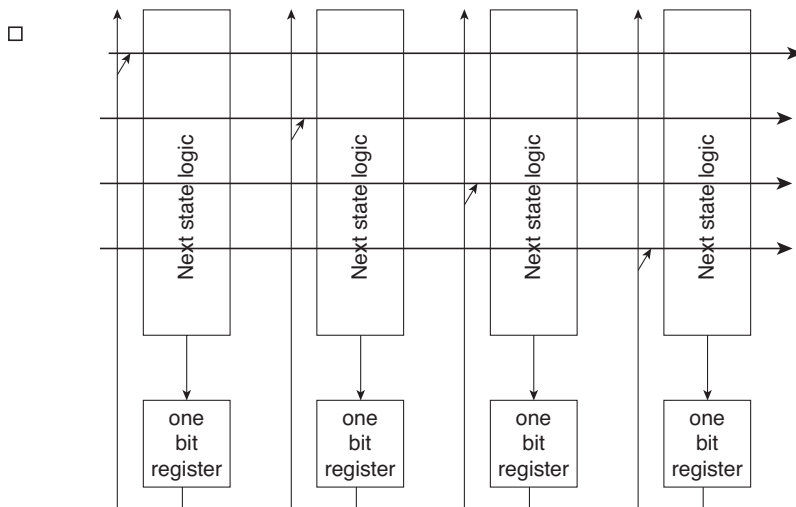
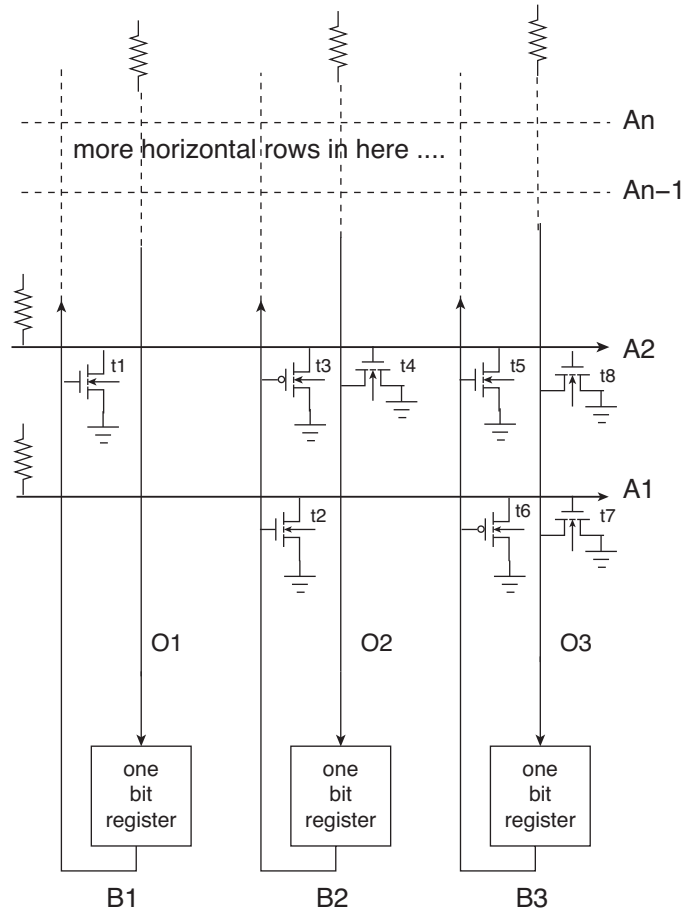


Fig. 5.21 The arrangement of an FSM in array form. This layout paradigm is well suited for arithmetic and logic units, and the area tends to grow as the square of the number of bits being operated on.

Fig. 5.22 The arrangement of a folded array logic. Here, the A lines stand for AND lines. Line A1 will be high if B3 is high and B2 is low, so it implements the equation $A1 = B3 \text{ AND NOT } B2$. The vertical feeds to the registers are termed OR lines. Line O3 for example will be high unless A2 or A3 are high, so it implements $O3 = \text{NOT}(A1 \text{ OR } A2)$.



proportional to the area being charged and discharged each cycle, times the square of the operating voltage, times the clock frequency. Since the area and thus the power are exponential in bit-vector length, it pays to limit finite-state machines to comparatively short bit-vectors. They tend to be used for things like control logic, instruction decoding, and bus protocol handling.

For certain specialized applications, it is possible to construct very large state machines on silicon, ones with hundreds or thousands of state bits, all updated each cycle. Areas in which this can be done include road traffic simulation (Milne *et al.*, 1993; Russell *et al.*, 1994; George, 2005; Tripp *et al.*, 2005) and the simulation of two-dimensional lattice gases (Shaw *et al.*, 1996). In these applications, the physical system that is being simulated is both two-dimensional and operates by means of local interactions. In a road traffic simulation, for example, a car has to respond only to the car ahead of it. One can thus lay out simulated roads on the silicon, with each road segment having a state bit to indicate if there is a car on it. Relatively simple boolean logic connects this state

bit to those immediately ahead and behind—or in the case of a junction, to those to the left and right. The resulting finite state automaton uses local wiring and has a wiring cost that is linear in the size of the physical system being simulated. Because all state bits in a finite state automaton can be simultaneously updated, this sort of architecture can be millions of times faster than a von Neumann computer performing the same task. The acceleration comes both from overcoming the von Neumann bottleneck and because transition rules encoded in logic gates can operate much faster than those implemented in software.

Although these approaches are theoretically promising, they have not been widely used. The rather specialized areas of application have meant that they have remained high cost. Without the mass production that has driven down the cost of standard von Neumann chips, they have not proven competitive.

6

Quantum computers

| | | |
|-----|--|-----|
| 6.1 | Foundations of quantum theory | 128 |
| 6.2 | The quantum rules | 136 |
| 6.3 | Qubits | 142 |
| 6.4 | Entanglement and quantum registers | 146 |
| 6.5 | Quantum computers | 153 |
| 6.6 | Quantum algorithms | 155 |
| 6.7 | Building a quantum computer | 157 |
| 6.8 | Physical limits to real number representations | 167 |
| 6.9 | Error rates in classical and quantum gates | 171 |

6.1 Foundations of quantum theory

There is an often-related story that in an address to the British Association for the Advancement of Science in 1900, Lord Kelvin said:

There is nothing new to be discovered in physics now. All that remains is more and more precise measurement.

In fact, the supposed quote is completely unsubstantiated, but it has gained traction in popular lore because, at the end of the nineteenth century many scientists genuinely believed that physics was almost complete, excepting only a small number of anomalies that would soon be cleared up. It is arguable that, in any era, some scientists have a tendency to overestimate the ratio of the known to the unknown. However, such hubris at the turn of the twentieth century is particularly remembered because it was proved so spectacularly ill-conceived so very soon afterwards, by the advent of two ground-breaking assaults on the classical worldview: relativity and quantum mechanics.

Of the two, quantum mechanics proved the greatest threat to the ‘commonsense’ scientific and metaphysical concepts that reached full fruition in the eighteenth and nineteenth centuries. Relativity does require a reconsideration of the nature of time and space, but the general philosophy of science is perturbed only a little. True, the physical realism of Galileo, Newton, and their successors must be modified so that the fundamental constituents of the external world are seen to be ‘events’ rather than particles and fields, but the basic concept of a mathematically describable mind-independent objective world is never thrown into doubt. In quantum mechanics, however, even this most cherished of ideas is questioned in a way that blurs the very notion of a unique objective reality. Niels Bohr (1885–1962), one of the individuals most deeply involved in the birth and early development of the theory, famously declared:

Anyone who is not shocked by the quantum theory has not understood it.

Like relativity, quantum theory has its origins in the apparently minor anomalies mentioned above which, at the dawn of the new century,

remained stubbornly unresolved, resisting the otherwise apparently inexorable explanatory advance of classical physics. In particular, classical electromagnetic theory was unable to account for the way in which hot bodies emit radiation. Without going into details, the problem was solved by Max Planck (1858–1947) who, in 1901, postulated that electromagnetic waves could only be emitted in quantized form, with allowable energies that are integer multiples of $h\nu$, where ν is the frequency (in hertz) and h is a fundamental constant of nature, now called *Planck's constant*, with a value of about 6.6×10^{-34} joule-seconds. Planck's was a revolutionary proposal, since it suggested that electromagnetic energy is constructed in a discrete, and not continuous, manner.

The idea that radiated energy might be formed from discrete particles was not entirely without historical precedent. Newton famously held that light was composed of a stream of 'corpuscles', but his theory had been unable to account for the phenomena of interference and diffraction and was soon overtaken by the explanatory power of the rival notion, championed by Huygens, that light was a wave disturbance. Although Newton remained convinced that the particle model was correct, arguing that light could travel through a vacuum, where no medium existed to be disturbed, his ideas fell into disfavour. The argument seemed settled for good when James Clerk Maxwell (1831–79) showed mathematically, in 1865, that light was indeed a wave disturbance, obeying the same laws as electric and magnetic fields.

A conceptual problem was perceived, however, which can be viewed in retrospect as a result of the deep-seated desire of the scientists of the time to see the objective physical world in strongly material terms. The very idea of immaterial fields capable of generating forces at a distance, although having great explanatory power, was in something of a conflict with the metaphysics of the age and many physicists, including Maxwell, believed that some invisible material medium must permeate all space in order to carry the fields and their disturbances: this undetected element of objective reality was called the *luminiferous aether*. It was the failure of all attempts to detect this elusive substance that started the sequence of events that led Einstein to propose the Theory of Special Relativity in 1905.

In the same year, Einstein also adapted Planck's quantization concept to explain the otherwise mysterious *photoelectric effect*, whereby light above a certain threshold frequency could liberate electrons from certain materials, *regardless of how low the intensity was*. Einstein proposed that light was indeed quantized into particle-like units that he called *photons*. At a given frequency, ν , a photon would always have energy $h\nu$, and light at that frequency could never manifest energies other than integral multiples of this basic unit. Einstein's solution is well known nowadays, but perhaps because it is so completely accepted, it is often forgotten just how strange it is. Unlike Newton's corpuscles, Einstein's photons do not eschew wave behaviour. The twentieth-century solution to the old Newton–Huygens conundrum is therefore quite peculiar: apparently,

light is somehow both particle *and* wave, but curiously it never exhibits the two character traits at the same time. Thus was born the concept of *wave-particle duality* and with it the first challenge laid down by quantum theory to classical realism.

Comprehending what it means to say that something can be both particle and wave is key to attempts to sustain a mental picture of objective reality. One obvious approach is to suggest that each photon is a small indivisible packet of waves. Unfortunately, this visualization, while perhaps comforting to the classical realist, does not agree with observations. To see this, it suffices to consider a variation of one of the most famous empirical demonstrations of all time, originally performed around 1801 by Thomas Young and known as the *double-slit experiment*. In the words of Richard Feynman *et al.* (1963), this reveals:

... a phenomenon which is impossible, absolutely impossible, to explain in any classical way, and which has in it the heart of quantum mechanics. In reality, it contains the only mystery.

In the double-slit experiment, light from a monochromatic source (one wavelength only) is shone through two parallel narrow slits, A and B and, on emerging, allowed to fall on a screen placed some distance behind (see Fig. 6.1). As is well known, if one of the slits, say A, is covered and the other left open, a simple bar of light will appear on the screen opposite the other, B, the edges of the bar made fuzzy by diffraction (Fig. 6.2(a)). If slit B is covered, with A open, an identical bar appears behind slit A. On the other hand, if both slits are uncovered, the screen displays the familiar *interference pattern* of alternately light and dark stripes, stretching between the positions formerly occupied by the diffraction bars (Fig. 6.2(b)). So far, this is unremarkable: in essence, it is the form of the experiment that Young used to demonstrate that light is a wave.

If we decrease the intensity of the light source, the brightness of the interference bands and the diffraction bars fades, eventually becoming too dim to see with the unaided eye. Now replace the screen with some mechanism for accumulating the sum of all the light hitting each point in its plane: sufficiently sensitive detectors would be able effectively to count the number of photons hitting them while the experiment runs.

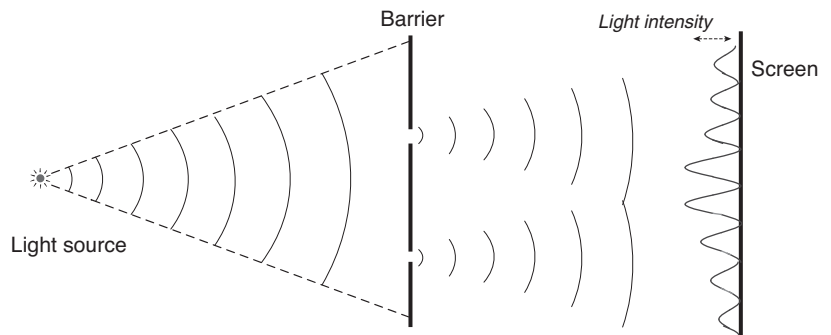


Fig. 6.1 The double-slit experiment.

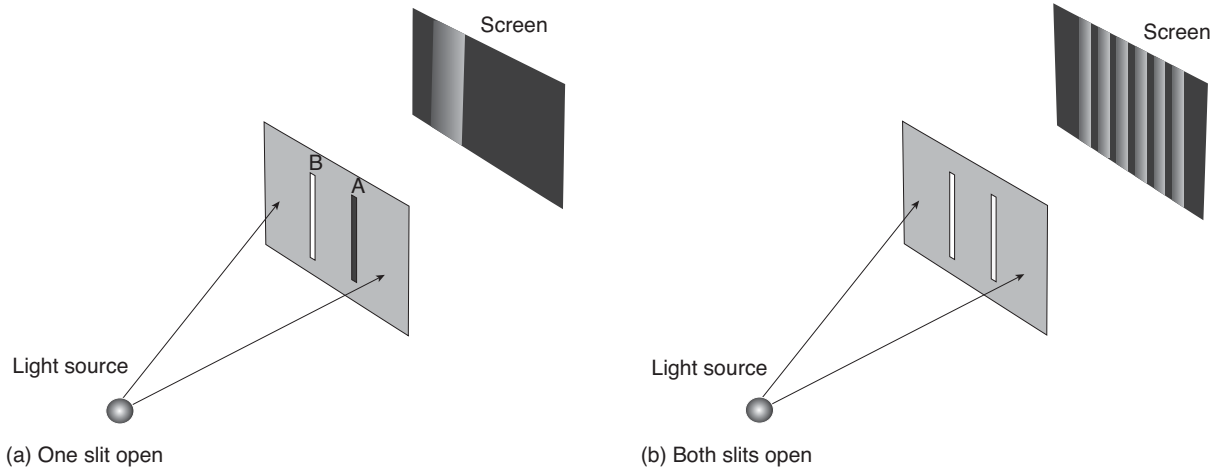


Fig. 6.2 Interference and diffraction.

For convenience, we will continue to call our array of photon detectors the ‘screen’. The final step is to decrease the light intensity to the point at which the source is emitting just one photon at a time. Consider this for a moment. If photons are indeed particles of light then, given what we normally mean by a ‘particle’, we would expect each photon to go through one slit or the other, but not both. Sending photons one at a time, therefore, might be expected to eliminate any possibility of interference. Sure enough, each individual photon does impact at exactly one spot on the screen, but as the cumulative effect of the photons is counted at that plane, surprisingly, the interference bands begin to reappear. If one of the slits, say A, is blocked, the interference patterns again disappear and are replaced with an accumulated diffraction bar opposite B.

In fact, if any measurement is made to determine which slit a photon goes through, the interference pattern disappears. This remains true even if all the photons going through both slits are allowed to proceed to the screen, although there are now two diffraction bars, one behind each slit. It is possible to detect which opening a photon has gone through while still allowing it to proceed to the screen; for example, by placing a beta barium borate (BBO) crystal behind each slit. Such a crystal acts to split a single photon into two identical offspring, each with half the energy of the original and able to be sent in different directions: one, usually called the *signal* photon goes to the screen, while the other, the *idler* goes to a detector that is possibly much further away. The splitting process is called *spontaneous parametric down-conversion* (SPDC) and the photons created are said to be *entangled* (something we will have more to say about later). With clever use of geometry and polarizers, it is possible to perform measurements on the idler photon from a pair

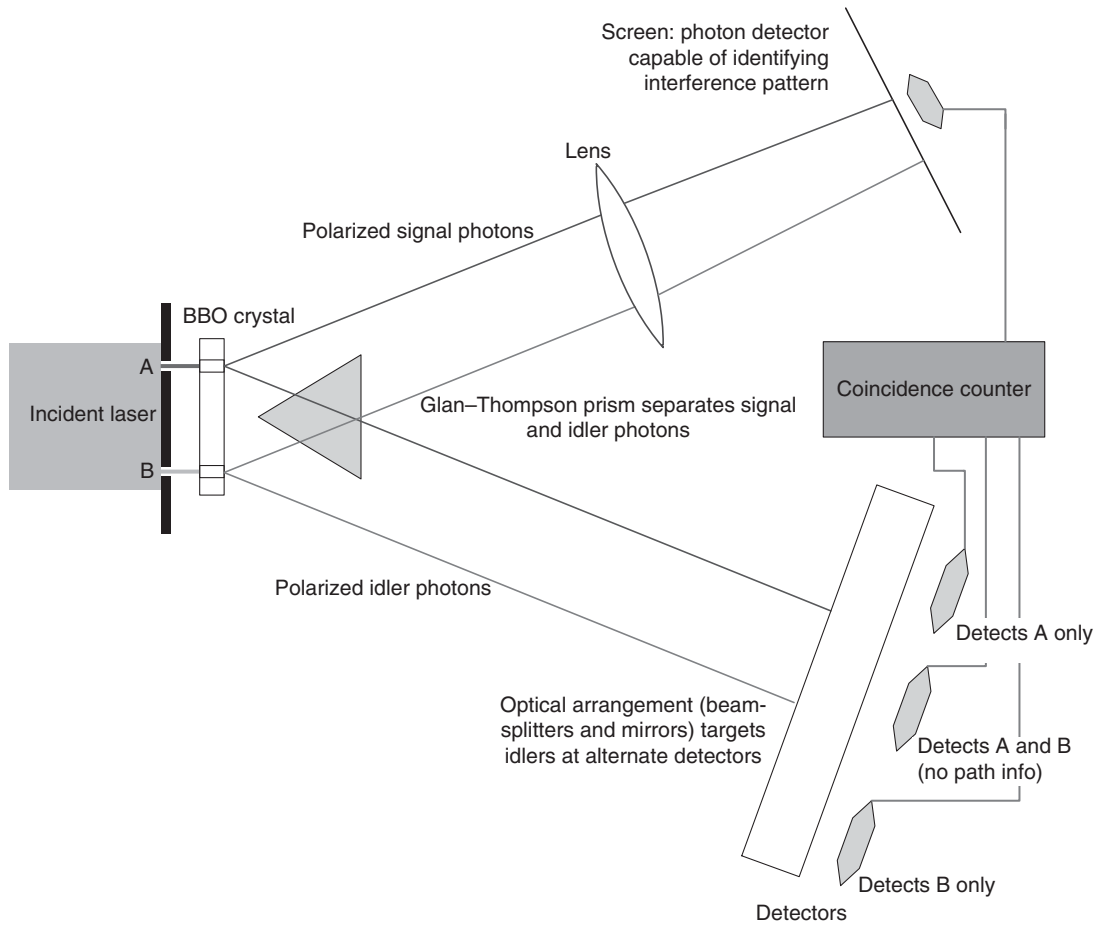


Fig. 6.3 The delayed choice quantum eraser experiment.

that indicate exactly which slit the corresponding signal photon has gone through. In an ingenious experiment called the *delayed choice quantum eraser* (Fig. 6.3), it has been shown (Kim *et al.*, 2000) that if the idler photons are measured to discover the ‘which path’ information, the signal photons will not show an interference pattern; if, however, they are not measured, the pattern reappears. Strangest of all, the choice of whether or not to measure the ‘which path’ information need not be made until after the signal photon has hit the screen!

How can this behaviour be rationalized? Photons behave like particles when they are detected, but like waves when left to their own devices. In an unattended double-slit experiment a photon apparently goes through both slits at once and interferes with itself. It seems to know what is happening to its entangled partner, however far away, and it is even able to anticipate the decision an observer will make about its

partner in the future! What, we feel compelled to ask, is going on behind the scenes, when nobody is looking, when no observation is being made? In short, nobody knows! However, that has not stopped many physicists and philosophers from speculating—and, indeed, there has been a minor industry in the invention of *interpretations*, models of a possible underlying reality that might allow us to visualize what is ‘really’ going on in the objective unmeasured world.

It was soon realized that it is not just light that is affected by the paradoxes of wave–particle duality. The quantum ideas of Planck and Einstein were adapted by Bohr to explain another previously intractable phenomenon, the energies of the light emitted by heated hydrogen gas. Bohr’s model of the structure of the hydrogen atom assumed that ‘orbital’ electron energies are also quantized and was thus able to predict the spectrum of the emitted light correctly. Unfortunately, the model only worked for hydrogen, and even Bohr himself realized that it was *ad hoc* and messy in nature. However, the idea that electrons also obeyed quantization principles was soon generalized: in 1924, Louis de Broglie advanced the idea that matter, just like light, exhibited wave-like properties, but that these were not normally observed because the wavelengths involved were usually so short. The de Broglie hypothesis suggested that any particle of matter of momentum p has a wavelength,

$$\lambda = \frac{h}{p} \quad (6.1)$$

where, as usual, h is Planck’s constant. This equation is exactly the same as that for a photon, remembering that photon energy and momentum are related by $E = pc$ with c the speed of light in a vacuum and, of course, that the wavelength λ and frequency ν obey the familiar relationship $\lambda\nu = c$. Some quick experimentation with Eq. 6.1 will show the reader that the wavelengths of particles of anything other than the tiniest mass will be so small as to be effectively unmeasurable. However, in 1927, Davisson and Germer were able to demonstrate experimentally that electrons do indeed exhibit diffraction and have a wavelength exactly as predicted by de Broglie. Subsequent investigation has confirmed this beyond all doubt: at the microscopic level, all matter exhibits exactly the same wave–particle duality as does light. Electrons too can be subjected to double-slit experiments and they demonstrate the same behaviour in all its peculiarity. At the microscopic level at least, it seems that the whole world is quantum!

What was lacking after all these developments was a coherent theory paralleling the role of Newton’s mechanics or Einstein’s relativity in the classical arena. In a spectacularly successful burst of activity in the late 1920s, the physics community delivered just such a theory via the independent work of two of the most outstanding scientists of the twentieth century, Erwin Schrödinger (1887–1961) and Werner Heisenberg (1901–76). Schrödinger’s theory was formulated around the concept of mathematical waves and is now referred to as *wave mechanics*. Heisenberg’s version, *matrix mechanics*, was more mathematically

abstract and less amenable to helpful pseudo-physical visualizations. However, it was soon demonstrated that the two approaches are mathematically equivalent and following the work of, among others, Max Born (1882–1970), Pascual Jordan (1902–80), Paul Dirac (1902–84) and Jon von Neumann (1903–57), the modern form of quantum mechanics had emerged by the early 1930s from a unification of the two approaches.

In the 80 or so years since its inception, quantum mechanics has proved itself the most successful scientific theory of all time. It has never been shown to be incorrect in its predictions, despite these often being in stark conflict with common sense. It is also, as far as anybody has been able to determine, a complete framework. This claim is succinctly summarized by Stapp (1972) in what is sometimes known as the *Weak Completeness Hypothesis*: ‘no theoretical construction can yield experimentally verifiable predictions about atomic phenomena that cannot be extracted from a quantum theoretical description’. A counter-example to this statement has never been demonstrated. So, although nobody knows how reality conspires to produce the results discussed above, we have a predictive theory that is so staggeringly accurate and successful that the question of what is really going on is, arguably, relegated to the philosophical domain.

To summarize, quantum mechanics provides impeccable tools for predicting the results of observations, but says nothing whatsoever about what really exists independently of the observer. It is disconcerting, to say the least, that the theory that lies at the foundation of the whole of science (after all, atomic theory proposes that all macroscopic entities are composed of particles that obey quantum laws) makes no assumptions whatsoever about ontology, excepting only the observer and the information obtained by measurement. This is a very important point that is often ignored in popular discourse, even by physicists who will talk about an electron, say, as if it was like a small classical object moving through space along some trajectory. But according to quantum theory, a particle is only a particle when it is observed and at any other time its nature is indeterminate. Counter-intuitively, even the linear tracks left in cloud chambers by elementary particles are not evidence of classical behaviour, as convincingly demonstrated by Mott (1929). But neither are such particles waves. The wave associated with an electron or a photon is not a material thing, but a function that yields the probability of the particle *being detected* at a given point in space and time, *if somebody were to look*.

Reactions to this odd state of affairs can be divided loosely into two broad camps. The first, often claimed to be the official view, follows Bohr into a revolutionary but disconcerting philosophy, sometimes said to be inspired by positivism, that the theory should be seen as an *instrumentalist* tool for making predictions about future observations. The instrumentalists generally claim that the nature of objective reality is a metaphysical matter and, if quantum mechanics is indeed complete, it is meaningless to speculate on what can never be decided by experiment.

The second camp, however, is committed to the belief that not only is the concept of objective reality meaningful (this is not *itself* in conflict with an instrumentalist view) but, crucially, that it is possible to visualize that reality in some mathematically accurate way. Of course, if quantum mechanics is complete, its predictions cannot be augmented by any additional theory of reality—and therein lies the problem. For, in fact, numerous interpretations can generate the quantum predictions, but while some are more visualizable than others, it is fair to say that all include concepts that are alien to classical thought. For example, building on an idea proposed by de Broglie to explain wave–particle duality, that particles are in some sense guided by ‘pilot waves’, David Bohm postulated a holistic non-local element of reality that he called the *quantum potential*, which is generated simultaneously by all particles in the universe. Bohm’s model (Bohm, 1952*a,b*) is detailed and technically impressive and reproduces the quantum predictions, but it adds no new ones and it is not unique. Unless completeness can be disproved and an experimental difference demonstrated between interpretations, realist models will always be vulnerable to the accusation from instrumentalists that, in their desire to keep hold of the classical idea that science reveals reality, they are postulating entities that are as imaginary as the luminiferous aether that led the thinkers of the nineteenth century so spectacularly astray.

Quantum mechanics was applied extensively in the decades after its development. In the early 1980s, Richard Feynman examined the problems associated with the simulation of many-body quantum-mechanical systems on conventional computers, as this can become infeasible for more than a very few particles. He suggested the possibility of creating a *quantum computer*, using components directly exhibiting quantum-mechanical behaviour, which he believed might be able more efficiently to simulate quantum systems (Feynman, 1986). The idea was further developed over the following years by several physicists, culminating in a description by David Deutsch (1985) of a *universal quantum computer* that can simulate any Turing Machine, and hence is capable of performing any computing task that is possible on a conventional computer. Although the inverse is also true (a quantum computer is not able to perform computations that a classical machine cannot undertake), it can, in principle at least, exploit what Deutsch called *quantum parallelism*, allowing certain probabilistic tasks to be performed much faster. In 1989, Deutsch made another key contribution to what then became a very active field, when he showed how a quantum computer could be built up using multiple copies of the most elementary quantum systems, called *qubits* (which are such that each measurement has only two possible outcomes), in conjunction with simple quantum operations known as *quantum gates* (Deutsch, 1989).

Deutsch’s proposals were, of course, entirely theoretical at the time, and a great deal of effort has since gone into establishing to what extent it might be practical to implement them. There are two key issues. First, how might a functioning and usable quantum computer be built?

Secondly, how are useful algorithms to be designed offering a worthwhile speed-up over conventional machines? The first problem requires usable qubits and quantum gates to be fabricated in some physical technology and, it has to be admitted, this is a very hard problem that has not yet been workably solved. On the second issue, some progress has been made and a number of interesting quantum algorithms have been discovered, the most famous of which is Shor's (Shor, 1999), that would allow an integer to be factored on a quantum computer in a time that is exponentially faster than any known conventional approach. However, the design of useful quantum algorithms is not easy either, and progress in this area has also been slow.

In order to understand the theory behind quantum computers and algorithms, it is necessary to have a basic knowledge of the details of quantum mechanics, and this is the task to which we now turn.

6.2 The quantum rules

Quantum mechanics is a fundamentally epistemological theory. It is arguable that all of science is of this nature, but classical physics can be stated in *strongly objective* language (d'Espagnat, 2006); that is, its statements describe a world whose attributes are independent of the community of observers. Since all science begins with sentient investigation, some have contended that strong objectivity is an illusion but, whatever the case, in the quantum theory it is not present. All statements of the latter concern not what *is*, but what will be observed when someone looks. On what occurs between observations the theory is silent, providing only a calculus for determining the constantly evolving probabilities of the outcomes of the next observations.

Quantum theory is also inherently non-deterministic, although this claim requires some elaboration. When a measurement is made of a quantum system, we can typically enumerate the set of possible results (which may be infinite), but we cannot in general say in advance which value will be obtained. The theory does, however, allow us to compute probabilities for each such possible result and, while no prediction can be made in an individual case, when the measurement can be repeated on multiple identical systems, the relative frequency of the outcomes is observed to converge to these probabilities. When no observation is being made, however, the measurement probabilities themselves evolve deterministically with time according to the famous *Schrödinger equation*, given knowledge of the nature of the system and its environment, specifically via the system's total energy. The energy of a system is in part due to its motion (kinetic energy) and in part due to the environmental forces acting on it (potential energy), and is usually formally summarized in a function of the system's physical parameters (position, momentum, and so on) called the *Hamiltonian*, after the nineteenth-century Irish mathematician William Hamilton (1805–65). So although the outcomes of quantum measurements are random, the

evolution of the probabilities associated with these measurements is deterministic.

To reiterate, then, quantum mechanics is concerned purely with predicting the outcomes of observations yet to be made. A quantum system is typically of atomic dimensions and assumed to be isolated from interaction with the environment, excepting that there may exist, in its Hamiltonian, a description of any fields in which the system may be located and that may endow it with potential energy. These are idealizations, of course, and they can be relaxed in order to extend the domain of the theory, but for the moment they will be accepted without further discussion. An observation involves making a measurement of one or more physical quantities associated with the system, and ideally we would like to have a quantum-theoretic description that would allow us to predict the outcome of any measurement. However, such a description can usually only be arrived at by knowing something of the history of the system; and since quantum mechanics is non-deterministic, it only allows us to predict probabilities of obtaining possible outcomes, not the actual outcomes that will in fact occur. Further, once a measurement of a quantum system is made, that measurement itself may change the system and our quantum-theoretic description of it must then change too.

In quantum mechanics, the quantities that we measure and that can change with time are called *dynamical variables* or *observables*. In the non-relativistic theory, mass is considered to be conserved and is treated as a constant, but all the components of position, momentum, energy, angular momentum, and so on are dynamical variables. When we conduct a measurement, possible outcomes are always considered to be real numbers, but the set of possible results for a given observable in a given system may be finite, countably infinite, or uncountably infinite. In most of what follows, we shall restrict ourselves to situations where the set of possible outcomes is finite, since this is the scenario applicable in quantum computation. As observed above, for a qubit system, there are no more than two possible outcomes for each observable.

Given that the quantities listed above (position, momentum, etc.) are all normally considered to take any value in some bounded subset of the real numbers, the fact that there are finite scenarios may seem surprising. In reality, of course, no measurement can be made with anything more than limited precision, so the set of outcomes of a measurement is, strictly, always finite; however, typically, when measuring quantities such as position, it is convenient to dismiss this awkward fact and assume, for analytical purposes, that real arithmetic is applicable. Even if we ignore the question of limited precision (as we shall), quantum systems, unlike classical ones, often exhibit inherently discrete behaviour. We have already seen this in the behaviour of electromagnetic radiation, which always manifests as an integral number of photons. As another example, if we consider the energy levels that an electron can take in an atom of, say, hydrogen, it has been clear since before quantum theory

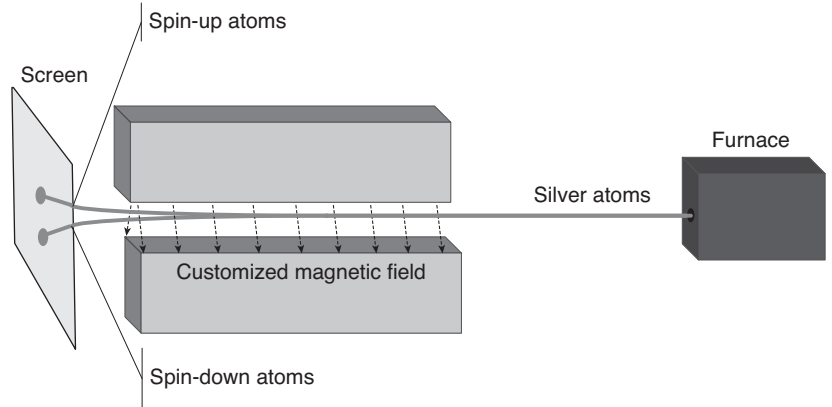


Fig. 6.4 The Stern–Gerlach experiment.

was formulated that only a countable set of characteristic values are possible.

When studying a quantum system, it is also often convenient to restrict the system so that only certain observables are relevant and those have finite sets of possible outcomes. A very well-known example of such a system is electron spin. As early as 1922, the famous Stern–Gerlach experiment (Fig. 6.4) showed that if electrons are passed through an appropriate magnetic field, half of them will be deflected upwards and the other half downwards (in fact, Stern and Gerlach used silver atoms, which each have a single outer electron but are electrically neutral and thus easier to study). This was interpreted as showing that if a measurement is made relative to some chosen direction in space (an axis), electrons will exhibit a component of angular momentum relative to that direction which is either $\pm\frac{1}{2}\hbar$ (joule-seconds) where $\hbar = h/2\pi$, with h being Planck’s constant. (\hbar is a quantity often encountered in quantum theory and is known as the *reduced Planck constant*: its value is very close to 10^{-34} J-s.) This component of angular momentum was later labelled *spin*, and the two allowable values became called *spin-up* and *spin-down*. Such labelling is analogously helpful, but even in a realist picture it must be admitted that it is misleading if taken too literally: an electron has no internal structure nor extension in space (indeed, it only has a particle nature when being measured), and it does not make sense to talk of it spinning in the way a macroscopic sphere might spin on its axis.

Perhaps a word of explanation is in order as to why the spin phenomenon causes electrons (or more accurately, silver atoms) to be affected differently by a magnetic field. Not only does an electron or silver atom have a measurable intrinsic angular momentum (its spin) but, associated with this, it acts as a tiny magnetic dipole, which may be either N–S or S–N depending on whether its spin is up or down. When a beam of such entities is passed through a uniform magnetic field, half of them will be deflected one way and half the opposite way, just as Stern and Gerlach observed.

In general, what we call a particle is a quantum system with many potential dynamical variables. For example, it can have position, momentum, and energy as well as spin. When modelling such a system quantum mechanically, it may be, however, that some of these variables are of no interest in a given situation, and it is then possible to restrict the quantum descriptions to those aspects that are contextually relevant. Thus, for example, in quantum computing, an observable such as electron spin, which has only two allowable measurement results, is extremely useful and we can try to set the system up so that observables such as position and linear momentum can be ignored. Indeed, a confined particle with just two spin states is, at least in principle, an ideal way of implementing a *qubit*.

Suppose that a quantum system is subjected to a measurement of some observable, A , and a value, a , is obtained. If the same observable is immediately measured again, the same result, a , should be obtained with probability 1. If another observable, B , is now measured and A is then measured yet again, even if this is done immediately, in general the outcome of the A measurement is no longer certain. Sometimes, however, B is such that the perfect predictability of A is not lost: we then say that the observable B is *compatible* with A .

In certain circumstances (e.g. by measuring compatible observables), a maximal description of the aspects of the system in which we are interested can be obtained and this description is then called a (pure) *quantum state*. In quantum theory, each possible state of a given system is associated with a vector in a Hilbert space over the field of complex numbers. A Hilbert space is a vector space with an inner (scalar) product defined and which is also *topologically complete* (this means that all sequences of elements of the space that converge do so to a limit that is also in the space). The dimension of the Hilbert space in question may be finite or infinite, depending on the aspects of interest in the quantum description.

In the case of electron spin, a Hilbert space of dimension 2 is generated and this is the basic space used in quantum computing. It is perhaps worth mentioning that all real or complex finite dimensional inner-product vector spaces are topologically complete and so are automatically Hilbert spaces. In Schrödinger's wave mechanics, on the other hand, a special case of the general theory, which was devised to handle position, momentum, and energy, the Hilbert space involved is the set of all complex-valued square-integrable functions and is infinite-dimensional; the state vector associated with a system is then known as a *wave function*. In what follows, we will only require finite-dimensional spaces.

Recall that a vector space is a set, V , of entities called *vectors*, which are related to another set, F , of distinct entities called *scalars* that obey the mathematical properties of a *field*. In what follows, the set of scalars will always be the complex numbers, \mathbb{C} . In a vector space of dimension n over \mathbb{C} , each vector can be represented as an ordered array of n complex numbers called 'components' of the vector. Such an array may be written down as an $1 \times n$ row matrix (components are indexed starting from

the left) or as a $n \times 1$ column matrix (components are indexed starting from the top). For a three-dimensional vector, these alternatives would look like

$$[a_1 \ a_2 \ a_3]$$

and

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

respectively. It can be shown that all vector spaces of dimension n over the same field have the same structure (they are said to be *isomorphic*); so, for example, there is really only one complex vector space of dimension 3, which is usually denoted by the symbol \mathbb{C}^3 . Vectors may be summed (by adding corresponding components) or multiplied by a scalar (multiplying all components by the same number). In an inner product space, they may also be combined via scalar multiplication (the inner product), which is a mapping, $V \times V \rightarrow F$, that obeys certain rules. The standard scalar product on \mathbb{C}^n of two vectors \mathbf{a} and \mathbf{b} is often denoted as $a \cdot b$ or (\mathbf{a}, \mathbf{b}) and is defined as follows:

$$a \cdot b = \sum_{i=1}^n \bar{a}_i b_i \quad (6.2)$$

where the overbar denotes the complex conjugate operation.

The essence of the above ideas can be encapsulated in a small number of quasi-axiomatic rules, which we will now review briefly. These rules are most commonly stated using the mathematical formalism of vector spaces and we will adhere to this approach here.

Rule 1. The first quantum rule is just that we can associate, with any quantum system of which we have a complete description, a *state* that is an element ψ of some Hilbert space, \mathcal{H} , defined by the scope of our model. Dirac called such an element a *ket* vector and used the notation $|\psi\rangle$. The way in which kets are used is outlined by the other rules, but it is worth noting a couple of points at this stage. First, multiplying a ket by any scalar (complex number) does not change the information it contains and so, in fact, each description is associated not with one vector, but with an entire one-dimensional subspace or *ray* in \mathcal{H} . Because of this, we often choose as a representative of the ray a *normalized ket* of length 1, derived from $|\psi\rangle$ by dividing it by its length or *norm* (conventionally denoted by $\| |\psi\rangle \|$).

As an aside, in a Hilbert space, the length of a vector is the square root of the scalar product of the ket with itself. In Dirac notation, the scalar product of two vectors $|\psi\rangle$ and $|\varphi\rangle$ is denoted by $\langle \varphi | \psi \rangle$. For any vector, $|\varphi\rangle$, we can define a linear mapping, f called the *dual* of $|\varphi\rangle$, from the Hilbert space \mathcal{H} into the complex numbers, \mathbb{C} by

$$f_\varphi(|\psi\rangle) = \langle \varphi | \psi \rangle \quad (6.3)$$

In Dirac notation, the dual is consistently and elegantly denoted as $\langle\varphi|$ and is called a *bra* (operator). We can thus write

$$\|\psi\|^2 = \langle\psi|\psi\rangle \quad (6.4)$$

Rule 2. The second quantum rule states that each observable is associated with a *hermitian* linear operator that maps the Hilbert space into itself. The outcome of a measurement of the observable associated with the operator A , is always one of its eigenvalues. After a measurement has occurred, if a value, a , has been obtained, the system will be forced into a state that is an eigenvector associated with eigenvalue a . If the system is initially in a state that is an eigenvector of eigenvalue a when the measurement is carried out, then the value a will be obtained with probability 1. This is the only case in which we can be certain in advance what the outcome of a measurement will be.

The *adjoint* of a linear operator, A , is another operator, usually denoted A^\dagger , which is such that, for all kets $|\psi\rangle$ and $|\varphi\rangle$,

$$\langle\varphi|(A^\dagger|\psi\rangle) = (A|\varphi\rangle|\psi\rangle) \quad (6.5)$$

Note for future reference that in Dirac notation it is customary to write $\langle\varphi|(A^\dagger|\psi\rangle)$ as $\langle\varphi|A^\dagger|\psi\rangle$. The existence of an adjoint is assumed without proof. A hermitian or *self-adjoint operator* is simply one where $A = A^\dagger$. It is straightforward to show that the eigenvalues of a hermitian operator are always real, which is just as required for the outcome of a physical measurement. Also, it can be shown (the so-called *spectral theorem*) that the eigenvectors of the different eigenvalues of a hermitian operator form an orthonormal basis of the space. This means that each ket in \mathcal{H} can be expanded in the basis defined by A , thus:

$$|\psi\rangle = \sum_k \langle a_k|\psi\rangle |a_k\rangle \quad (6.6)$$

where the a_k are a complete set of eigenvalues of A and $|a_k\rangle$ are corresponding normalised eigenvectors. Strictly, this assumes that none of the eigenvalues are *degenerate*, where they have two or more linearly independent eigenvectors (these would therefore generate an eigenspace of two or more dimensions). The theory can easily cope with situations in which an operator has degenerate eigenvalues, but it complicates the notation and we will avoid further discussion here.

Rule 3. The third rule explains what happens in the case in which a system in a state $|\psi\rangle$ (which we will assume to be normalized) is subjected to a measurement associated with the observable operator A . We assume here that A has a discrete set of eigenvalues (the following can be generalized to the continuous case, but this is not discussed further here). Let the eigenvalues of A be a_1, a_2, \dots, a_n , so that these are (as stated in Rule 2) the only possible results of a measurement of A . Further, let the normalized eigenvector associated with a_k be $|a_k\rangle$. Rule 3, which is commonly known as the *Born Rule*, says that the probability, w_k , that the measure outcome will be a_k is given by

$$w_k = |\langle a_k|\psi\rangle|^2 \quad (6.7)$$

Note that the probabilities are the squares of the coefficients of the spectral expansion in terms of A 's eigenvectors. It can easily be established from this that if A is measured many times on identical systems in the same state $|\psi\rangle$, the mean value of the results that will be obtained, denoted $\langle A \rangle$, is given by

$$\langle A \rangle = \langle \psi | A | \psi \rangle \quad (6.8)$$

Rule 4. The fourth rule is that the state vector of a system subject only to external forces will change deterministically in time according to the time-dependent Schrödinger equation:

$$H |\psi(t)\rangle = i\hbar \frac{\partial}{\partial t} |\psi(t)\rangle \quad (6.9)$$

where H is the Hamiltonian, the energy observable, of the system. This rule obviously defines explicitly how an undisturbed quantum system will evolve in time and states clearly that the evolution is dependent only on the energy operator.

Rule 5. The fifth rule, to which we will return later, says that if two systems, S_1 and S_2 , can be described by kets from the Hilbert spaces \mathcal{H}_1 and \mathcal{H}_2 , respectively, then the composite system composed of S_1 and S_2 together is described by kets from the tensor product space $\mathcal{H}_1 \otimes \mathcal{H}_2$. The tensor product is a space constructed out of its constituents and whose dimension is the product of those of the two spaces. An example will be given in Section 6.4.

Although the above can be cast somewhat more generally, the five rules stated here encapsulate the basic operation of quantum theory and will suffice for our purposes in the remainder of this chapter. We now proceed by restricting attention to the simplest quantum systems; namely, those that can be described by the two-dimensional Hilbert space over the complex field. All two-dimensional Hilbert spaces over \mathbb{C} are isomorphic to \mathbb{C}^2 , which means that the vectors can always be represented as a pair of complex values once an orthonormal basis has been chosen.

6.3 Qubits

Consider a quantum system that can be described by the space \mathbb{C}^2 . All observables on such a system can have only two possible measurement outcomes (this is obviously assumed acceptable to the observer's requirements for the given system, since it is implicit in the choices made to restrict the model to two dimensions).

If we now select one observable, the two normalized eigenstates, conventionally labelled $|0\rangle$ and $|1\rangle$, or sometimes $|\uparrow\rangle$ and $|\downarrow\rangle$, form an orthonormal basis of the space. As a result, any other state of the system, $|\psi\rangle$, can be represented in the form

$$|\psi\rangle = a|0\rangle + b|1\rangle \quad (6.10)$$

where a and b are complex numbers with $a = \langle 0|\psi\rangle$ and $b = \langle 1|\psi\rangle$. If $|\psi\rangle$ is normalized, then we must have

$$|a|^2 + |b|^2 = 1 \quad (6.11)$$

$|\psi\rangle$ is often represented as a 2×1 column vector, thus:

$$\begin{pmatrix} a \\ b \end{pmatrix} \quad (6.12)$$

As is well known from elementary linear algebra, once a basis of an n -dimensional vector space over a *field* F has been established, all linear operators on the space can be represented as $n \times n$ matrices. If the basis is the set $|e_i\rangle$, then the elements of the matrix, A , representing the operator a in that basis are given by

$$a_{ij} = \langle e_i|a|e_j\rangle \quad (6.13)$$

Further, the set of all such operators is itself a vector space of dimension n^2 isomorphic to the space F^n . Thus the set of all linear operators on \mathbb{C}^2 is a four-dimensional vector space isomorphic to the set of all 2×2 matrices over \mathbb{C} , which we will denote $\mathbb{C}_{2 \times 2}$.

The elements of $\mathbb{C}_{2 \times 2}$ can, of course, be added or multiplied by a scalar, as in any vector space. However, matrices can also be multiplied by each other, an operation that is associative and has an identity, but is not in general commutative.¹

It is trivial to show that the matrices

$$P_\uparrow = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, P_\downarrow = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, S^+ = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, S^- = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad (6.14)$$

form a basis for this vector space. Less obviously, but still easily verified, so do the matrices

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (6.15)$$

X , Y , and Z are called the *Pauli matrices* and all have eigenvalues $+1$ and -1 . Z in particular has as its eigenvectors the basis states $|0\rangle$ and $|1\rangle$. This again is easy to verify (using matrix multiplication). The eigenvectors of Z are often referred to as the standard or *computational basis* of \mathbb{C}^2 .

The original physical motivation for the study of quantum mechanics in \mathbb{C}^2 was just the investigation of the phenomenon of electron spin mentioned earlier. Spin behaves as an angular momentum, for which a general quantum-theoretic treatment was worked out early in the history of the subject. Any angular momentum can be thought of as a vector in three-dimensional space with, therefore, three components. However, interestingly, the observables for these components in quantum mechanics are not compatible, so it is not possible to know values for more than one at a time; it is, however, possible to know the

¹As an aside, bringing this multiplication operation into play endows the vector space with the additional structure of what mathematicians call a *non-commutative ring*, forming a mathematical entity called an *associative algebra*.

total magnitude of the vector at the same time as exactly one of the components. It is conventional to label the default direction along which the chosen component will be measured as z . Thus it is possible to measure simultaneously the total angular momentum, L , and its z -component L_z , but if an attempt is now made to measure, say, L_x , the z -component information is no longer valid. In the context of electron spin, the total spin, S (a specific manifestation of L), is always $\frac{\sqrt{3}}{2}\hbar$ for every electron (we will omit the units, joule-seconds, from now on) while, as discussed above, the value of the z -component, S_z , is either $\pm\frac{1}{2}\hbar$. In the basis defined by the S_z operator, the eigenstates $|0\rangle$ and $|1\rangle$ are, respectively, called *spin-up* and *spin-down* and, in that basis, the operator itself is represented by the matrix $\frac{1}{2}\hbar Z$. Without further proof, we simply note that the matrices for the observables representing the x and y components of spin are, respectively, $\frac{1}{2}\hbar X$ and $\frac{1}{2}\hbar Y$.

If an operator is hermitian, then the matrix representing it in any basis is self-adjoint (also called hermitian). A matrix is said to be self-adjoint if it is equal to its adjoint (the complex conjugate of all the elements in its transpose). It follows that X , Y , and Z , as the matrices of observables, must be self-adjoint (as can be easily verified). It is also the case that

$$X^2 = Y^2 = Z^2 = 1 \quad (6.16)$$

so that each of the Pauli matrices is its own inverse as well as its own adjoint. Any operator or matrix whose inverse is its adjoint is said to be *unitary*.

Most unitary operators are not, however, hermitian and so do not represent observables, although the spin operators are clearly exceptions. Nonetheless, unitary operators are extremely important because they represent reversible *transformations* of a Hilbert space. It is easy to see that unitary operators preserve scalar products and thus vector lengths (norms). This also means that a unitary transformation preserves orthogonal relationships between vectors. Unitary operator eigenvalues always have modulus 1, although these eigenvalues need not be real (the general form is $e^{i\theta}$). However, as with hermitian operators, eigenvectors corresponding to different eigenvalues are orthogonal and, furthermore, form a basis of the space.

If U is a unitary transformation of \mathbb{C}^2 , then it is easy to prove that $U|0\rangle$ and $U|1\rangle$ are also an orthonormal basis. Conversely, if the set $\{|f_0\rangle, |f_1\rangle\}$ is any other orthonormal basis, then the operator mapping $\{|0\rangle, |1\rangle\}$ on to $\{|f_0\rangle, |f_1\rangle\}$ is a unitary transformation, \mathbf{u} , with a matrix in the $\{|0\rangle, |1\rangle\}$ basis given by:

$$U = \begin{pmatrix} \langle 0|f_0\rangle & \langle 0|f_1\rangle \\ \langle 1|f_0\rangle & \langle 1|f_1\rangle \end{pmatrix} \quad (6.17)$$

This is a linear mapping of the space on to itself, which maps the old basis elements on to the new ones and so carries every vector x on to $\mathbf{u}x$.

All this theory can be generalized in a fairly obvious way to n -dimensional spaces, but is most easily illustrated in the two-dimensional

case. For example, another basis of \mathbb{C}^2 is given by the pair of vectors

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (6.18)$$

which are eigenvectors of X . The unitary transformation that carries the standard basis to this one is (expressed as a matrix in standard basis components):

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (6.19)$$

This operator will play an important role in some of what follows and is called the *Hadamard* transformation. The Hadamard operator is also hermitian and unitary, being its own inverse.

An interesting question arises as to whether the transformations described by unitary operators could be implemented in practice on a physical qubit system. In the real world, as discussed in Rule 4, any change in a quantum system is driven by its Hamiltonian according to the Schrödinger equation. Now if the Hamiltonian is time-independent, it can be shown that the evolution in time of a state known at $t = 0$ to be, say, $|\psi(0)\rangle$, will be given at time t by

$$|\psi(t)\rangle = U(H, t) |\psi(0)\rangle \quad (6.20)$$

where $U(H, t)$ is a unitary operator called the *propagator*, which is dependent only on the Hamiltonian and time. Thus, in such a system, evolution in time is not only deterministic, but can be described entirely by a unitary operator acting on the time-independent initial state. If a unitary transformation is to be implemented in the real world, this is how it must be done: a suitable Hamiltonian must be found and applied for a specified time. In a single qubit system this might be achieved by, for example, applying an external magnetic field to a system composed of an otherwise isolated spin $\pm\frac{1}{2}$ particle such as an electron (as we have seen, an electron has magnetic properties associated with its spin and will respond to such a field in a predictable way).

Real-world implementations of unitary transformations on a quantum system are, however, very difficult to implement in practice. Nonetheless, the success of such an enterprise is central to any attempts to build a quantum computer and, in this context, unitary transformations of \mathbb{C}^2 , particularly any physical implementations thereof, are usually referred to as one-qubit *quantum gates*. Thinking in these terms, we can easily see, for example, that the X operator acts on the standard basis as

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle \quad (6.21)$$

and for this reason is often referred to as the quantum NOT gate. Quantum gates can be represented on a quantum circuit diagram (Fig. 6.5), where each gate is drawn as a box with one input and one output. Quantum ‘wires’ connect the boxes indicating a sequence

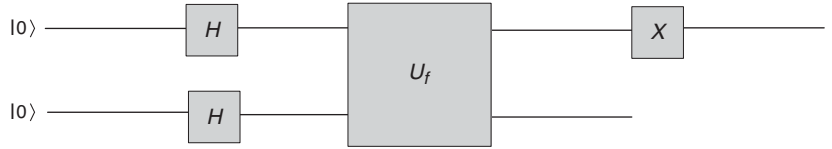


Fig. 6.5 A quantum circuit.

of unitary transformations applied in succession, denoting function composition. The wires should not normally be imagined as real material entities in any sense, but rather as a sequence of operations to be applied to a single qubit system *in situ* (although this is not always the case). Crucially, because any composition of unitary operators is itself unitary, quantum circuits based on single qubit gates are always unitary in their overall operation.

It has to be admitted that one-qubit transformations are not a great deal of practical use on their own or to do any significant quantum computation; multiple qubit systems and their unitary transformations must be considered. This is the subject of the next section.

In closing, observe that unitary transformations have another conceptually distinct but related use, in that they can recompute the components of vectors and matrices relative to a new basis, thus engineering a change of basis rather than a transformation of the space. In this case, the task is as follows: given a column vector or matrix expressed in components relative to the first basis, find the components of the same vector or matrix relative to a new basis. A little algebraic manipulation shows that components of a column vector, W , or an operator matrix A expressed in the old basis, can be expressed in the new one if transformed thus:

$$W' = U^\dagger W, \quad A' = U^\dagger A U \quad (6.22)$$

Note that in this picture it is the basis that has changed. The vectors and operators have not changed, but their matrix representations now record their components relative to the new basis.

6.4 Entanglement and quantum registers

When a system is composed of two component subsystems, Rule 5 says that the compound system is described by state vectors drawn from the tensor product of the state spaces \mathcal{H}_1 and \mathcal{H}_2 of the components. But what is a tensor product of two spaces?

The motivation is to create product vectors composed of any state, $|\psi\rangle$, from \mathcal{H}_1 , simply placed alongside any state, $|\varphi\rangle$, from \mathcal{H}_2 . Unlike scalar products or the vector products from elementary vector geometry, these *tensor products of vectors* or *product states*, as they are called in quantum theory, denoted $|\psi\rangle \otimes |\varphi\rangle$ or often just $|\psi\rangle |\varphi\rangle$, are not elements of either \mathcal{H}_1 or \mathcal{H}_2 , nor of the field \mathbb{C} over which the spaces are defined.

Instead, they belong in a new, bigger space, which we call the *tensor product* of \mathcal{H}_1 and \mathcal{H}_2 , denoted $\mathcal{H}_1 \otimes \mathcal{H}_2$ and also defined over \mathbb{C} .

However, there is a problem. A vector space has to be closed under addition and multiplication by scalars and if $\mathcal{H}_1 \otimes \mathcal{H}_2$ were only the set of all product states, it would not satisfy this requirement and so would not qualify as a vector space at all. To resolve this issue, a tensor product space must include, in addition to all possible product states, all possible linear combinations of these. So elements such as

$$|\Psi\rangle = \lambda |\psi_1\rangle |\varphi_1\rangle + \mu |\psi_2\rangle |\varphi_2\rangle \quad (6.23)$$

where $\psi_1, \psi_2 \in \mathcal{H}_1$, $\varphi_1, \varphi_2 \in \mathcal{H}_2$, and $\lambda, \mu \in \mathbb{C}$, must also lie in the space and can also describe states of the compound system. For consistency, incidentally, we must have that tensor products of vectors are linear in both elements; that is,

$$(\lambda |\psi_1\rangle + \mu |\psi_2\rangle) \otimes |\varphi_1\rangle = \lambda |\psi_1\rangle |\varphi_1\rangle + \mu |\psi_2\rangle |\varphi_1\rangle \quad (6.24)$$

and

$$|\psi_1\rangle \otimes (\lambda |\varphi_1\rangle + \mu |\varphi_2\rangle) = \lambda |\psi_1\rangle |\varphi_1\rangle + \mu |\psi_1\rangle |\varphi_2\rangle \quad (6.25)$$

It is now easy to show that although some such linear combinations can be expressed as product states, many cannot be. An element of $\mathcal{H}_1 \otimes \mathcal{H}_2$ that cannot be expressed as a tensor product of two vectors is called an *entanglement state*.

It can also be shown relatively easily that if we choose a basis of \mathcal{H}_1 and a basis of \mathcal{H}_2 , then the product states of the two bases form a basis of $\mathcal{H}_1 \otimes \mathcal{H}_2$. It follows, as mentioned above, that the dimension of $\mathcal{H}_1 \otimes \mathcal{H}_2$ is the product of the dimensions of \mathcal{H}_1 and \mathcal{H}_2 . Further, to be a Hilbert space, $\mathcal{H}_1 \otimes \mathcal{H}_2$ needs a scalar product and, fortunately, one can be easily defined using the existing scalar products on \mathcal{H}_1 and \mathcal{H}_2 . Thus:

$$\langle \psi_1 \otimes \psi_2 | \varphi_1 \otimes \varphi_2 \rangle = \langle \psi_1 | \varphi_1 \rangle_{\mathcal{H}_1} \langle \psi_2 | \varphi_2 \rangle_{\mathcal{H}_2} \quad (6.26)$$

Physically, quantum mechanics predicts that a compound system composed of two components can be described by a product state or by an entangled state. In a product state, it is clear that each component system can be considered independently as if it can be described by a state of its own. In the state $|\psi\rangle |\varphi\rangle$, for example, system one is described by state $|\psi\rangle$ and system two by $|\varphi\rangle$. In this situation, the two component systems are said to be *separable*. However, in an entangled state no individual state assignment can be made to the component systems at all, and they are said to be *entangled*. Further, it follows fairly simply from the nature of the Schrödinger equation that once an entangled system is created, it will continue to be entangled until it is subjected to a suitable measurement. In general terms, this means that, in a sense, the two components can no longer be considered to have a physically meaningful independent identity unless a measurement can be made that separates them. Curiously, the quantum theory predicts that

this will be the case even if the two components are permitted to travel light years apart. If a measurement is then made on one component that forces it into a state of its own, the other component is also forced into a corresponding state, regardless of how far away it is. Now if states are only descriptions, this sudden change, or *reduction* as it is called, may be considered of merely epistemological significance, with no physical implications. However, states do contain information about probabilities of measurement outcomes and the implication is that such probabilities may be affected for a very distant observer by the actions of a local one. Despite this rather disturbing conclusion, now experimentally confirmed, it was quickly realized that such influences would not permit any form of superluminal signalling and so would not cause any conflict with relativity.

As we have observed, those with a metaphysically realist mindset usually like to identify the elements of a scientific theory with elements of some ontological model of physical reality, and one of the most sophisticated champions of this outlook was Einstein. Despite the ruling out of superluminal signalling, he objected to the fact that, due to its treatment of entanglement, quantum mechanics seems to suggest a form of instantaneous influence across arbitrary distances, even if the nature of that influence extends no further than a change in probabilities of measurement outcomes. Along with Boris Podolsky and Nathan Rosen, he devised a thought experiment which has become known as the *EPR paradox*, wherein the authors tried to demonstrate that quantum mechanics could not be a complete theory. We will return to this issue later, but for the moment we will merely observe that the EPR conclusion is dependent on assumptions about reality which, while apparently obvious to a classical thinker, are in fact potentially unwarranted. It is, in truth, a realist paradox and, even amongst realists, requires resolution only by those who insist on metaphysical models of a certain type.

As observed earlier, simple entangled systems can easily be created by a number of physical processes. If, for example, two entangled photons are produced, they will continue to be entangled until the system is subjected to a measurement. However, tensor product spaces and Rule 5 can be extended to more than two components. Indeed, n systems can be described by an element of the appropriate tensor product space of the n Hilbert spaces involved. Further, if one system interacts with another, quantum theory predicts that the result will be further entanglement, which will be described by the Schrödinger equation evolution under the Hamiltonian of the interaction. Clearly, unless a system is isolated from all such interaction, it will gradually become more and more entangled with surrounding systems, until our simple state-vector description of it has become impossibly lost in the complexity of its environment. This process occurs to any non-isolated quantum system and is called *decoherence*; however, it happens faster the larger and more complex the isolated system and, while it may be possible to isolate a single-particle quantum system from interaction for some

time, there is absolutely no hope of doing so for anything approaching macroscopic dimensions.

Decoherence has some interesting implications. It can be shown that if local measurements are made on a distributed entangled quantum system, the observed part of the system may appear to behave classically (e.g. interference effects will disappear) even although, if sufficiently precise measurements were made on the system as a whole, its quantum nature would be apparent. If a qubit, for example, is left alone for a while and is not adequately isolated from its environment, it will experience decoherence and will apparently lose its quantum nature, at which point any computation it is involved in will fail. What has happened, in fact, is not that the qubit has stopped being a quantum system; it is simply that it has become so entangled with the environment that its local behaviour appears classical, and only impossibly complex measurements on it and its environment as a whole would reveal any quantum behaviour at all. Interestingly, this phenomenon casts some light on an curious puzzle: if the entire universe is composed of entangled quantum systems, why do we perceive objects at the macroscopic level which appear to be localized and classical? In a fascinating investigation, Joos and Zeh (1985) have shown that the classical world emerges because its observers are localized and only capable of making limited measurements on partial systems subject to quantum decoherence. It is arguable (d’Espagnat, 2003) that the classical world appears as it does merely because of our limitations as observers, and that therefore what we consider to be ‘objective’ in the classical sense is as much a function of us as of any underlying reality. This argument applies even to realism, with which it is entirely compatible, and points the way to something quite alien to pre-quantum notions of what is in fact real: the idea that the objective world may, in fact, be of a quite different character from the reality that we perceive through ordinary experience. Realist notions of this type are sometimes described as *far realist* to distinguish them from the more conventional classical ideas (*near realism*), which generally assume that what we see is a more-or-less good reflection of what really is.

Now, however, returning to the matter at hand, as discussed, a single qubit is not much use for implementing quantum algorithms and so quantum circuits with many qubits are of much more interest. Often, a group of n qubits will be employed together, and the full power of such an arrangement relies on us allowing these qubits to become entangled with each other while isolating the entire assembly, or *quantum register*, from the environment, as we subject it to appropriate quantum gate (unitary) transformations. Fortunately, the mathematics needed to describe a quantum register is exactly that needed to model entangled systems, namely that of tensor product spaces.

Beginning with a register of just *two* qubits, the fifth quantum rule says that the descriptive state vectors will be elements of the four-dimensional tensor product space $\mathbb{C}^2 \otimes \mathbb{C}^2$, which is isomorphic to \mathbb{C}^4 . Some states in this space are product states, but many are *entangled*. When the two-qubit system is in one of these entangled states, the

individual qubits are not in definite states of their own at all. From the general discussion above, if we take the standard basis of each space, the product states $|0\rangle|0\rangle$, $|0\rangle|1\rangle$, $|1\rangle|0\rangle$, and $|1\rangle|1\rangle$ are a basis of $\mathbb{C}^2 \otimes \mathbb{C}^2$. The state $|i\rangle|j\rangle$ is usually written $|ij\rangle$, so that any state of the two-qubit system can be written as

$$|\Psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle \quad (6.27)$$

or, equivalently, as a four-dimensional vector with the components a_{ij} ; again, if $|\Psi\rangle$ is normalized, the squares of the magnitudes of the coefficients must sum to 1. The set of $|ij\rangle$ is, unsurprisingly, generally called the *computational basis* of $\mathbb{C}^2 \otimes \mathbb{C}^2$. For even greater conciseness (especially when we go beyond the two-qubit case), Eq. 6.27 is often written as follows:

$$|\Psi\rangle = a_0|0\rangle + a_1|1\rangle + a_2|2\rangle + a_3|3\rangle \quad (6.28)$$

where a subscript 2 can be added to the kets if there is ambiguity about the dimension of the space under consideration.

A general vector in the space $\mathbb{C}^2 \otimes \mathbb{C}^2$ can be represented using a 4×1 matrix as discussed above:

$$\begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix} \quad (6.29)$$

Being a Hilbert space, a tensor product space such as $\mathbb{C}^2 \otimes \mathbb{C}^2$, defined in the quantum context, has hermitian observables and unitary transformations just like its constituents. If A and B are single qubit operators, then we can define a tensor product operator, $A \otimes B$, on $\mathbb{C}^2 \otimes \mathbb{C}^2$ in the obvious way by

$$A \otimes B |\Psi\rangle = \sum_{i,j=0,1} a_{ij} A|i\rangle \otimes B|j\rangle \quad (6.30)$$

Further, if A and B are hermitian (unitary), then $A \otimes B$ is also hermitian (unitary). This immediately allows us to construct two-qubit unitary operators such as $X \otimes X$ or $H \otimes I$. Such operators would take the 4×4 matrix forms (verification is straightforward but a little fiddly)

$$\begin{bmatrix} 0 & X \\ X & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \quad (6.31)$$

As an example, consider the action of the operator $H \otimes H$ on the basis product state $|0\rangle|0\rangle$. The effect of this is as follows:

$$H \otimes H |0\rangle|0\rangle = H|0\rangle \otimes H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (6.32)$$

$$= \frac{1}{2}(|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle) \quad (6.33)$$

Note that this is still a separable (product) state; in fact, it should be obvious that a product operator such as this will always carry a product state to another product state.

However, not all two-qubit unitary operators are products of single-qubit operators. As in the single-qubit case, we can contemplate trying to implement such operators in physical systems as two-qubit gates but, perhaps unsurprisingly, these are even more difficult to construct. Since compositions of unitary operators are always unitary, however, we can try to build up some transformations from quantum circuits and here an interesting result comes to the rescue.

It can be shown that a small set of quantum gates can be used to build any unitary operator on $\mathbb{C}^2 \otimes \mathbb{C}^2$ and, curiously, the set needs only one two-qubit gate, an operator called the *controlled NOT* or *CNOT* and defined by the matrix

$$\begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix} \quad (6.34)$$

or, in fully expanded form,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (6.35)$$

This gate ignores the basis states $|00\rangle$ and $|01\rangle$, but flips $|10\rangle$ and $|11\rangle$. The effect is sometimes expressed as

$$C_{ij}|i\ j\rangle = |i, i \oplus j\rangle \quad (6.36)$$

for which reason it is sometimes called the *quantum XOR* gate (Fig. 6.6). CNOT is not a product operator and does entangle its inputs. Note that the first qubit is never altered and is known as the *control qubit*. Clearly, a CNOT gate with the second qubit as the control is equally valid and may be denoted C_{ji} .

The ideas applied here to two-qubit registers can easily be generalized to n bits where the space involved would have dimension 2^n . However, it turns out that any unitary operator on an n -qubit register can also be constructed out of the two-qubit CNOT and a suitable selection of one-qubit gates, so that these, in essence, are the only components that we need—with one final exception.

Unitary operators define all the transformations that can be undergone by a quantum register if it is allowed to evolve without interacting with another system. There are two ways in which such interaction can

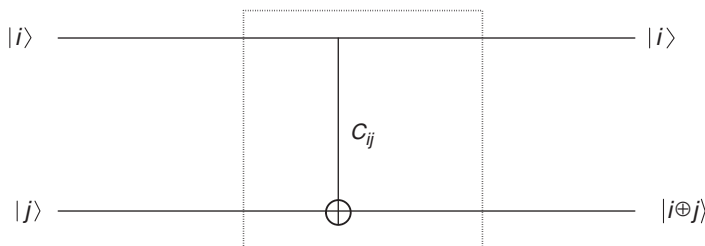


Fig. 6.6 The CNOT gate.

happen: one is unwanted interference from its surroundings—that is, decoherence—while the other is deliberate measurement. Recall again that in quantum theory all time evolution operations are unitary and reversible, whereas measurement causes a discontinuous irreversible change in the state vector. Once a quantum register has been transformed by all the unitary operations (gates) desired, it must be understood that its final state merely gives probabilities of what will be observed if someone looks at the state of the qubits. When a measurement is made of any qubit, the result will either be 0 or 1, the eigenvalues of the Z operator, which we used to determine the original standard computational basis: a superposition will never be observed. When a qubit is measured, that qubit’s state will thereupon be either $|0\rangle$ or $|1\rangle$. However, if only one qubit is measured, the others will still be in a superposed state and may continue to be processed if desired. To complete the set of components necessary for a usable quantum register, therefore, we need a device that can measure a single qubit relative to the computational basis. In quantum circuits, we denote such a device as a special gate called a *measurement gate*.

A measurement gate uniquely does not represent a unitary operation and is not reversible. Applying Z , which is unitary, to a qubit flips its second component but does not reduce the state to an eigenvector $|0\rangle$ or $|1\rangle$. There is, however, an important class of operator that does perform this task. For any vector $|\psi\rangle$ in a Hilbert space \mathcal{H} , there exists a linear operator P_ψ that takes any vector $|\varphi\rangle$ in \mathcal{H} and projects it on to the ray defined by $|\psi\rangle$, thus:

$$P_\psi|\varphi\rangle = \langle\psi|\varphi\rangle|\psi\rangle \quad (6.37)$$

P_ψ is called a (one-dimensional) *projection operator* or *projector*, and in a sense it measures how much of $|\psi\rangle$ ‘is in’ $|\varphi\rangle$ (see Fig. 6.7). In Dirac notation, it is often denoted thus:

$$P_\psi = |\psi\rangle\langle\psi| \quad (6.38)$$

for reasons that are apparent from the above.²

Projection operators are hermitian, but they are not invertible and so not unitary. They are, however, *idempotent*, which means that $P^2 = P$, or applying the operator twice is just the same as applying it once. They always have two eigenvalues, 0 and 1: the eigenspace for 1 is the ray defined by $|\psi\rangle$, while that for 0 is the set of all vectors perpendicular to $|\psi\rangle$. Projectors are not measurement gates, but they are related: if a qubit $|\psi\rangle$ is measured and the eigenvalue 1 found, the post-measurement state will be the projection of $|\psi\rangle$ on to $|0\rangle$; if -1 is found, the post-measurement state will be the projection on to $|1\rangle$. Of course, a projector is deterministic while a measurement is not, and so a measurement of an observable, A , is effectively a random application of one of the projectors on to the eigenvectors of A .

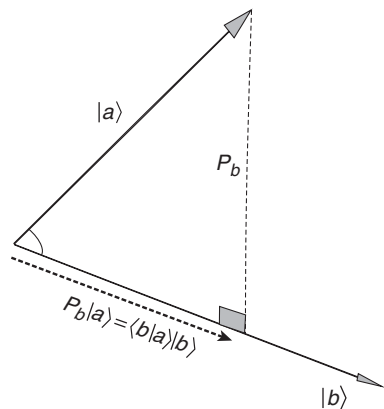


Fig. 6.7 Visualization of the projection operator.

²Note that the Dumaresq shown in Fig. 3.10 computes by analogue means a projection operation of this type.

6.5 Quantum computers

Without concerning ourselves at this point about how a physical quantum computer might be implemented, let us examine a general model of quantum computation. There are actually several such models that are ultimately computationally equivalent, but here we focus on the one that has gained the most widespread prominence and was used to design the first quantum algorithms. In this *quantum circuit model*, a quantum computer is assumed to have at its disposal a number of qubits arranged in two quantum registers: an *input register* of, say, n qubits and an *output register* of m qubits. The input register can ‘contain’ any n -bit binary word or any superposition thereof in the sense that it can be described by a state vector that is an element of \mathbb{C}^{2^n} . Likewise, the output register can contain any m -qubit value that can be described by an element of \mathbb{C}^{2^m} .

Now suppose that we wish to compute some chosen function $f(x)$ of x , where x and $f(x)$ are n -bit and m -bit binary integers respectively: this is ultimately a completely general computational task. The idea is that we initialize the input register to x and the output register to some selected value y (often $|0\rangle_m$) and then design a quantum circuit that can, through normal unitary time evolution, transform the combined input and output registers to a form where the value of $f(x)$ can be extracted.

The quantum circuit must be an implementation of some unitary operator (this is how all quantum systems transform, via the Schrödinger equation), U_f say, and so it must be invertible. However, not all functions are invertible (think of constant functions, for example), so we use a simple technique that allows us to map any function, invertible or not, on to a unitary operator and so a quantum circuit. This is achieved by having the operator act on all the $m + n$ input and output qubits as follows. The input register is left unchanged by U_f , while the output register is described by the state $y \oplus f(x)$, where the \oplus symbol denotes bitwise exclusive-OR. So we have

$$U_f |x\rangle_n |y\rangle_m = |x\rangle_n |y \oplus f(x)\rangle_m \quad (6.39)$$

There are two points of immediate interest: first, if we initialize y to $|0\rangle_m$, the output register contains $f(x)$ as desired; secondly, it is easy to check that U_f is its own inverse (this follows from the fact that $y \oplus (y \oplus f(x)) = f(x)$).

Of course, this all glosses over how a circuit can be designed to implement U_f for a given f but, in fact, this can be done for functions of interest by using networks of quantum gates, or even by customizing Hamiltonians as discussed earlier. If the above arrangement is used with x set to a product state from the standard basis, the quantum computer will deliver $f(x)$ for the chosen value just as a classical machine would. In fact, since this is a general model of computation, it follows that a quantum computer can compute any function that a classical computer can. The real power of quantum computation, however, is only revealed when x is set to a superposition of states from the standard basis.

From Eq. 6.33 above, it is clear that in the two-qubit case if each qubit is set to $|0\rangle$ and then a Hadamard operator is applied, the result is a balanced superposition of all the standard basis elements. Do note, however, that this is still a product state: there is no entanglement yet. This technique extends in the obvious way to the n -qubit case as follows:

$$H^{\otimes n}|0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle_n \quad (6.40)$$

where $H^{\otimes n} = (H \otimes H \cdots \otimes H)$, n times. So if we augment this n -qubit Hadamard operator with the m -qubit identity operator $H^{\otimes n} \otimes I_m$ and apply this to the input and output registers initialized to $|0\rangle_{m+n}$, the input register is placed into the balanced superposition of Eq. 6.40 while the output remains at $|0\rangle_m$. If U_f is now applied, then the final state of the registers is as follows:

$$U_f(H^{\otimes n} \otimes I_m)|0\rangle_n|0\rangle_m = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle_n |f(i)\rangle_m \quad (6.41)$$

This looks as if the output register is in a quantum superposition of all the function evaluations for all possible n -bit values of i and that the feat of simultaneously computing 2^n values of f has been achieved in the time for U_f to operate once (Fig. 6.8). This is ultimately the source of what is called *quantum parallelism*. However, there is a problem!

Although, in principle, the 2^n values of f have been computed simultaneously, there is no way of retrieving all the results. If we subject the m qubits of the output register to a measurement (pass the final state of the output register through an array of m measurement gates), exactly one result, $f(i)$, will be returned with the single value of i involved being entirely random. Furthermore, once such a measurement is made, the system is left in a state $|i\rangle_n |f(i)\rangle_m$, which is no longer a superposition of the other results and gives us only one value of f . An obvious trick might be to try to make many copies of the output state before passing them through measurement gates, but it turns out that such a thing is impossible because of a result called the *no-cloning theorem*. This is not hard to prove (see, e.g., Mermin, 2007) and, in the quantum computing context, it states that there is no way of copying the state of one set of n qubits to another similar set in some other initial state. Fortunately, however, by manipulating the input and output registers in various clever ways both before and after application of U_f , it is possible to extract useful information about the function f much more rapidly than could

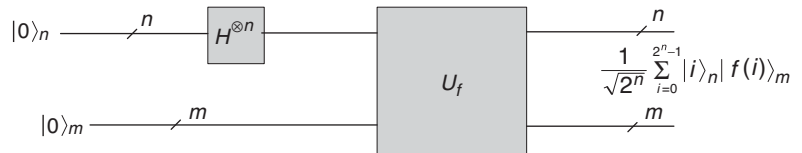


Fig. 6.8 Generalized function computation.

be done on a classical computer. We shall see an example of this in the next section.

6.6 Quantum algorithms

A number of clever quantum algorithms have been found, of which the most famous is the one discovered by Peter Shor in 1994 which is capable—given a quantum computer, of course—of factoring an integer exponentially faster than any known classical equivalent. Shor’s algorithm is relatively involved and a discussion of the details too lengthy to include here. Instead, we will look at a more concisely describable, and almost equally famous, algorithm discovered shortly afterwards by Lov Grover (Grover, 1996), who proved that a quantum computer could also be used to speed up certain search operations beyond anything possible by classical means.

Consider a function $f(x)$, where $x \in \{0,1,\dots,N\}$ such that $f(x) = 0$ except at some unknown value of x —say, $x = a$ —where $f(a) = 1$. Here, we assume there is only one value for which f is non-zero, although that constraint can be relaxed in more generalized versions of the algorithm. However, nothing else is assumed to be known about the function except that there is a computational means available, which we will call an *oracle*, that will compute it for any selected input value of x . The problem is quite simply to find the value of a .

If we can assume nothing except the above about the function, then the only classical approach is to search through the possible values of x , applying the oracle once for each one, until a is found. On average, a will be discovered after $N/2$ invocations by a classical search, but Grover showed that a quantum search can be done much more efficiently, with only an average of \sqrt{N} calls. For a large search, this is a very useful gain (although not as overwhelming as that in Shor’s algorithm). For example, if $N = 1\,000\,000$, a classical search will require on average 500 000 evaluations of f , while a quantum search only needs 1000.

The details of the algorithm are reasonably easy to describe. We will assume that $N = 2^n - 1$ for some n , in which case the quantum computer required will have an input register of n qubits; an output register of just one qubit is also needed. The input register is initialized to $|0\rangle_n$. We assume that f is available in the form of a unitary operator O (the oracle) that acts on the quantum registers as described in Eq. 6.39:

$$O|x\rangle_n|y\rangle_m = |x\rangle_n|y \oplus f(x)\rangle_m \quad (6.42)$$

Now, if instead of setting the output qubit to zero it is initialized to $H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ then, when we apply O , we get

$$O|x\rangle_n H|1\rangle = O|x\rangle_n \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = (-1)^{f(x)}|x\rangle_n H|1\rangle_m \quad (6.43)$$

This means that every time O is applied it leaves the output register unchanged; if the input register is in the basis state $|a\rangle_n$, its phase is changed; in any other basis state it is unaltered.

The first step is carried out just once and involves applying an n -qubit Hadamard operator to the input register as described in Eq. 6.40. After this, the algorithm enters an iterative stage, each iteration consisting of four steps as follows:

1. O is applied.
2. $H^{\otimes n}$ is applied to the input register.
3. A special phase shift operator, P , is applied, that changes by π the phase of all standard n -qubit basis states (equivalent to multiplying them by -1) except $|0\rangle_n$, which it leaves unchanged.
4. $H^{\otimes n}$ is applied to the input register again.

This sequence is repeated an integral number of times, R , where $R \leq \frac{\pi}{4}\sqrt{N}$, and then the input register is measured. The algorithm does not guarantee that the measurement will yield a , but it will do so with a high probability (this can be checked with a final invocation of O). If it does not, the entire sequence must be run again. This probabilistic nature is a general feature of quantum computation and repeated runs of most algorithms may be needed to obtain a solution.

We now make a few observations that will explain why Grover's algorithm actually works. First, note that the operator P can be expanded as follows:

$$P = -I_n + 2|0\rangle_n\langle 0|_n \quad (6.44)$$

The application of the four operators in the order given is sometimes called the Grover operator, G , so

$$G = H^{\otimes n} P H^{\otimes n} O \quad (6.45)$$

Remembering that H is its own inverse, it is now easy, via some trivial operator algebra, to prove that

$$H^{\otimes n} P H^{\otimes n} = 2|\Phi\rangle_n\langle\Phi|_n - I_n \quad (6.46)$$

where

$$|\Phi_0\rangle_n = H^{\otimes n}|0\rangle_n \quad (6.47)$$

which we note is the state the input register starts out in after initialization phase. Now we know that this state is an equally weighted linear combination of all the $|x\rangle_n$, one of which is $|a\rangle_n$. We can thus write $|\Phi_0\rangle_n$ as

$$|\Phi_0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (1 - f(i)) |i\rangle_n + \frac{1}{\sqrt{2^n}} |a\rangle \quad (6.48)$$

$$= \frac{1}{\sqrt{2^n}} |\phi\rangle + \frac{1}{\sqrt{2^n}} |a\rangle \quad (6.49)$$

where $|\phi\rangle = \sum_{i=0}^{2^n-1} (1 - f(i)) |i\rangle_n$ contains all the terms that are not a and is therefore orthogonal to $|a\rangle$ (remember that the $|i\rangle_n$ form an

orthonormal basis of the product space): note, however, that although $|a\rangle$ is normalized, $|\varphi\rangle$ is not. By the Born Rule, the probability of obtaining the desired value, a , if we measure the input register at this stage, is just the square of the amplitude of $|a\rangle$.

What the Grover operator does now is to transform $|\Phi_0\rangle_n$ to a new vector $|\Phi_1\rangle_n = G|\Phi_0\rangle_n$, then $|\Phi_2\rangle_n = G^2|\Phi_0\rangle_n$, and so on. The crucial point is that, at each stage, the amplitude of $|a\rangle$ is boosted (in the sense that its magnitude gets larger) from its initial very low value ($1/\sqrt{2^n}$), at the expense of the amplitudes of everything else. It can be shown that after R iterations, the magnitude of the amplitude of $|a\rangle$ has become close to 1 and all the other amplitudes are correspondingly close to 0. At this point, a measurement of the input register will yield $|a\rangle$ with a very high probability. Note that if iterations are continued beyond R , the amplitude of $|a\rangle$ will begin to decrease again, so it is important to terminate the process at the right time.

The method can easily be extended to search a list where there is more than one solution to $f(x) = 1$. If there are M solutions in a space of N elements, the process is terminated after R steps, where $R \leq (\pi/4)/\sqrt{N/M}$. The algorithm is computationally optimal in the sense that any algorithm that searches using the oracle operator O for a function, as specified above, will need to apply O at least as often as here (Zalka, 1999). Grover's approach provides just a *quadratic* speed-up over classical counterparts and while this is invaluable if N is large, other quantum algorithms, such as Shor's, provide a much more significant advantage. Space precludes us from considering any further examples, but the interested reader is referred to Mermin (2007) for a more detailed exploration. Here, we will close this brief examination of this fascinating area by looking at the practical obstacles facing implementation of a real quantum computer, and the philosophical implications for our understanding of the nature of the world.

6.7 Building a quantum computer

Grover's algorithm suggests that the key to a successful quantum algorithm is to ensure that the result of the computation is encoded in a suitable weighting of the amplitudes of the basis elements in the final resultant state of the quantum registers. It also shows, incidentally, that obtaining the result may well involve measuring the input rather than the output register. However, when that final measurement is made, a quantum computer will, in general, yield a correct answer only with a certain probability, and to retrieve it may well require multiple runs.

As might be suspected from this, successful quantum algorithms are hard to design. To be useful, obviously, a quantum algorithm must offer some significant advantage over an equivalent on a classical machine and to engineer this is not a trivial exercise. Notable examples such as Grover's and Shor's algorithms are still relatively few, and in fact

Shor himself has expressed disappointment at the limited success so far, examining possible reasons (Shor, 2004).

In terms of computational complexity, factoring is in complexity class NP, but it is not NP-complete and, despite much effort, nobody has been able to demonstrate that quantum computers can solve problems in the latter category in polynomial time. We have already seen that a quantum computer cannot search a space of size N in less than $O(\sqrt{N})$ time. As for hypercomputation, attempts have been made to use quantum computing to solve Hilbert's Tenth Problem (Kieu, 2003): the famous challenge to construct a general method for deciding whether a given Diophantine equation has an integer solution. It is known that this problem can be solved if and only if the halting problem can be solved, so success would herald the potential creation of a hypercomputer. However, Kieu's solution, which harnesses the *quantum adiabatic theorem*, has been convincingly challenged by various authors (see, e.g., Smith, 2006a).

But design of algorithms is not the major problem: that, rather, is the question of how to build the hardware. Quantum digital computers depend on the controllable entanglement of many qubits and, as we have seen, this can only be achieved if the entire system can be prevented from experiencing decoherence for long enough to allow a computation to complete. Yet interaction with the environment is inevitable for any quantum system and even if every effort is made to minimize the effect, it will introduce noise and thus error into the process. Bounding this error is one of the key issues that must be confronted if quantum computing is to become a practical proposition. Several technologies are being extensively researched as potential candidates for such practical implementation and it is not yet clear which will prove the most successful. DiVincenzo (2000) has listed the requirements that any such technology must meet in a commendably succinct list: a scalable physical system with well-characterized qubits; the ability to initialize qubits; long decoherence times; a universal set of implementable quantum gates; and a qubit-specific measuring capability.

Many technologies have been investigated with a view to attaining these goals. Some, such as *liquid nuclear magnetic resonance* (NMR), are easier to use at present for very limited experiments with only a few qubits, but are believed to have poor scope for scaling to practically useful sizes (realistically, at least thousands of qubits will be needed); others may scale much more successfully but are harder to get working in the first place. It has to be admitted that at present, although there are many promising avenues, it is unclear either when or along what technological path the first seriously usable quantum computers will emerge. Here there is only scope, briefly, to list, with no pretence at full inclusivity, some of the approaches that have been tried or proposed:

1. Liquid NMR, mentioned above, uses the spin contributed by the nuclei of carefully chosen isotopes embedded in customized molecules to implement qubits. The system is operated in liquid state at room temperature and the qubits are manipulated

(addressed and operated on) using external magnetic fields. However, measuring a single qubit is not possible and so ensembles of qubits have to be assembled together and averages taken. This would not be problematic if all the copies of a qubit could be described by the same pure quantum state but, in fact, this level of control is not available and the copies are therefore a mixture of different states requiring adaptation of quantum algorithms to produce results. Liquid NMR has successfully been used to create systems of about 10 qubits, but the addressing technique used makes scaling beyond this very difficult. Even worse, thermal noise is too great to allow the qubits to be controllably entangled and it seems therefore that, although small machines have proved relatively easy to construct, thanks to the maturity of NMR technology, they do not satisfy the full requirements for universal quantum computing and their true nature is really semi-classical (Menicucci and Caves, 2002).

2. *Ion traps* use the ground and first excited states of electrically confined ions, instead of spins, to implement qubits. Here, lasers can be used to effect changes of qubit state as well as implementing a set of quantum gates. Genuine single-qubit implementations are possible and coherence times are good, but gates are relatively slow and, as more ions are added to a trap, they become more difficult to address individually: a possible solution might be multiple traps coupled by photons, but this remains under investigation.
3. *Solid-state NMR* may avoid some of the limitations of the liquid-state approach, but is harder to pursue. Here, the nuclei implementing qubits are impurity ions embedded in a matrix of silicon 28 (which has no nuclear spin of its own). The whole system is operated at low temperature ($\sim 1\text{--}2\text{ K}$), where coherence times are good. There is also, in principle, more scope for individual addressing of many qubits using small electrodes. However, placement of the impurity atoms requires atomic precision and readout of the individual spins is still a difficult problem.
4. Superconducting systems of various types have been proposed, one with a qubit design based on the magnetic flux passing through a superconducting ring containing a Josephson junction. In this design, the ring is arranged to have a double-well potential, with the two minima corresponding to the two states of the *flux qubit*. Magnetic fields are again used to implement gates, while measurement relies on the use of *SQUIDS* (*superconducting quantum interference devices*). Flux qubits are only one proposal for the use of superconductive technology to implement quantum computers, but all rely on low-temperature implementation. In each case, however, there are great difficulties to be overcome before any prospect of reasonable scaling might be envisaged.
5. Qubits can also be based on *quantum dots*, structures that can be grown on certain semiconductor materials and which confine electrons in traps that then quantize their allowed energy levels.

Either energy levels, via so-called excitonic states, or electron spin can be used to implement qubits within dots and control can be arranged via lasers or magnetic fields. Electron spins have longer decoherence times than excitons, although not as long as those of nuclei. However, electron spins do respond more strongly to magnetic fields than in the nuclear case and, consequently, gates implemented this way will be faster. Nonetheless, there are significant difficulties with measurement of individual dots that remain to be solved before any such scheme might be practical.

6. A proposal that is strikingly different from all of the above is *linear optics quantum computing (LOQC)*, which uses photons as qubits, relying on polarization along two orthogonal axes (say, horizontal and vertical) to implement the two standard basis states. The difference here is that photons move and so quantum gates and wires are more like those of a familiar classical machine, although composed of optical rather than electronic elements. There are some advantages: photons are easy to create and initialize and have great resistance to decoherence. Inevitably there are difficulties too: first, this very resistance to decoherence makes it very difficult to entangle multiple photons; secondly, photons are destroyed as soon as they are measured, and while apparently ideal for processing (given suitable optical gates), are not well suited for storage. While these difficulties are not necessarily insurmountable, they make the construction of a quantum computer on these principles especially challenging.

Whatever the technology that ultimately succeeds, and regardless of how long the average decoherence time that it may allow, a quantum computer will face a familiar general problem, in that random errors can affect any calculation at any time. Just as in a classical machine, these errors may occur during storage or gate operation and, if not fixed, will become cumulative. In the classical case, things are relatively simple, in that an error always amounts to one or more bits being accidentally flipped and this can be protected against by the expedient of adding redundancy bits to enable error detection and correction for all syndromes with probabilities of occurrences above some acceptable threshold. For example, instead of storing or transmitting one copy of a bit, we can store or transmit three. The only allowable representations are then 000 and 111, so any single-bit error can be not only detected but corrected (with maximum likelihood, by choosing the legal word with the closest Hamming distance).

Of course, replicating bits twice is a very crude and inefficient error-correcting code compared to those used in practice but, in effect, they all work on the same principle and it serves to illustrate a key point; namely, that this approach does not work for qubits! A qubit will in general, at any time, be described by a state that is a superposition of $|0\rangle$ and $|1\rangle$, as in Eq. 6.10. However, it is much more vulnerable than a classical bit because, while the latter can only be flipped, a qubit can suffer a number of different types of error: for example, in addition to a

bit flip, which would take

$$a|0\rangle + b|1\rangle \quad \text{to} \quad a|1\rangle + b|0\rangle \quad (6.50)$$

there might be a phase error such as

$$a|0\rangle + b|1\rangle \quad \text{to} \quad a|0\rangle - b|1\rangle \quad (6.51)$$

Other possibilities include continuous, as opposed to discrete phase, errors and even accidental measurements that effectively collapse the superposition to one of the basis states.

Note immediately that the classical trick of making multiple redundant copies is not available in the case of qubits because of the no-cloning theorem. At first sight, the constraints appear to militate against any scheme of effective quantum error correction but, surprisingly, all the above errors can be successfully protected against, admittedly at some cost in additional qubits. The general approach is to entangle the qubit to be protected with some additional redundant qubits, using CNOT gates and then take measurements to see whether or not an error has occurred to any of them, applying a correction to any one affected. To see how this works, we will examine here a simplified version of the idea, which corrects only bit flips using a total of five qubits, two to carry copies of the original and two to allow measurements to be taken.

To protect a qubit in the state $|\psi\rangle = a|0\rangle + b|1\rangle$ from bit flip errors, two additional qubits are initialized to $|0\rangle$ and then passed through the y inputs of two CNOT gates controlled by the original qubit. The three-qubit system is now in the state

$$|\psi_1\rangle = a|000\rangle + b|111\rangle$$

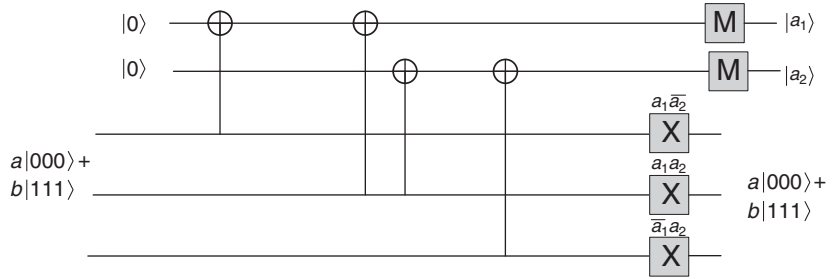
Note immediately that this is an entangled state and not a cloned version of $|\psi\rangle$ (which would be a product state). If a bit flip occurs on any one of the three qubits (which we will label 3, 2, and 1 from left to right), the corrupted state will be of a form such as

$$|\psi_e\rangle = a|010\rangle + b|101\rangle$$

Note that to flip the i th qubit is equivalent to operating on it with an X (NOT) gate while leaving the other qubits alone: in the above example, the operator required is just $I \otimes X \otimes I$, which we can denote simply as X_2 (X applied to the second bit).

Now introduce two further *ancillary* qubits, $|a_1\rangle$ and $|a_2\rangle$, both initialized into the $|0\rangle$ state (Fig. 6.9). Pass $|a_1\rangle$ through two CNOT gates controlled, respectively, by qubits 1 and 2 and pass $|a_1\rangle$ through another two CNOT gates controlled by qubits 2 and 3. Then measure $|a_1\rangle$ and $|a_2\rangle$ to determine their states. If they are both $|0\rangle$, there has been no bit flip. If $|a_1\rangle$ is $|1\rangle$ and $|a_2\rangle$ is $|0\rangle$, then qubit 1 has been flipped, and so on. Once it has been determined that there has been a bit flip and to which qubit it applies, that qubit can be selectively corrected with an X (NOT) gate. Of course, the ancillary qubits must now be reset if they are to be used again.

Fig. 6.9 Basic error correction.



This illustrates the principle of quantum error correction, but is, in reality, a clumsy and restrictive scheme, which is not able to rectify errors other than bit-flips. It turns out that just as a bit-flip can be represented as an X operator, the most general type of error is a linear combination of I , X , Y , and Z operators, and the above ideas for tackling X can be adapted to correct the other syndromes as well. The first code that was capable of achieving this was discovered by Peter Shor (1995), but required nine physical qubits per logical one and has been significantly improved upon since. It is now known that the most efficient code capable of correcting all single-qubit errors needs five qubits; however, although this is the theoretical minimum, it is cumbersome to implement and a more practical scheme is the seven-qubit code discovered by Andrew Steane (1996).

The existence of these codes is in a sense quite surprising and makes the prospect of workable quantum computers much more likely if the technological hurdles to qubit and gate implementation can be successfully negotiated. It is worth pointing out that the quantum circuit model is not the only approach to quantum computing under investigation. In over two decades since Deutsch's key paper, other models such as the *quantum adiabatic computer* have also been shown to be computationally equivalent to the circuit model (Aharonov *et al.*, 2007, Mizel, 2007) to within polynomial overhead. The adiabatic approach (Farhi *et al.*, 2000), based on controlled adiabatic Hamiltonian evolution, has been the subject of some effort. It is the basis not only for Kieu's hypercomputation claim, but also for the technology behind the controversial announcements of quantum computer implementations by the Canadian company *D-wave*. More speculative still is *topological quantum computing* (Freedman *et al.*, 2003), which is based on the use of 'anyons' (Wilczek, 1982), two-dimensional quasi-particles channelled through gates which are formed from *braids* of world-lines in a three-dimensional space-time. Quasi-particles are pseudo-physical mathematical constructs, of which the commonest examples are the 'holes' of semiconductor physics and the 'phonons' of solid-state physics; braids are a topological generalization of the familiar twisted-string forms of the same name. Theoretically, a topological quantum computer would be much more resistant to decoherence than one based on quantum circuits. However, at present this line of investigation remains

at an even more hypothetical level of development than the rest of a discipline that is still some way from bearing practical fruit.

Finally, we may ask whether quantum computing sheds any light on how the quantum theory should be interpreted. Recall that the theory is a mechanism for predicting probabilities of the results of future observations, and that it has no ontology of its own beyond that of the observer and the information. It can be viewed as an entirely instrumentalist tool and is consistent with an anti-realist metaphysics—although, of course, any anti-realist position needs to address the question as to the nature of the ‘something that says no’ (d’Espagnat, 2006), to many elegant human theoretical constructions. For a realist, however, quantum mechanics does clearly place constraints on any prospective model of observer-independent reality, since such a model must make the same observational predictions; however, the theory itself is of little help beyond this point. It does, of course, predict intersubjective agreement between different observers (*weak objectivity*), but this in no way implies that a strongly objective description of nature is attainable.

Bohr was willing to stop there. In his opinion, if the concept of objective reality (that which exists entirely independently of all observing minds) makes sense at all, its ontological nature could only ever be accessible via its epistemological interface and so it is scientifically meaningless to ask questions that can never be answered about what, so to speak, lies beneath. Whether Bohr’s personal belief was truly anti-realist, or more a Kantian view,³ is hard to tell, for the consequences of these metaphysically divergent positions are not really distinguishable at the level of scientific investigation. In any case, his highly influential view of quantum mechanics—succinctly, if not entirely accurately, encapsulated by David Mermin’s famous phrase, ‘Shut up and calculate’—is commonly referred to, in honour of Bohr, as the *Copenhagen interpretation*. It emphasizes the theory as a means of *human* investigation and eschews the prospect of using science to uncover any objective reality that may (or may not) underlie our experiences. However, as ‘interpretations’ of quantum theory are normally characterized by their models of an axiomatically presumed objective reality and Copenhagen explicitly denies that such models have any scientific validity, some who take the Bohrian view claim that the title is a confusion. Rudolph Pieiris puts it thus (Davies and Brown, 1993):

I object to the term Copenhagen interpretation . . . because this sounds as if there were several interpretations of quantum mechanics. There is only one . . . when you refer to the Copenhagen interpretation what you really mean is quantum mechanics.

There is an element of definitional ambiguity here for sure (it all depends on what we mean by an interpretation), but the complaint is perhaps not unlike that of an atheist or agnostic accused of being, him or herself, an adherent of just another religion.

³Immanuel Kant accepted that objective reality, the *noumenal world* as he called it, does exist, but that we can never determine its nature from phenomenal experience. He was in this sense a *far realist*.

For many scientists and philosophers, however, the weakly objective nature of quantum theory is deeply troubling, especially given its foundational position in the entire scientific realm, to say nothing of its overwhelming empirical success. Indeed, it is hard to overestimate the significance of its, so far unrefuted, Completeness Hypothesis, on the ambitions of modern science fully to understand the world: it is, in effect, the philosophical equivalent of a ‘no-go theorem’. In d’Espagnat’s view (d’Espagnat, 2006), a change from the classical worldview to one based on a weakly objective quantum mechanics is a process that is comparable to the Copernican revolution and, like the latter, will take decades, perhaps even centuries, fully to establish its influence across the scientific domain—especially in areas where the objects of study are macroscopic and, on the surface, classical in aspect.

When all is said and done, a key motivation for those who choose to study science, at least as it has been seen since The Enlightenment, is not merely manipulation of the human environment, but insight into the true nature of the world. To be forced to accept that this goal is either meaningless or inaccessible to scientific method would be a bitter pill to swallow, and the response to this philosophical threat has been exactly what typically occurs in the absence of information: speculation and the adoption of positions of faith. We have already mentioned the work of David Bohm, who built upon ideas originally proposed by de Broglie concerning *pilot waves*, but his model seems contrived and unattractive to many, even among those who find Copenhagen unsatisfactory.

In 1957, Hugh Everett III proposed a striking interpretation that attempted to reconcile the awkward fact (for proponents of strong objectivity) that measurement appears in the quantum theory as a quite distinct process from the deterministic unitary evolution of an unmeasured system. This dichotomy is entirely unsurprising if the epistemological nature of the theory is accepted at face value, since measurement is then associated with a discontinuous change in the knowledge of the observer, but, to many, the idea that the best science can do is to describe the experiences of observers is unappealing, to put it mildly. To rescue the notion of a knowable, mind-independent world requires something in the way of additional metaphysical assumptions. Bohm tackled this by introducing a new hidden ontological element, the *quantum potential*; Everett instead proposed accepting the quantum formalism at face value and then taking the further step of reifying it—in other words, claiming that it is a direct description of objective reality. As the short discussion below will demonstrate, this is not a scheme without many philosophical and pragmatic difficulties, but it does have some immediate benefits. One of these is that it naturally gives meaning to the concept of a quantum state for the entire universe, an idea that makes no sense in a weakly objective context, since there are no observers outside the universe, and yet that is a necessary assumption of much current cosmological speculation.

Unfortunately, the device of reifying the quantum state, called the *relative states* approach by Everett, brings with it considerable ambiguity,

and he himself did not succeed in resolving this. When a measurement of a superposition of eigenvectors takes place, rather than one outcome occurring at random, the relative states interpretation holds that all outcomes occur and the state vector does not undergo any discontinuous irreversible change such as that suggested in Rule 2 (as imposed by a measurement gate, for example). But this is puzzling since, as observers, one outcome is exactly what we experience. What, ontologically (and remember that relative states is an ontological interpretation), has happened to the other possibilities?

Since Everett's published work left this question open, a number of suggestions have been put forward, of which the best known is that popularized later by Bryce DeWitt and called the *Many Worlds Interpretation (MWI)*. Here, all outcomes occur, but each happens in a different universe. The world thus splits or branches at each such point—and so objective reality, in fact, consists not of a single universe, but of a *multiverse* that is constantly growing at every instant as new quantum splits occur. Of course, this solution raises many questions. What exactly causes the universe to split? Since the aim is to reduce measurement to the status of a quantum interaction, by definition there can be nothing special about the former and, frankly, given the nature of entanglement, it is hard to identify a clear definition of the latter. Also, as a quantum state can be written as a superposition of vectors from different bases generated by incompatible operators, which basis governs the split?

MWI enthusiasts have struggled with these issues for many years (and have claimed some success), but the difficulty of achieving answers that are convincing in the context of the original aims of the interpretation (observer-independence key among them), has led others to suggest that relative states should be interpreted as implying that just one universe exists, but that it evolves directly in accordance with the Schrödinger equation and so is always in a macroscopic superposition of eigenstates of every observable. The problem here is then to explain why an observer, whose brain must be similarly in a superposition, experiences a single outcome when a measurement is made. Following d'Espagnat (1976), we merely note here that, given these assumptions, the two obvious ways out of the conundrum are: to accept that consciousness is, in some sense, special, so that uniquely it attaches to one branch even though everything else remains superposed; or that the consciousness itself also enters a superposition—yet either one element of the superposition is favoured, as the observer still seems to perceive a single definite outcome, or we assume that the consciousness really does split even though nothing else does. These variants are sometimes referred to as Many Minds interpretations and it has to be said that none of them is without some conceptual difficulties. Deutsch invokes MWI as, in his view, the only convincing explanation as to why quantum parallelism is possible. In his key paper Deutsch (1985), explicitly adopts Many Worlds language and justifies it thus:

In explaining the operation of quantum computers I have, where necessary, assumed Everett's ontology. Of course the explanations could always be

‘translated’ into the conventional interpretation, but not without entirely losing their explanatory power. . . . The Everett interpretation explains well how the computer’s behaviour follows from its having delegated subtasks to copies of itself in other universes. On the days when the computer succeeds in performing two processor-days of computation, how would the conventional interpretations explain the presence of the correct answer? Where was it computed?

Many MWI enthusiasts agree, but not everybody is convinced. David Mermin, who has philosophical objections to what he describes as the ‘reification of abstractions’ (Mermin, 2009), observes tellingly that quantum parallelism is not all it seems. When a unitary operator is applied to an input register initialized to a Hadamard superposition, it appears (Eq. 6.41) that the function is evaluated at all domain points simultaneously, but—and it is a major ‘but’— *there is no way of accessing the information* except, underwhelmingly, for one random element of the set of inputs. In the words of Mermin (2007):

If there were a way to learn the state of such a set of Qbits, then everyone could join in the rhapsodic chorus. (Typical verses: ‘Where were all those calculations done? In parallel universes!’, ‘The possibility of quantum computation has established the existence of the multiverse.’ ‘Quantum computation achieves its power by dividing the computation among huge numbers of parallel worlds.’) But there is no way to learn the state. The only way to extract any information from Qbits is to subject them to a measurement.

Who is right? Deutsch’s Many Worlds is perfectly consistent with quantum mechanics but, when all is said and done, it has to be an arbitrary choice. Preference of interpretation is not driven by the quantum theory itself, nor by empirical results, but rather by the metaphysical leanings of the chooser. For this reason, arguments about interpretations, like most philosophical debates, can go on and on. For anyone who finds the open realism of Copenhagen too hard to swallow, a rich array of interpretative positions is available to choose from, all compatible with the predictions of quantum mechanics. Ironically, one position that is no longer on offer, at least while maintaining logical consistency, is the one that has shaped the world of the past 300 years and that dominated unchallenged at the time of Lord Kelvin’s supposed remark: classical materialism.

Since the time of Copernicus and especially since the Enlightenment, progressive thought has gradually but inexorably demoted the observer from the central role occupied in ancient philosophies, to a position of almost total insignificance when placed against the enormity of cosmological space and time. Ironically, quantum mechanics threatens to bring this diminished character back to centre stage once again, and the struggle for a strongly objective theory is seen as crucial by those who, perhaps perspicaciously, sense a serious threat to what has become known as ‘modernist’ thought. That such theories are possible, in multiplicity, is without doubt; whether they have any basis in reality is entirely unproven and, if Completeness holds, unprovable. It is perhaps

fitting to close with a famous quote attributed to the great quantum physicist John Wheeler:

At the heart of everything is a question, not an answer. When we peer down into the deepest recesses of matter or at the farthest edge of the universe, we see, finally, our own puzzled faces looking back at us.

6.8 Physical limits to real number representations

In Section 7.5, we discuss some objections to the idea of infinite precision reals. In this, we had recourse to the argument that the theory of relativity in conjunction with quantum theory implies a granularity to space that undermines the notion of the divisible real number line. However, all other considerations aside, the idea of being able to represent real numbers physically to any desired degree of accuracy turns out to be in direct contradiction with quantum mechanics alone. To understand the significance of this statement, it is important to recognize that quantum theory and its extension, quantum field theory, render obsolete and replace classical physics, and that this has been so comprehensively demonstrated empirically that it is beyond reasonable doubt. In its fundamental (micro-)nature, the world is not classical, and classical models can never be more than approximations applicable in a limited domain. Recent developments in quantum theory suggest that the classical appearance of the world is, in fact, an artefact of decoherence, whereby the state vector of a quantum system becomes entangled with the environment (Joos and Zeh, 1985; Omnes, 1994), and that consequently quantum mechanics can indeed be seen as a universal theory governing macroscopic as well as microscopic phenomena. In what follows, we use only the basic axioms of standard quantum theory (d’Espagnat, 1976); in particular, those governing the deterministic time-evolution of an undisturbed quantum system according to Schrödinger’s equation and the discontinuous change of state vector experienced when a system is measured. From these principles, it is easy to demonstrate that there are fundamental limits on the accuracy with which a physical system can approximate real numbers. Suppose that we wish to build an analogue memory based on setting, and later retrieving, some property of a material body, A . To be suitable for analogue representation of real values to arbitrary accuracy, any such property must be continuously variable and so, according to quantum theory, can be represented as a Hermitian operator with a continuous spectrum of real eigenvalues. The most obvious approach to storing a real number in analogue fashion would be to use a positional coordinate of A , with some appropriate degree of precision, as the measurable property. Since only one coordinate is required, in what follows we will assume that A moves in only one dimension and that the value stored is given by its x coordinate, an eigenvalue of the x -position operator \mathbf{X} .

It is clear that any such system will have limits imposed by quantum mechanics: the aim here is to establish just how those limits would constrain any conceivable technology. As a first approximation, assume that A is a free body and is not therefore placed in a potential field. A natural scheme might store a real value, say, x_V , at time t_0 , by placing A at a point, at distance $x = x_V \pm \Delta x$ from some origin. Δx is the acceptable limitation on the precision of the analogue memory, determined by the physical length of the device and the number of values it is required to distinguish. If 10^R real values ($R \times \log_2 10$ bits) are allowed and the maximum value is L (the length of the device), then

$$\Delta x = \frac{L}{2 \times 10^R} \quad (6.52)$$

We will denote the interval $[x_V - \Delta x, x_V + \Delta x]$ by I_V .

In quantum-mechanical terms, just prior to t_0 , A is described by a state vector $|\psi_0\rangle$ (Dirac's formalism). Placing A at the chosen point involves 'collapsing' $|\psi_0\rangle$ into a new state confined to a positional subspace spanned by the eigenkets $|x\rangle$, with $x \in I_0 = [x_V - \Delta x_0, x_V + \Delta x_0]$. Δx_0 represents the accuracy of the measuring device and it is essential that $\Delta x_0 < \Delta x$ so that $I_0 \subset I_V$. Define $K > 1$ by $K = \Delta x / \Delta x_0$.

This process of 'state preparation' is entirely equivalent to performing a measurement on A that leaves it somewhere in the required subspace. Unfortunately, any actual measurement has a non-zero probability of failing to yield a result in I_0 . In fact, the probability of success is given by

$$P(x_V - \Delta x_0 < x < x_V + \Delta x_0) = \int_{x_V - \Delta x_0}^{x_V + \Delta x_0} |\langle x | \psi_0 \rangle|^2 dx^2 \quad (6.53)$$

It is reasonably easy, however, to circumvent this problem. Since the store operation, being a measurement, returns a positional value, it is easy to tell at once if it has failed (if the value lies outside I_0) and we can assume that any number of further attempts can be made until success is achieved. For the sake of simplicity, suppose the store operation succeeds on the first attempt at time, t_0 , whereupon the new state vector of A is given by

$$|\psi_s\rangle = \frac{1}{N} \int_{x_V - \Delta x_0}^{x_V + \Delta x_0} |x\rangle \langle x | \psi_V \rangle dx \quad (6.54)$$

where N is a normalization constant. In wave-mechanical terms, this describes a wave packet confined to the region I_0 . From the postulates of quantum mechanics, at a time immediately following t_0 , a second measurement on A will also retrieve a value within I_0 ; however, if left undisturbed, $|\psi_s\rangle$ will evolve deterministically, according to the Schrödinger equation and a later measurement will have no such guarantee. The Schrödinger equation can be solved analytically for certain shapes of wave packet, such as the Gaussian, but a more general argument is presented below, applicable to a packet of any form. The conclusions are universal and the argument can be extended easily

to a packet in a locally constant potential field (e.g. generated by neighbouring atoms). The key point is that, as soon as the wave packet is formed, it begins to spread (dispersion) outside I_0 . So long as it remains inside the interval I_V , a retrieve operation (measurement) will yield a result in I_V , but once components outside I_V develop, the probability of a read error becomes non-zero and then grows rapidly. Let Δt be the maximum time interval after t_0 during which a measurement is still safe. The analogue memory must be refreshed by performing a new measurement before or an erroneous result may be generated. Note that any real measurement on an interval of length x_V will take at least $2x_v/c$, where c is the speed of light in a vacuum, since a light beam must travel to and from A to perform the measurement. It follows that if $\Delta t < 2x_v/c$, then the memory will not be feasible.

Since momentum and position are conjugate observables, their x -dimensional operators \mathbf{X} and \mathbf{P}_x obey the commutation relation

$$[\mathbf{X}, \mathbf{P}_x] = i\hbar \quad (6.55)$$

where \hbar is Planck's constant divided by 2π . From this, it follows that the uncertainties (root mean square deviations over a quantum ensemble of identical systems) in the initial (time t_0) values of x and the p_x satisfy the so-called 'Uncertainty Relation'

$$\Delta p_x \cdot \Delta x_0 \geq \frac{\hbar}{2} \quad (6.56)$$

Since the mass of A written m_A , after the 'store' measurement at t_0 , A is moving with an expected velocity of $\Delta p_x/m_A$ away from the prepared position. Naively, therefore, we might expect that at time Δt , A will have travelled a distance of $(\Delta p_x \cdot \Delta t)/m_A$ away from its prepared position. This, however, is a quasi-classical argument and it underestimates the problem, as we might suspect from noting that the Uncertainty Relation is an inequality and $\hbar/2$ a lower bound. The actual positional uncertainty, Δx , after the critical time Δt (at which the spreading wave packet exceeds its safe bounds), can be obtained via an application of Ehrenfest's Theorem from which it can be concluded (Cohen-Tannoudji *et al.*, 1977, p. 342) that

$$\Delta x^2 = \left(\frac{\Delta p_x \cdot \Delta t}{m_A} \right)^2 + (\Delta x_0)^2 \quad (6.57)$$

In order to determine the theoretical limitations of this system, we can now place an upper bound on Δt . Since

- $\Delta x = K \Delta x_0$
- $2 \times 10^R \Delta x = L$, and
- $\Delta p_x \geq \frac{\hbar}{2\Delta x_0}$

it follows that the maximum value Δt after which the memory becomes unsafe is given by

$$\Delta t \leq \frac{\sqrt{K^2 - 1}}{K^2} \times \frac{L^2}{4 \times 10^{2R}} \times \frac{2m_A}{\hbar} \quad (6.58)$$

It is easy to verify that

$$\max\left(\frac{\sqrt{K^2-1}}{K^2}\right) = \frac{1}{2} \quad (6.59)$$

Approximating $\hbar \simeq 10^{-34}$ gives

$$\Delta t \leq 0.25 \times 10^{34-2R} \times L^2 m_A \quad (6.60)$$

For viability, $\Delta t \geq 2x_V/c$ so, since $c \approx 3 \times 10^8$ (m s⁻¹),

$$m_A > \frac{8 \times 10^{2R-42}}{3L} \times \frac{x_V}{L} \quad (6.61)$$

This depends on the value being stored. However, the memory must work for all values of x_V , so we can set $x_V = L$. We now have a final result:

$$m_A \times L > 2.6 \times 10^{2R-42} \quad (6.62)$$

or, alternatively,

$$m_A \times \Delta x > 1.3 \times 10^{R-42} \quad (6.63)$$

The limitation is applicable to any scheme that relies on the physical position of an object to store a real number. To increase the range of a positional analogue representation by 1 bit of precision requires the mass \times length product to increase by approximately a factor of four. It is very easy to see that a system of this type cannot scale unless allowed to grow arbitrarily in physical size, with consequent implications not only for the feasibility of construction but also for the speed at which stored data might be retrieved.

In contrast to a digital machine, where the resources used to perform higher-accuracy computation grow linearly with the number of bits of accuracy, the resources needed by the analogue machine grow exponentially with the accuracy of the calculation.

The following table gives approximate values for the parameters required for one-dimensional analogue positional memories at several values of precision (in bits), where possible, for $L = 10$ m. The 32-bit example requires masses of the order of a few atoms of uranium placed to an accuracy of 2.5 nm, technically not an outrageous scenario:

| Precision ($R \log_2 10$) | Length | Approx. mass | $2\Delta x$ |
|-----------------------------|--------|------------------------|-------------------------|
| 32 | 10 m | 5×10^{-24} kg | 2.3×10^{-9} m |
| 64 | 10 m | 10^{-4} kg | 5.4×10^{-19} m |
| 128 | 5 km | 6×10^{31} kg | 1.4×10^{-35} m |

However, above 32-bit precision, the systems become increasingly implausible, even with exotic technologies. For the 128-bit example,

L is chosen to ensure that Δx exceeds the Planck Length, L_p ($\sim 1.6 \times 10^{-35}$ m), which is the minimum precision possible in any physical measurement of position. However, even accommodating this fundamental constraint, m_A is of the order of 50 solar masses and must occupy a space at least equal to its Schwarzschild radius, S_A . As this is given by

$$S_A = m_A \times 1.48 \times 10^{-27} \quad (6.64)$$

A would be at least several kilometres across, even if it was a black hole. When dealing with objects with sufficient gravitational fields, the Uncertainty Principle has to be modified to take account of gravitational forces and, while the detail is still not fully settled, in most current theories of quantum gravity it is believed (Scardigli, 1999) to take a form such as

$$\Delta x \geq \frac{\hbar}{2\Delta p} \times kL_p^2 \frac{\Delta p}{\hbar} \quad (6.65)$$

where k is a constant. This inequality (the so-called Generalized Uncertainty Principle) is interesting because it predicts that when dealing with massive objects, the Δx associated with a given Δp may be significantly higher than in the non-gravitational case. The GUP also confirms that the uncertainty in position can never be less than L_p regardless of how imprecise the momentum knowledge is: this is in agreement with the claim that L_p is the smallest measurable quantum of length. We conclude as follows:

1. Any physically buildable analogue memory can only approximate the reals and there are very definite limits as to the accuracy achievable.
2. Analogue storage of reals will, for high precision work, always be outperformed in terms of device economy by digital storage.
3. Physically buildable analogue computers cannot rely upon the availability of exact stored real numbers to outperform digital computers.

6.9 Error rates in classical and quantum gates

We will discuss quantum computing in greater depth in Chapter 6, but this is an appropriate place to look at the power-consumption limits of quantum computing.

Both classical and quantum computer gates are subject to stochastic errors. In the classical case, these are thermodynamic in character; in the quantum gates, they are due to vacuum or zero-point noise.

Banacloche (see Gea-Banacloche, 2002; Gea-Banacloche and Kish, 2003) shows that for a quantum logic gate, the error rate ϵ_q is given by

$$\epsilon_q > \frac{fh}{E} \quad (6.66)$$

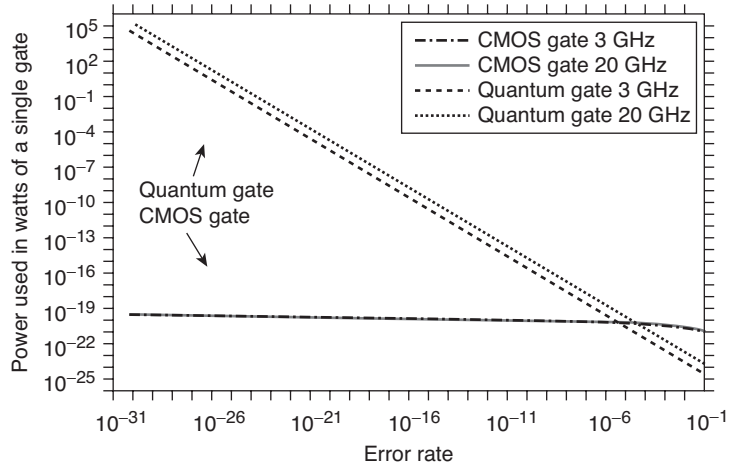


Fig. 6.10 The comparative power used in CMOS and quantum gates as a function of the error rate.

where f is the frequency of operation, h is Planck's constant, and E is the energy used to operate the gate.

For a classical gate, the error rate is given by

$$\epsilon_c > \frac{2}{\sqrt{3}} e^{\frac{-E}{kT}} \quad (6.67)$$

where T is the gate temperature and E is the minimum energy dissipated in the classical circuit in the course of the operation. The exponential increase in the error rate with rising temperature is why cooling is so important.

If we consider clock frequencies in the GHz range, we can look at the minimal power dissipated by CMOS and quantum gates as a function of their error rates. These are shown in Fig. 6.10, which is derived from Kish (2004). At any error rate below 10^{-6} the CMOS system has a lower energy requirement. Current CMOS technology can achieve error rates of the order of 10^{-25} . To achieve this error rate, the quantum gate would require around 100 joules for each switching operation. A single quantum gate operating in the GHz range would be using of the order of 100 MW of power (Kish, 2004).

Whilst quantum computing does hold promise as a means of reducing the complexity of algorithms, it seems unlikely that it will allow us to escape from the power-consumption limits posed by classical computing.

Beyond the logical limits of computing?

7

7.1 Introduction

In Chapter 4, we quickly surveyed how the formal foundations of mathematics undermine themselves through the complementary mechanisms of self-reference and substitution. Now, self-reference, and the paradoxes it engenders, have the feel of a dodgy sleight of hand. We humans are able to formulate and identify paradoxes, so perhaps TMs—and maybe even meta-mathematics itself—are just not powerful enough to capture our meta-reasoning abilities. Indeed, Nagel and Newman (1959) argued precisely that Gödel’s paradoxes show the superiority of human reasoning over formalism and mechanism.

TMs have a number of obvious restrictions. First, they only have one tape. Perhaps allowing TMs to manipulate *multiple tapes* would expand their computational powers. After all, human brains are made up of multidimensional networks of cells.

However, for any multi-tape TM it is straightforward to build an equivalent single-tape machine that simulates it. One approach is to interleave alternate cells of the tapes, as shown, for example, in Fig. 7.1.

Conceptually, a multi-tape TM is no different to a RAM, where each byte consists of eight bits that are manipulated in parallel.

Perhaps a multi-tape TM is actually a poor simulacrum for a human brain where several brain cells may be accessed at the same time. A better analogue might be a TM with *multiple tape heads* all manipulating the same tape.

Once again, for any multi-head TM, it is straightforward to construct an equivalent single-head machine that simulates it. Given that a multi-head TM must necessarily access different tape cells at each stage in a computation, then the equivalent single-head TM may be arranged to perform the operations for each head in turn. One way is to use cells on the single tape to record the positions of the multiple head, just as the UTM used tape cells to record information about the TM being simulated.

Conceptually, a multi-head TM is the same as a multi-core CPU, where each core shares the same memory, but only one core can access a single memory location at a time.

| | | |
|-----|---------------------------------------|-----|
| 7.1 | Introduction | 173 |
| 7.2 | Oracles, complexity, and tractability | 174 |
| 7.3 | Beyond the Turing Machine? | 176 |
| 7.4 | Numberology | 177 |
| 7.5 | What is real about the reals? | 180 |
| 7.6 | Real measurement | 181 |
| 7.7 | Back to Turing | 184 |
| 7.8 | Reservations about Cantor | 185 |

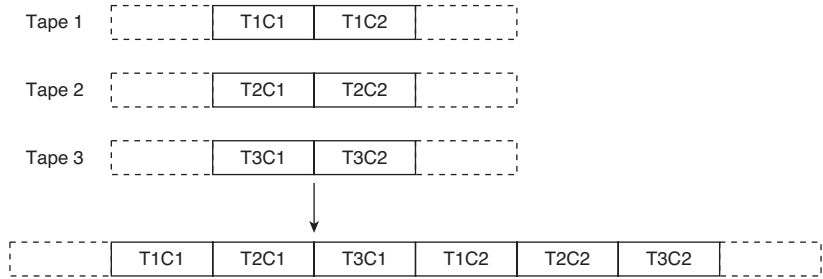


Fig. 7.1 Multi-tape and single tape.

Finally, human brains are *non-deterministic* in that cells fire simultaneously and in what appear to be unpredictable orders. In contrast, a TM is *deterministic* in executing instructions in a highly predictable manner depending on the current state and tape symbol. Thus, we might make TMs non-deterministic by relaxing the requirement that there is at most one instruction for each possible state/symbol combination and allowing multiple instructions for the same state and symbol. Then, at each stage in a computation, an arbitrary matching instruction may be chosen.

And once again, it turns out that every non-deterministic TM has a deterministic equivalent. Note that a successful run of a non-deterministic TM—that is, one that halts having found a solution to a problem—involves the execution of a finite sequence of specific instructions. As a TM is controlled by a program with a finite number of instructions, it is possible to enumerate all the different finite paths through them, in order of path length, starting with single instruction paths. Each path is then executed in turn. If path execution fails, as it contains an illegal state/symbol requirement, or ends without having solved the problem, then the next path is tried. Thus, if the original non-deterministic TM halts having solved the problem, then so must one of the explored deterministic TM paths.

Non-deterministic TMs have an equivalent in parallel computers; for example, multi-core CPUs or multi-processor clusters, where alternative instruction sequences may be applied to the same problem at the same time.

7.2 Oracles, complexity, and tractability

While Turing explicitly thought that the notion of effective procedure was fully captured by TMs, he also briefly explored what it would mean to be more powerful than a TM. Thus, in Turing (1939), he discusses the possibility of augmenting a TM with an *oracle* that is somehow able to solve arbitrary number-theoretic problems, noting that:

We shall not go any further into the nature of this oracle apart from noting that it cannot be a machine.

(Davis, 1965, pp. 166–7).

Unsurprisingly, Turing then deftly shows, by self-application contradiction, that the termination of TMs with oracles is itself undecidable.

While oracles have no physical reality, they remain important in distinguishing between different *complexity classes*; that is, broad groups of problems which are solvable but that take significantly different amounts of time to find solutions. While it is often possible to make very detailed analytic models of the behaviours of algorithms, their time complexities are usually characterized as a bound rather than in precise form, using ‘big O’ notation (Knuth, 1976). The idea here is to capture how the time taken for an algorithm to process some data changes with the size of the data.

For example, an ‘algorithm’ to turn a bank of lights on or off from a central switch takes the same amount of time regardless of the number of lights. This is characterized as $O(1)$, where the 1 indicates that the time is constant. In contrast, walking down a corridor turning on each of an equally separated sequence of N lights in turn takes a *constant amount of additional* time with each additional light. This is characterized as $O(N)$, where the N shows that the time is linear in the data size.

For example, comparing two unordered lists of names, of lengths M and N respectively, to see which entries are common to both takes time proportional to the product of the list lengths: $O(MN)$. For example, finding all the pairs of names from a list of length N takes time proportional to $N * N$: $O(N^2)$. These are both examples of the more general class of *polynomial* time algorithms.

In contrast, the time to construct a truth table for a propositional formula doubles with each new variable. One variable has two possible truth values; two variables have four possible combinations of truth values; three variables have eight possible truth values; and so on. In general, N variables have 2^N possible truth values, so the complexity is designated as $O(2^N)$. Such complexity is termed *exponential*.

Figure 7.2 shows how $O(N^2)$ and $O(2^N)$ complexities grow with N . Note that $O(2^N)$ rapidly outgrows $O(N^2)$ for even small values of N . In general, a time proportional to some sum of powers of the amount of data is termed *polynomial* (P) and a P problem is said to be *tractable*. And, in general, any complexity greater than polynomial is termed *non-polynomial* and a non-polynomial problem is said to be *intractable*.

There is a class of problems termed *non-deterministic polynomial* (NP) that take non-polynomial time to solve, but whose alleged solution algorithms could be checked against an answer in polynomial time by a deterministic TM given an oracle to provide that answer.

| Complexity | $N=1$ | $N=2$ | $N=4$ | $N=6$ | $N=8$ | $N=10$ | $N=12$ | $N=14$ | $N=16$ |
|------------|-------|-------|-------|-------|-------|--------|--------|--------|--------|
| $O(N^2)$ | 1 | 4 | 16 | 36 | 64 | 100 | 144 | 196 | 256 |
| $O(2^N)$ | 2 | 4 | 16 | 64 | 256 | 1024 | 4096 | 16394 | 65576 |

Fig. 7.2 Growth rates for $O(N^2)$ and $O(2^N)$.

7.3 Beyond the Turing Machine?

As discussed in Chapter 4, all models of computation since TMs and the λ calculus have been shown to be equivalent in computational expressivity. Nonetheless, there have been numerous attempts to propose models that in some sense transcend the limits of the Church–Turing thesis, termed variously *super-Turing* or *hypercomputational*.

In Cockshott and Michaelson (2007), we list what we see as requirements for a new system to demonstrably go beyond Turing Machines. Before we rehearse these, we will recapitulate fundamental notions of decidability.

A TM is said to encode the corresponding decision procedure. If that TM will terminate with a solution after a finite number of execution steps, then the problem is said to be decidable. If it is not possible to construct such a TM, then the problem is said to be undecidable. If a TM can be constructed but it cannot be determined by computable means whether or not it will terminate, then the problem is said to be semi-decidable. The halting problem has become a *canonical* exemplar of undecidability: if any other problem is reducible to an instance of the halting problem, then it must be undecidable.

In general, a demonstration that a new system is more powerful than a Church–Turing system involves showing that while all terms of some Church–Turing system can be reduced to terms of the new system, there are terms of the new system that cannot be reduced to terms of that Church–Turing system; in other words, the new system has greater expressive power than that Church–Turing system and hence of any Church–Turing system. Incidentally, it would be truly astonishing if a new system were demonstrated whose terms could not be reduced to those of a Church–Turing system; that is, the new system and Church–Turing systems had incomparable expressive power.

More concretely, we think that requirements for a new system to conclusively transcend the logical limits of Church–Turing are, in increasing order of strength:

1. A demonstration that some problem known to be semi-decidable in a Church–Turing system is decidable in the new system. That is, there is some problem that has undecidable termination when expressed as a TM, and has decidable termination when expressed in the new system.
2. A demonstration that some problem known to be undecidable in a Church–Turing system is semi-decidable in the new system. That is, there is some problem that cannot be expressed as a TM, but that can be expressed in the new system, albeit with undecidable termination.
3. A demonstration that some problem known to be undecidable in a Church–Turing system is decidable in the new system. That is, some problem that cannot be expressed as a TM can be expressed as a decidable terminating construct in the new system.

4. Characterizations of classes of problems corresponding to (1)–(3). That is, it is possible to identify the essential properties of the sorts of problems that are semi-decidable or decidable in the new system, but not as TMs, thus giving a handle on which class some new problem corresponds to.
5. Canonical exemplars for classes of problems corresponding to (1)–(3). That is, there are instances of each of these classes of problem to which other members of the same class are reducible, as with the halting problem for TMs.

7.4 Numberology

In Chapter 8, we will explore in some detail various attempts to go beyond the limitations of systems that satisfy the Church–Turing thesis. Many such attempts are based on arguments that the decidability limitations of TMs are situated in their discrete and finitistic foundations, and seek to elaborate novel machine constructions using properties of infinity or the real numbers.

In this and the final section of this chapter, we will survey conceptual difficulties with characterizations of infinity and real numbers. We will first ground that discussion in a survey of epistemological attitudes to the very notion of number.

7.4.1 Realism

There is a fundamental dichotomy between conceiving of numbers as actually existing real entities as opposed to abstract mathematical constructs. For *Platonic realists*, numbers have real existences as ideal entities that are denoted by symbolic representations (Wilder, 1978). The term ‘realist’ is confusing, as Platonists are philosophical *idealists* who think that material reality is an illusion, at best a pale reflection of some underlying universe of ideal forms.

Platonism is now a minority, if not uncommon, taste in Mathematics, but is still prevalent in the Oxford school of Computer Science. Their highly influential *denotational semantics* is based on the idea that the meaning of computer programs is to be found in the idealized functions from inputs to outputs that they denote (Michaelson, 1993).

7.4.2 Aristotle and infinity

Plato’s student Aristotle (2000) was much exercised by infinity and by arguments from infinite regress, questioning whether infinity was in any sense real:

... nothing infinite can exist; and if it could, at least the notion of infinity is not infinite ...

(Book II, part 2).

Unlike his teacher, who accepted the reality of some ideal infinity, Aristotle drew an explicit distinction between *potential* and *actualized* infinities:

But also the infinite and the void and all similar things are said to exist potentially and actually in a different sense from that which applies to many other things, e.g. to that which sees or walks or is seen. For of the latter class these predicates can at some time be also truly asserted without qualification; for the seen is so called sometimes because it is being seen, sometimes because it is capable of being seen. But the infinite does not exist potentially in the sense that it will ever actually have separate existence; it exists potentially only for knowledge.

(Book IX, part 6).

An actualized infinity requires the entire infinity of entities to be physically realized. That is, the infinity must be a *completed totality*. One example is found in Post's machine (Post, 1936), developed contemporaneously with the TM, which has an infinite tape as an initial condition.

In contrast, a potential infinity is one which is never actually fully achieved. That is, a potential infinity may be approached by continuously adding elements to some collection, but at each stage the collection remains finite in size. Thus, a TM tape may have cells added to it indefinitely by a non-terminating computation, but at each state transition is of finite length.

As we shall see below, Cantor's diagonalization argument depends on an actualized infinity of reals.

7.4.3 Logicism, formalism, and intuitionism

Nearly 2000 years after Plato and Aristotle, Kleene (1952) identified three main schools of thought on the foundations of mathematics: logicism, formalism, and intuitionism current from the late nineteenth to the mid-twentieth centuries.

The *logistic* school, typified by Russell, sought to reduce classical mathematics, including by implication actualized infinities, to logic. As Kleene notes:

Logicism treats the existence of the natural number series as an hypothesis about the actual world ('axiom of infinity').

(p. 46).

The *formalist* school, typified by Hilbert, sought, as we have discussed, to formulate classical mathematics as a consistent axiomatic theory. Originally, it was thought that the consistency of axiomatized mathematics was to be found in a model from some other consistent theory: this smacks of tautology or teleology. Instead, Hilbert proposed that a theory for classical mathematics should be proved to be free of internal contradictions using the meta-mathematical techniques discussed in Chapter 4. Thus, for the formalists, the epistemological status of number was not important.

The *intuitionist* school, typified by Brouwer, entirely rejected arguments that required the assumption of actualized infinities. That is, the only mathematical objects of certain epistemological status were finite, with properties that could be exhaustively examined. Note that this does not exclude potential infinities.

Thus, intuitionism excludes the *law of the excluded middle* for existence proofs over infinite sets. This axiom is usually expressed as $A \vee \neg A$; that is, something is either true or not true. This is the basis of *proof by contradiction*, where one assumes something and derives its negation. However, an existence proof by contradiction of, say, a number having some property, by assuming the non-existence of a number with that property, implicitly assumes that the infinite set of numbers exists (Kleene, 1952).

Philosophically, intuitionism is not unproblematic. While it rejects trying to ground mathematics in science or logic, it claims that mathematics corresponds to ‘the exact part of our thinking’ (Kleene, 1952, p. 51, quoting Heyting, 1974). That is, intuitionism sees human intuition as the ultimate arbiter of mathematical truth.

Nonetheless, there is much of value in intuitionism in trying to get to grips with claims for hypercomputation based on appeals to infinity. In particular, intuitionists favour *constructive* methods such as induction for making mathematical objects, by augmenting base values a finite number of times. This is actually less restrictive than might first appear. As Kleene notes:

Most non-constructive existence proofs can be replaced by constructive ones.
(p. 52).

Contemporary variants of intuitionism include *constructivism*, and, ultimately, *finitism*, which only accepts mathematical entities that can be built in a finite number of rule applications from base entities (Tiles, 1989).

To conclude this brief survey of the philosophies of mathematics, according to Cantor (Barrow, 2005), there were three sorts of infinity: *absolute* infinity in the mind of God, *mathematical* infinity in the minds of people, and *physical* infinity in the material universe. Barrow (2005) provides a wry truth table, after Rucker (2004), of the attitudes of different philosophers to infinity—see Table 7.1.

Table 7.1 Philosophers and infinities (Barrow, 2005, p. 94).

| Philosopher | Mathematical | Physical | Absolute |
|------------------|--------------|----------|----------|
| Abraham Robinson | No | No | No |
| Thomas Aquinas | No | No | Yes |
| Plato | No | Yes | No |
| Luitzen Brouwer | No | Yes | Yes |
| David Hilbert | Yes | No | No |
| Kurt Gödel | Yes | No | Yes |
| Bertrand Russell | Yes | Yes | No |
| Georg Cantor | Yes | Yes | Yes |

7.5 What is real about the reals?

7.5.1 Origins

The concept of real numbers arose in Greek mathematics as a consequence of the theorem of Pythagoras. The Pythagoreans had a conception of the universe based on integers:

Contemporaneously with these philosophers and before them, the so-called Pythagoreans, who were the first to take up mathematics, not only advanced this study, but also having been brought up in it they thought its principles were the principles of all things. Since of these principles numbers are by nature the first, and in numbers they seemed to see many resemblances to the things that exist and come into being—more than in fire and earth and water (such and such a modification of numbers being justice, another being soul and reason, another being opportunity—and similarly almost all other things being numerically expressible); since, again, they saw that the modifications and the ratios of the musical scales were expressible in numbers;—since, then, all other things seemed in their whole nature to be modelled on numbers, and numbers seemed to be the first things in the whole of nature, they supposed the elements of numbers to be the elements of all things, and the whole heaven to be a musical scale and a number.

(Aristotle, 1960, p. 9).

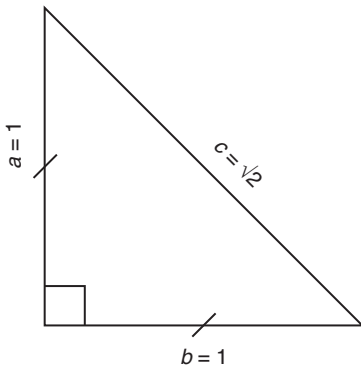


Fig. 7.3 A right-angled triangle with two unit sides has third side $\sqrt{2}$.

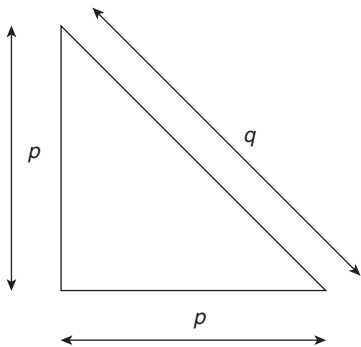


Fig. 7.4 A scaled version of the unit right angle triangle with sides, p, q .

A problem with this conception arose with the theorem of Pythagoras about the sum of the squares on the sides of a right-angled triangle. Suppose that we have the triangle shown in Fig. 7.3 with two sides of unit length; then the third side will have a length that, when squared, will have an area of 2 square units. Its length will then be a number that, when squared, will be 2.

Suppose that you set out such a triangle with sides of 1 m: the diagonal will be measured, using a ruler graduated in millimetres, to be a bit over 1414 mm. The question naturally arises as to what is the exact measurement of the length of the hypotenuse?

If we choose some sufficiently small unit of measurement for each side, will we get an exact measure for the hypotenuse?

For that to be the case, the length of the hypotenuse and the other side have to stand in a definite ratio q/p , with q, p being whole numbers as in Fig. 7.4.

There are many proofs that this can't in fact be the case. Here is one example.

1. Let $\sqrt{2} = q/p$: assume that p and q are mutually prime (numbers with no common factors), since we want to express the factors in the simplest way.
2. Their squares must still be mutually prime, for they are built from the same factors.
3. Therefore, the fraction q^2/p^2 cannot cancel out. In particular, q^2/p^2 cannot cancel down to equal 2.
4. Therefore, $q^2/p^2 \neq 2$, so there can be no two integers p, q such that $\sqrt{2} = q/p$.

This appears to have, as a consequence, the infinite divisibility of space. Suppose that we start with a right isosceles triangle with two sides measuring 1 m and the third side $\sqrt{2}$. We then divide the equal sides into p subdivisions; then there will be an integers $r > p$ such that the length of the hypotenuse, in these subdivisions, lies in the range $r..r + 1$.

Using our earlier example, divide into $p = 1000$ mm, then $r = 1414$ mm, and the hypotenuse is between 1414 mm and 1415 mm.

Now replace p with r , and recurse. The implication is that space will be infinitely divisible. This is the basic intuition or metaphor that we have for real numbers: the idea of a real number line that is infinitely divisible, which itself is a metaphorical extension of the idea that a line is itself arbitrarily divisible into smaller units.

7.6 Real measurement

Measurement using finer and finer scales requires that we have a finer and finer ruler to do the measurement. For practical purposes, when we wish to establish the length of the standard metre we do it by means of optical interferometry, using light emitted by a particular type of atom or molecule. The International Standards Organization specifies certain transitions of I_2 as having specified wavelengths in metres (Quinn, 1999). By counting the number of wavelengths in a given distance, one can then arrive at a measure of the distance.

Measurement of length thus requires the use of photons that allow us to measure to an integer accuracy in terms of wavelengths.

We can envisage an experimental set-up that allows us to test Pythagoras' theorem in the real world. Look at Fig. 7.5. Suppose that we have previously measured AB and AC by means of interferometry to be the same integral number of wavelengths long. Clearly, if it is the case that the optical path round the entire system is also an integral number of wavelengths, then we would either have shown that $\sqrt{2}$ is a

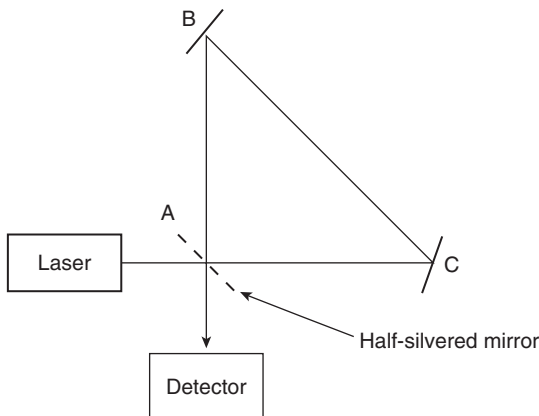


Fig. 7.5 A possible experimental set-up for testing Pythagoras' theorem in the physical world.

rational number or that Pythagoras' theorem did not hold in the real world.

But there appear to be big practical problems with performing such a test. We can't, with the same arrangement of mirrors, simultaneously and independently measure all the lengths. In order to measure the length AC , for example, we would have to rotate the mirror at C so that it faces directly towards A , and likewise for measuring the distance AB . But this would perturb the situation with respect to our final measurement. How are we to be sure that our rotation of the mirrors has not perturbed the position of point C so that after rotation of the mirrors, the distance AC is no longer an integral number of wavelengths?

Further, even if this could be done, how do you ascertain that the triangle is exactly a right-angle one?

If we assume initially that space does have a geometry such that Pythagoras' theorem holds, then we could initially set up an arrangement such that the distance $AC = \frac{3}{4}AB$ and the distance $BC = \frac{5}{4}AB$, giving a 3,4,5 right-angled triangle. We could then extend AC until $AC = AB$, returning to the situation shown in the figure. But there is a logical problem with this. The means of setting up the right angle assume the validity of the Pythagorean theorem in the real world, which is what we are trying to test in seeing how close BC is to $\sqrt{2}AB$. But if we can't initially assume that the theorem holds physically, how can we be sure that we have set up a right angle?

By very careful work, we might overcome all the practical if not the logical problems, but we would then face another. There would be an uncertainty in our measurement of any of the distances, an uncertainty that is proportional to the wavelength of the light used.

The shorter the wavelength of the light we use, the more accurate our measurement, since our unit of measure gets shorter and shorter. Hence to measure the hypotenuse of a 1 m right-angled triangle more and more accurately, we need shorter and shorter wavelengths of light.

Could we measure to an arbitrary degree of accuracy in this way, or is there a fundamental limit?

It appears that there is. The predictions of quantum mechanics and of relativity theory conspire to indicate that there is a smallest scale at which measurement can be done.

7.6.1 The energy of a photon

The energy and wavelength of a photon are related by

$$E = hc/\lambda \quad (7.1)$$

where E is the energy of the photon, h is Planck's constant, λ its wavelength, and c the speed of light.

Using $E = mc^2$, we can express this energy as an equivalent gravitational mass:

$$m = h/\lambda c \quad (7.2)$$

It is clear that as we reduce the wavelength of the light that we are using, the gravitational mass of the photons increases. For practical measurements at the scales we can reach today, this does not matter, but if keep pushing the wavelengths down, we reach a point at which, according to relativity theory, they will start to significantly distort space with their own gravitational mass. As this happens, definitions of distance start to break down.

The radius of a black hole The ultimate distortion of space occurs when mass is sufficient to create a black hole. The radius of a black hole is given by

$$r_s = 2Gm/c^2 \quad (7.3)$$

where r_s is the Schwarzschild radius, G is the gravitational constant, m is the mass of the gravitating object, and c is the speed of light in a vacuum.

Note that the radius is proportional to the mass of the hole—not, as you might expect, to the cube root of the mass. If we look at Fig. 7.6, we see the result of plotting Eqs 7.2 against 7.3. The two lines intersect at what is known as the Planck length, 1.6163×10^{-35} m.

The infinitely divisible space of Euclidean geometry is thus not ‘real’. The theorem of Pythagoras applies in the model of space assumed by Euclidean geometry, but is not well defined at very small scales around the Planck length. In particular, Euclid’s assumption of points with position but no magnitude is not well defined in physical reality. The geometrical metaphor we are taught of the ‘real’ number line must be mistaken.

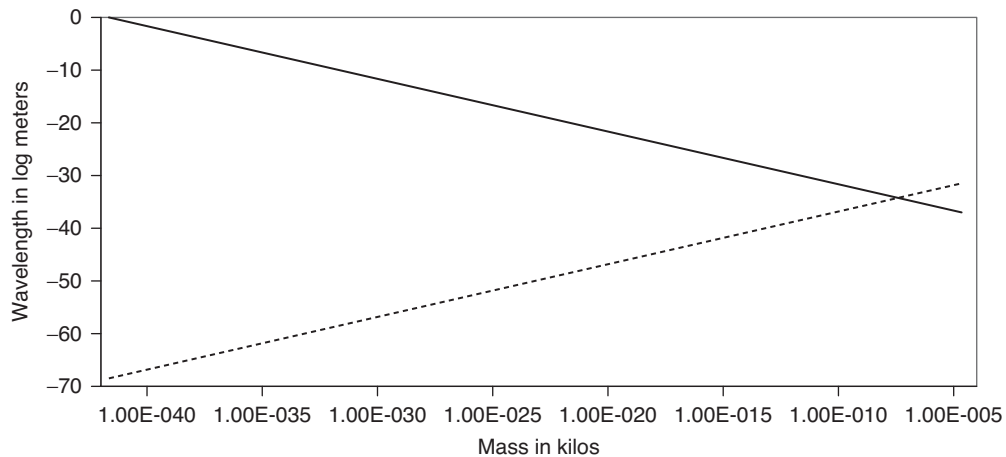


Fig. 7.6 A plot of the radius of a black hole of a given mass against the mass of a photon of a given wavelength. The vertical axis measures the logarithm of the radius or the wavelength in metres, and the horizontal axis measures mass in kilograms. The solid line is the wavelength of a photon of a given equivalent gravitational mass, and the dotted line is the black hole radius. The length and mass at which they intersect are the Planck distance and the Planck mass.

7.7 Back to Turing

Turing (1937) defined a computable real as a number whose decimal expansion can be computed to any degree of precision by a finite algorithm. Clearly we can change this definition to one in which we say that its binary expansion can be computed to any degree of accuracy required without losing Turing’s meaning. If R is a computable real, there is a function $R(n)$ which when given an integer n will return the n th digit of the number.

Any computer program can be considered as a binary integer, made up of the sequence of bytes of its machine code. Most such integers are not valid programs for computing real numbers. Most will produce no output, or cause the computer to crash.

In mathematics, we assume that

$$\text{Reals} \supset \text{Rationals} \supset \text{Integers}$$

On the contrary, from a computer science perspective,

$$\text{Integers} \supset \text{valid Programs} \supset \text{computable reals}$$

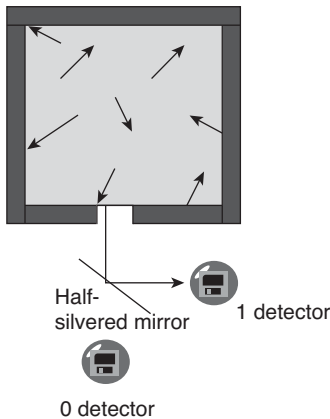


Fig. 7.7 Using the thermal energy of a black body radiation cavity as a random bit source.

A computable real has an encoding that is shorter than its output. For any program, we can measure its length in bits. Suppose that $\text{Length}(R(n)) = k$ bits; then by setting $n > k$ we can compute more bits of the real than the program itself contains.

Chaitin (1999) defines a random bit sequence of length n as one for which no program of length $< n$ exists that will print it out. Hence any non-computable real is random in Chaitin’s sense.

How many random reals really exist? Arguably, there are none.

A cavity at above background temperature can act as a random bit source, as shown in Fig. 7.7. But in providing bits it cools down and provides only a finite number of bits. As it cools, entropy is removed to supply the bits. Thus any finite thermal system can only release a finite amount of information and thus only a finite leading bit sequence of a random real—a random integer, not a random real.

Note that what we have is the release of information as the subsystem moves to thermal equilibrium. In principle, this argument can be extended to the whole universe. The amount of the universe that is within our event horizon is finite. That is, the volume within which light setting off since the Big Bang can have reached us, and thus can have a material influence on us, is finite. From this, it follows that there can only be a finite—a vast, but finite—number of bits in the universe. Lloyd (2002) carries this reasoning further to estimate the total computational capacity of the entire universe. On the basis of some, admittedly uncertain, assumptions about the resources available, he estimates the universe to have entropy of about 10^{90} bits. Other estimates (Sazonov, 1995; Kornai, 2003) suggest upper bounds of 2^{512} or 2^{1000} bits, the latter figures being deliberately cautious.

It is clear, however, that not all this information is available to perform arbitrary calculations. To perform a calculation, one has to prepare

one's computing machine in a known state. Classically, this is a Turing Machine with a specified programme at the start of the tape and the rest of the tape set to zero. But if we consider the universe as a whole, such a clearing operation is inconsistent with the laws of thermodynamics. We have established in previous chapters that computation involves the dissipation of heat, and that in consequence feasible computing configurations have a power that is proportional to the surface area through which they can get rid of waste heat—area, rather than volume, is the crucial consideration. This emphasis on area rather than volume is reinforced by the holographic principle (Beckenstein, 1981; Bousso, 2002) arising from quantum gravity, which shows that the information content of a volume of space is a function of its surface area.

Whatever bound we set, there is a limit to the size of any integer that could be represented in the largest physically feasible computing machine. By extension, there is a limit to the precision with which any real number can exist. For non-random reals, there exists a more compact representation of the number in the form of their producing algorithm, but even if we have a compact algorithm for a number, we do not actually know much about it unless we have its complete decimal or binary expansion. Kornai (2003) points out that although Ackerman's function gives us a compact notation for certain very large numbers, we would be powerless to even specify the first few digits of $A(4,4)$ with the computational resources of the entire universe.

7.8 Reservations about Cantor

In the light of this consideration we have, as computer scientists, to be sceptical about Cantor's results. The diagonal argument rests on the assumption that all of the reals can be listed as completed infinities. Suppose instead that we approach the Cantor argument from the view of feasible computation.

Apply the Cantor argument to IEEE 32-bit floating point representations of the reals. In these, the number is represented by a 23-bit binary fraction or mantissa multiplied by 2^e , where e is the exponent part of the number. We can list all the 2^{23} possible mantissas in order:

| Binary | line number |
|------------------------------|-------------|
| .000000000000000000000000 | 1 |
| .0000000000000000000000001 | 2 |
| .0000000000000000000000010 | 3 |
| ... etc | |
| .000000000000000000000010001 | 17 |
| .000000000000000000000010010 | 18 |
| .000000000000000000000010011 | 19 |
| .000000000000000000000010100 | 20 |
| .000000000000000000000010101 | 21 |
| .000000000000000000000010110 | 22 |
| ... etc | |

We then apply the diagonal procedure, inverting bits on the diagonal:

| Binary | line number |
|---------------------------|-------------|
| .100000000000000000000000 | 1 |
| .010000000000000000000000 | 2 |
| .001000000000000000000000 | 3 |
| ... etc | |
| .000000000000000000000001 | 17 |
| .000000000000000000000000 | 18 |
| .000000000000000000000001 | 19 |
| .000000000000000000000010 | 10 |
| .000000000000000000000010 | 21 |
| .000000000000000000000010 | 22 |
| ... etc | |
| .111111111111111111111010 | 8388587 |
| .111111111111111111111011 | 8388588 |
| .111111111111111111111100 | 8388589 |
| ... etc | |
| .111111111111111111111011 | 8388604 |
| .111111111111111111111100 | 8388605 |
| .111111111111111111111101 | 8388606 |
| .111111111111111111111110 | 8388607 |
| .111111111111111111111111 | 8388608 |

and we obtain the number 1111111111111111101011 by going down the diagonal—but this number is already included in our list in position 8388588, so the diagonal procedure fails to produce a new number. Clearly, if we extend our precision to 24 bits the same situation will apply, but the number on the diagonal will now be found about twice as far down the list. Each time we increment the precision of the reals, we again find the diagonal binary number further down the list, but the distance down the list that we have to traverse to find it grows exponentially. The Cantor argument relies for its plausibility on already having a completed infinity of digits of all the binary fractions prior to putting them in order. But if one takes the Aristotelian principle that only potential infinities may sense—that is, computational processes that are potentially unbounded provided that sufficient computational resources are provided—then one can use a conventional argument from induction, based on our example with the IEEE reals, to show that however accurate our representation of the reals is (that is, with potentially unbounded TM tapes), the number generated by the diagonal algorithm is never new.

Hypercomputing proposals

8

There have been many attempts to articulate new systems for computability that are claimed to transcend the limitations of Church–Turing systems, this being termed, for example, *hypercomputing* or *super-Turing*. Copeland (2002) provides a thorough summary. Our view is that none of these proposals have yet made a convincing case for dropping the fundamental theoretical basis on which computer science has rested until now. In this final chapter, we will review some of the alternative models of computation that have been put forward recently. We will look at:

1. Proposals to use infinities in Turing Machines.
2. Proposals to use infinitely precise analogue computers.
3. Proposals to use black holes for computing.
4. Proposals to use new calculi.

| | | |
|------------|---|------------|
| 8.1 | Infinite Turing Machines | 187 |
| 8.2 | Infinitely precise analogue computers | 190 |
| 8.3 | Wegner and Eberbach’s super-Turing computers | 201 |
| 8.4 | Interaction Machines | 202 |
| 8.5 | π-Calculus | 206 |
| 8.6 | $\\$-Calculus | 211 |
| 8.7 | Conclusions | 214 |

8.1 Infinite Turing Machines

Universal Computers proposed by Turing are material apparatuses that operate by finite means. Turing assumes that the computable numbers are those that are computable by finite machines, and initially justifies this only by saying that the memory of a human computer is necessarily limited. By itself this is not entirely germane, since the human mathematician has paper as an *aide-mémoire* and the tape of the TM is explicitly introduced as an analogy with the squared paper of the mathematician.

Turing is careful to construct his machine descriptions in such a way as to ensure that the machine operates entirely by finite means and uses no techniques that are physically implausible. His basic proposition remained that: ‘computable numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means’.

Turing thus rules out any computation by infinite means. If infinite computation were to be allowed, then the limitations introduced by the TM would not apply.

Copeland (2002) proposes the idea of accelerating TMs whose operation rate increases exponentially, so that if the first operation were performed in a microsecond, the next would be done in $\frac{1}{2} \mu\text{s}$, the third

in $\frac{1}{4} \mu\text{s}$, and so on. The result would be that within a finite interval it would be able to perform an infinite number of steps.

The machine could, for example, compute π exactly:

Since a Turing Machine can be programmed to compute π , an accelerating Turing Machine can execute each act of writing that is called for by this program before two moments of operating time have elapsed. That is to say, for every n , the accelerating Turing Machine writes down the n th digit of the decimal representation of π within two moments of operating time.

(Copeland, 2002, p. 284).

Another supposed advantage is that they make the halting problem readily solvable. The accelerating machine simulates the specified TM for an infinite number of steps within a finite period and gives a definite answer as to whether or not it will halt.

This obviously evades Turing's stipulation that computations must be by finite means, and, in the process, evades all possibility of physical realization. A computing machine must transfer information between its component parts in order to perform an operation. If the time for each operation is repeatedly halved, then we soon reach the point at which signals travelling at the speed of light have insufficient time to propagate from one part to another within an operation step. Beyond this speed, the machine could not function.

In a hypothetical Newtonian universe without the constraint of a finite speed of light, this particular obstacle would be eliminated, but we would immediately face another. In order for the Newtonian machine to compute infinitely fast, its (now presumably mechanical) components would have to move infinitely fast and thus require infinite energy. Given that the machine would be dissipative (Landauer, 1961), the heat released would raise its temperature to an infinite extent, causing it to disintegrate.

Hamkins (2002) discusses what could be computed on Turing Machines if they were allowed to operate for an infinite time, but Turing ruled this out, with obvious good reason.

8.1.1 Black hole computing

Etesi and Némethi (2002) extend the idea of accelerating Turing Machines by proposing a gravitational mechanism by which the acceleration can be done. They suggest the use of a pair of computers with one (A) orbiting and a mathematician, or another computer, (B) falling towards the event horizon of a Kerr black hole. As the computer (B) approaches the event horizon, its time is slowed down relative to the passage of time for (A). The closer it gets to the event horizon, the slower its passage of time gets relative to that of (A). They propose that computer (A) be made to work through the infinitely many instances of some recursively enumerable set. An instance they cite is running through the infinitely many theorems of ZFC set theory to see if any are false. If among the infinite sets, one is found to be false, a light signal is sent to (B) indicating this. Because

of the slowdown of time for (B), things can be so arranged as to ensure that the light signal arrives before the event horizon is crossed.

The idea takes advantage of the slowdown in time produced by intense gravity fields. Suppose that, from a safe distance, we watch a blue LED torch fall into a black hole. As it gets closer and closer to the event horizon of the hole the light, from the torch will turn green, yellow, orange, red, and then become invisible infra-red. As it climbs out of the potential well caused by the black hole, the light loses energy and in consequence shifts towards the red, less energetic end, of the spectrum. Red light has a lower frequency than blue light, so an implication of this is that a time interval equivalent to one cycle of blue light, about 1.3×10^{-20} seconds, in the frame of reference of the torch has become the time for one cycle of red light, 3.1×10^{-20} seconds, in the frame of reference of the distant observer. From a distance, the torch's time, or that of anything else falling in, seems to slow down. If we reverse the perspective, an unlucky observer falling into a black hole will see the universe outside the intense gravity field of the black hole speed up. The closer you get to the event horizon, the more the time of the rest of the universe seems to accelerate. Eventually, just as you pass the event horizon, the rest of the universe's time becomes infinitely fast.

Esti and Nemeti show remarkable ingenuity in working out the details of this scheme. They have a plausible response to the most obvious objection: that the light signal from (A) would be blue shifted to such an extent that (B) could not detect it. They suggest that (A) computes the degree of blue shift that the signal would experience and selects a suitably long wavelength and modulation system to compensate. This is not, however, an adequate response. There remain serious objections.

Computer (B) is assumed, like any other TM, to operate with clock ticks. The clock cycle is the smallest time within which the machine can respond and carry out any action. There will, within (B)'s frame of reference, be a finite number of clock ticks before the event horizon is crossed. Consider the last clock tick before the horizon is crossed; in other words, the clock cycle that is in progress as the horizon is crossed. Prior to the start of this cycle, machine (A) will have only searched a finite number of the theorems of ZFC set theory. During the final clock cycle of (B), the entire infinite residual of the set of theorems are checked by (A). But any message sent from (A) whilst processing the infinite residual must occupy less than a clock cycle from (B)'s perspective. As such, it will be too brief to register at (B).

Any signal that (B) can respond to will correspond to (A) only having searched a finite part of the infinite set of theorems.

If we consider things from the standpoint of (A), what we are demanding is that it continues to operate reliably, searching through theorems, not just for millions of years, but for an *infinite* number of years. The assumptions of infinite reliability and an infinite energy source to keep (A) operating are clearly impossible.

For A to orbit the black hole for an infinite amount of time, the black hole would have to exist from now until infinity. If Hawking (1974) is right, black holes have very long but still finite existences.

The proposal requires a computer so reliable that it will continue to function not just for a few years, but for all eternity. No matter how reliable you make the computer, if it has any probability of failure and you sum that over an infinite period, then failure becomes a certainty. So the computer would have to be constantly backed up and maintained by an eternal civilization able to dedicate an infinity of time to this task. Who but the immortals might venture it?

Even the gods or fairies would face problems. Infinite sets of theorems involve individual theorems that are infinitely long. Zeus and Umbriel would have to gather the entire universe to feed the computer's memory. Long before any answer came out, the computer would collapse under its own gravitational mass to form another black hole, taking its secrets with it.

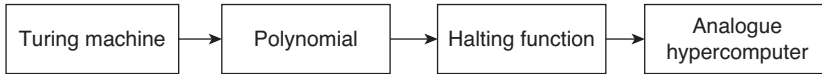
Umbriel can avert this fate by spinning his computer memories from ever lighter fairy stuff, to ensure that their density is so low that the computer always stays larger than the memory's Schwarzschild radius. But we are talking eternity here. As Umbriel's mega-computer toils through the eons, the stars will wink out, depriving it of power, and within a vast but still finite time, the black hole into which our intrepid mathematician has dived will evaporate. He, along with the rest of the black hole, will disperse across the universe as random thermal energy long before the hour of the infinite strike.

The black hole is essentially irrelevant to the argument. Nemeti's proposed infinite computations still take place in our mundane finite reality, in the world of death, decay and entropy. The black hole is the fairies' hall into which poor Tam Lynne is lured by their queen—a place in which time stands still—only to emerge to find his world in ruins.

8.2 Infinitely precise analogue computers

We have already discussed the limits that quantum measurement imposes on the possibility of constructing computing machines that would store information in an analogue mechanical fashion. Our example was given in terms of a mass whose position encoded a number. The idea of encoding numbers in mechanical positions is not absurd: machines described in Chapter 3 used this sort of encoding. But we demonstrated in Section 6.8 that there is an inherent limit to the accuracy of such encoding.

A general motivation for the construction of such machines is given by Da Costa in a paper ambitiously titled 'How to build a hypercomputer' (da Costa and Doria, 2009). The greater part of this account is given over to a mathematical justification of why the construction of a hypercomputer is feasible. Da Costa relies on the concept of an universal

**Fig. 8.1** Da Costa's proposal.

Diophantine polynomial. He cites Martin Davis (1973) as describing an algorithmic procedure out of which, given a Turing Machine with input a $M_m(a)$, we obtain a polynomial $p_m(a, x_1, \dots)$ so that it has roots if and only if the Turing Machine converges (outputs some result). From this, he then defines a real valued and real parametered halting function such that the function will be less than 1 if the Turing Machine halts and greater than one if it goes into an infinite loop. This function, it should be noted, initially requires an integral over an infinite range, though he later proposes to compactify this.

Given this mathematical solution it is, he says, just a matter of engineering to build the machine: Fig. 8.1.

If we could have this halting function, surely we could just get a conventional computer to evaluate it.

But we know we can't do this, because were we able to do that we contradict Turing's proof. The problem is that we cannot compute on a TM the required infinite integrals—these integrals substitute for the potentially infinite time required to determine if a TM will halt. Da Costa writes that building the hypercomputing machine from the mathematical specification is just a matter of engineering.

Well, in that case it is a matter of engineering more suited to Olympians than us mere mortals, since it requires, among other things, encoding π in the machine to an infinite precision. Da Costa gives us no clue as to how he proposes to achieve this degree of precision. Others have rashly come forward with concrete proposals. It is rather easy to come up with ideas for computing devices that have a superficial plausibility, because they are expressed in terms of idealized physical models in which some real-world constraints are absent. In this section, we will look at a few such proposals. In every case we shall see that their apparent plausibility rests on a judicious choice to ignore key features of the physical world as understood today.

8.2.1 Newtonian computing

Newtonian mechanics is a particularly beguiling area for those exploring the possibilities of infinitary computation. What we call Newtonian mechanics is both that form of abstract maths originating in the calculus of Newton, and a set of laws that allow us to use this mathematical apparatus to make predictive models of reality. The abstract maths was initially controversial, with its fluxions and infinitesimals (Wisdom, 1953). But its predictive powers proved to be so great that philosophical doubts about the idea of infinite divisibility were put into abeyance.

When Newton's laws are used in a practical sense, what they do is stipulate a set of algorithms for making finite predictions about things such as observed planetary positions. They specify at a relatively

abstract level what a mathematician should do to compute the position of, for example, Saturn on a particular day. The calculations may be done by hand or they may be done on a computer.

Suppose that they are done on a computer. There are, then, a large number of possible programmes, in various programming languages that are valid applications of Newton's laws to the problem. According to the care with which the codes have been written, the numerical precision of the floating point calculation, and so on, these will give results of greater or lesser accuracy. But whatever we do, the predictions are always given as computable numbers—to some chosen finite number of digits.

The fact that calculations are always to a finite precision is not a fundamental problem. We can always choose a number of digits sufficient for a given practical purpose—whether it is pointing a telescope or navigating a probe to Jupiter. We can choose an algorithm in which the numerical errors will be a lot less than our uncertainties in observation and measurement of the actual angular position of Saturn or the actual orbit round Jupiter that our probe takes up.

Because we can use Newton's laws to write algorithms that compute movements to an arbitrary degree of numerical precision, a temptation arises to believe that in reality physical bodies do move with an infinite accuracy.

It has been known since the early twentieth century that, whilst Newtonian mechanics makes excellent predictions of a wide range of physical systems, at extremes of velocity and density and on very small scales, its success falters. Classical physics seems to falter at just the places where a relentless pursuit of its logic would lead us to infinities. Quantum theory was introduced in Einstein's paper on the photoelectric effect (Einstein and into English, 1965) in order to deal with the paradox created for classical electrodynamics by one such infinity—the so-called ultraviolet catastrophe. Einstein opened his paper by pointing to the inadequacies of a continuum approach, in this case the continuum presupposed by Maxwell's theory:

A profound formal distinction exists between the theoretical concepts which physicists have formed regarding gases and other ponderable bodies and the Maxwellian theory of electro-magnetic processes in so-called empty space. While we consider the state of a body to be completely determined by the positions and velocities of a very large, yet finite, number of atoms and electrons, we make use of continuous spatial functions to describe the electro-magnetic state of a given volume, and a finite number of parameters cannot be regarded as sufficient for the complete determination of such a state. According to the Maxwellian theory, energy is to be considered a continuous spatial function in the case of all purely electro-magnetic phenomena including light, while the energy of a ponderable object should, according to the present conceptions of physicists, be represented as a sum carried over the atoms and electrons. The energy of a ponderable body cannot be subdivided into arbitrarily many or arbitrarily small parts, while the energy of a beam of light from a point source (according to the Maxwellian theory of light or,

more generally, according to any wave theory) is continuously spread an ever increasing volume.

(Einstein and into English, 1965).

His response was to propose that light was quantized in the form of photons, and from this eventually followed the rest of the quantum theory.

If a modern paper suggests that some form of classical mechanics allows certain forms of infinitary calculation, what does this mean?

1. That we can logically deduce that certain equations will produce infinities in their solutions?
2. That certain real physical attributes can take on infinite values?

In the light of quantum theory, we have to answer ‘no’ to the last question even if we say ‘yes’ to the first. If the maths you are using to represent the material world gives infinities in your equations, then that tells you more about the errors in your mathematical model than it does about reality.

Smith (2006*b*) gives as an example of uncomputability in Newtonian physics certain N -body problems involving point masses interacting under gravity. Because these can approach arbitrarily close to one another, at which point their mutual gravitational attraction becomes arbitrarily high, he suggests that we could so configure a set of initial conditions that the particles would move through an infinite number of configurations in a finite time interval. He argues that no Turing Machine could simulate this motion in a finite time. This could be interpreted either:

- as a limitation on the ability of computers to simulate the world, or
- as a means by which, with suitable encoding, a physical system could be used to determine algorithmically uncomputable questions

But let us consider the very properties that would allow this infinitary process—infinite velocities, point masses with position and no magnitude. These are the very points where Newtonian mechanics breaks down and has to be replaced with relativistic or quantum mechanics. The ability of a theory to produce infinities points to a weakness in conceptualization. Smith concurs: he goes on to show that once relativistic constraints on either velocity or density are introduced, the equations of motion for the system give finite results, and in consequence become algorithmically soluble.

The infinities in the initial differential equations thus emerge as a consequence of the axiomatic structure of the calculus rather than a property of the real world.

Beggs and Tucker (2006) also explore the extent to which Newtonian mechanics would allow the hypothetical construction of infinitely parallel computing engines. They do not claim that such machines could actually exist, but ask: what sort of mechanics would allow such machines to exist?

This is an interesting, if speculative, line of enquiry. But to pursue it, we have to be consistent. It would be reasonable enough, when investigating the hypothetical computational possibilities of a Newtonian universe, to ignore constraints imposed by relativity theory and quantum theory. But Newtonian mechanics imposes other constraints that may not immediately be obvious.

Beggs and Tucker derive the ability to perform hypercomputation from an infinite plane of conventional computers, which purport to use tricks of Newtonian mechanics to allow infinitely fast transmission of information.

They use two tricks. On the one hand, they propose to synchronize the clocks of the machines by using infinitely long, rigid rods. The rod is threaded through all the machines and a push on the rod starts all the computers synchronously. They concede that for this to happen the rod must not only be perfectly rigid, but it must either be massless or have a density which exponentially tends to zero as we move away from the starting point. This is necessary if the rod is to be moved by applying a finite force.

It is not clear in what sense such rods can be said to be Newtonian.

There is an old technical term for rods like this: wands. When Turing had recourse to ‘Oracles’ he deliberately used magical language to indicate that this recourse was make-believe.

They propose that the infinite collection of computers will be able to return results to an initiating processor using an ability to fire cannonballs up and down along parabolic trajectories at arbitrarily high velocities. The arrival of such a cannonball transmits a binary truth value. They further propose that in a finite time interval an infinite number of cannonballs can be projected, in such a way that at any given instant only one is in flight.

Their argument is that given a projectile of mass m , we can project it at arbitrarily high speed if we use enough energy. Given a distance b that the cannonball has to travel, we can make the time of flight arbitrarily small by selecting a sufficiently high velocity of travel. The proposal is that we use cannons and an arbitrarily large supply of gunpowder to achieve this. This argument contradicts Newtonian mechanics on several points:

1. The immediate problem is that whilst there is, according to Newton, no limit to the ultimate velocity that a particle subjected to uniform acceleration can reach, this velocity is not reached instantaneously. Let us suppose that we use perfect cannons, ones in which the accelerating force f due to the combustion of gunpowder remains constant along the length of the barrel. Achieving this is very hard: it requires all sorts of constraints on the way the powder burns. It was a sought-after but never-achieved goal of nineteenth-century ballistic engineering: much experimentation with powder grain size went into the quest (see, e.g., Schenck, 1883). The acceleration of a ball of mass m will then be $a = f/m$. A cannonball spends a period in the cannon v/a

being accelerated that is proportional to the velocity ultimately attained. Thus whilst the flight time b/v tends to zero as velocity increases, total travel time $(b/v) + (v/a)$ does not.

2. There is a further problem with assuming that cannons can attain an arbitrary velocity. As we increase the charge in a gun, an increasing amount of the work done by the powder consists in accelerating the powder itself down the barrel. The limiting velocity achievable is that of the exit velocity of the gas of a blank charge. This, in turn, is limited by the speed of sound in the driving gas. For a highly energetic hydrogen/oxygen explosion, this sets a limit of about 2100 m s^{-1} . Techniques such as hybrid explosive and light gas guns can produce a limited improvement in velocity (Crozier and Hume, 1957), but certainly not an arbitrary speed. Rail guns (Rashleigh and Marshall, 1978) can achieve higher velocities using electromagnetic acceleration. It is not clear whether electromagnetic propulsion is permissible in Begg and Tucker's chosen model of mechanics.
3. There is also a problem of the trajectory. Begg and Tucker further assume parabolic trajectories to ensure that the cannonballs fly clear of intervening obstacles such as other cannons prior to hitting their targets (see Fig. 8.2). The balls will take a finite time for gravity to retard their upward velocities.

Even on its own Newtonian terms, the proposed hypercomputer is inconsistent. It is analogous to Fredkin's billiard ball computer—superficially plausible but, on examination, inconsistent.

8.2.2 Bournez and Cosnard's analogue super-Turing computer

An examination of a concrete example of another couple of proposed super-Turing analogue computers (Bournez and Cosnard, 1995; Blakey, 2008) illustrates the sorts of errors that would vitiate its operation. Bournez and Cosnard propose to use two real-valued variables corresponding to the x, y coordinates of particles (presumably photons)

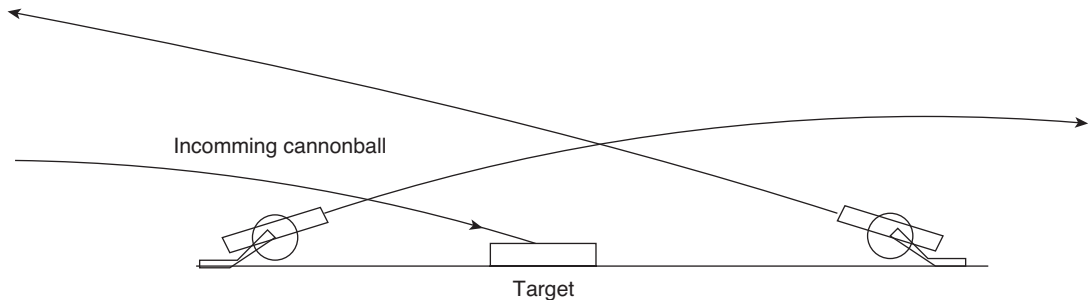
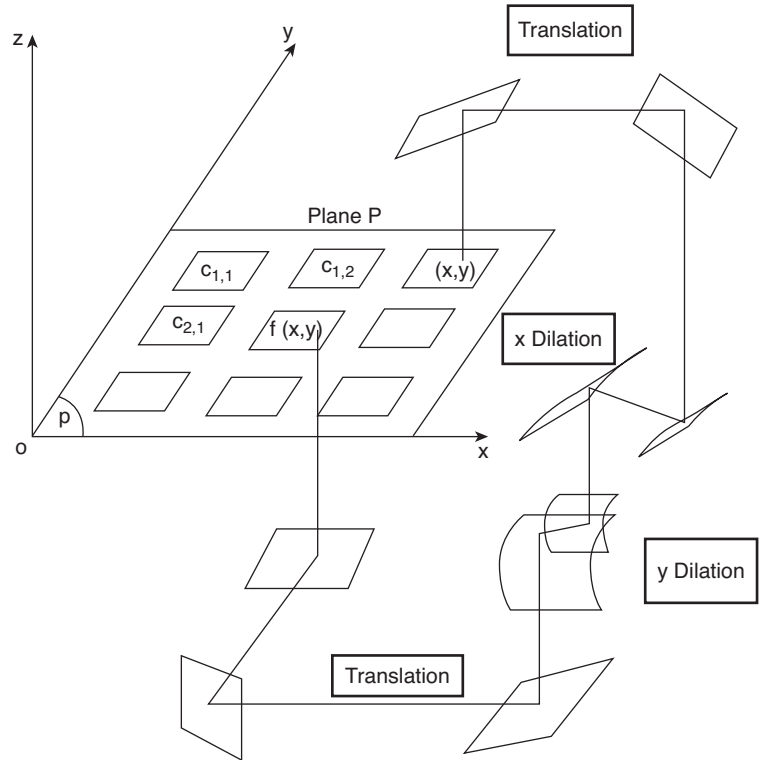


Fig. 8.2 Cannonballs and their clearances.

Fig. 8.3 An analogue computer proposed by Bournez and Cosnard.



passing through plane P in Fig. 8.3 (reproduced from Bournez and Cosnard, 1995). The binary expansion of these real-valued coordinates could then be used to emulate the left and right parts of a TM tape (Koiran and Moore, 1999; Bunimovich and Khlabytova, Bunimovich and Khlabytova). They argue that the machine could, in addition, be used to simulate a class of two stack automata whose computational powers might exceed those of TMs. The gain in power comes from the ability of their proposed stack automaton to switch on the basis of the entire contents of an unbounded stack, rather than on the basis of what is directly under the TM head. They suggest that if we had available iterated real-valued functional systems based on piecewise affine transforms, such analogue automata could be implemented. In the proposed physical embodiment given in Fig. 8.3, multiplication by reals would be implemented by pairs of parabolic mirrors, and translation by arrangements of planar ones.

The authors, in striking contrast to researchers active in the area 50 years earlier, fail to identify the likely sources of error in their calculator. Like any other analogue system, it would be subject to parameter, operator, and variable errors. The parameters of the system are set by the positions and curvatures of the mirrors. The placement of the mirrors would be subject to manufacturing errors, and to distortions due to temperature, mechanical stress, and so on. The parabolic mirrors

would have imperfections in their curvature and in their surfaces. All of these would limit the number of significant digits to which the machine could calculate. But let us, for the moment, ignore these manufacturing errors and concentrate on the inherent uncertainty in the variables.

Because of the wave-particle duality, any optical system has a diffraction limited circle of confusion. We can say that a certain percentage of the photons arriving from a particular direction will land within this circle. The radius of the circle of confusion is inversely proportional to the aperture of the optical system and directly proportional to the focal length of the apparatus and to the wavelength of the photons used. The angle to the first off-centre diffraction peak $\Delta\Theta$ is given by

$$\sin(\Delta\Theta) = \frac{\lambda}{A} \quad (8.1)$$

where A is the aperture and λ is the wavelength.

By constraining the position of the photon to be within the aperture, we induce, by Heisenberg's principle, an uncertainty in its momentum within the plane of the aperture.

To see what this implies, we give some plausible dimensions to the machine. Assume the aperture of the mirrors to be 25 mm and the path length of a single pass through the system from the first mirror shown in Fig. 8.3 back to Plane P to be 500 mm. Further assume that we use monochromatic light with wavelength $\lambda = 0.5 \mu\text{m}$. This would give us a circle of confusion with a radius $\Delta_{f(x,y)Mirror} \approx 10 \mu\text{m}$.

If plane P was one-tenth of a metre across, the system would resolve about 5000 distinct points in each direction as possible values for $f(x, y)$. This corresponds to about 12 bits accuracy.

The dispersion $\Delta_{f(x,y)Mirror}$ accounts only for the first pass through the apparatus. To begin with, let us look at the parametric uncertainty in x, y .

We want to specify x, y to greater accuracy than $f(x, y)$ so that $\Delta_{x,y} < \Delta_{f(x,y)}$. Assume that we have a source of collimated light whose wavefronts are normal to the axis of the machine. A mask with a circular hole could then constrain the incoming photons to be within a radius $< 10 \mu\text{m}$. Any constraint on the position of the photons is an initial aperture. If this aperture $\leq 10 \mu\text{m}$, its diffraction cone would have a $\Delta\Theta \approx 0.05$ radians. Going through the full optical path, the resulting uncertainty in position $\Delta_{f(x,y)Mask(\Delta_{x,y})} \approx 25 \text{ mm}$. We have $\Delta_{f(x,y)Mask(\Delta_{x,y})} \gg \Delta_{f(x,y)Mirror}$.

The narrower the hole in the mask, the more uncertain will be the result $f(x, y)$. In other words, the lower the parametric error in the starting point, the greater is the error in the result.

There will be a point at which the errors in the initial position and the errors due to the diffraction from the mirrors balance: when $\Delta_{f(x,y)Mirror} + \Delta_{f(x,y)Mask(\Delta_{x,y})} \leq \Delta_{x,y}$. From simple geometry, this will come about when the ratio $\Delta_{x,y}/L \approx \lambda/\Delta_{x,y}$ so

$$\Delta_{x,y} \approx \sqrt{L\lambda} \quad (8.2)$$

where L is the optical path length of the computation. For the size of machine that we have assumed above, this implies $\Delta_{x,y} = 500 \mu\text{m}$. Its accuracy of representation of the reals is thus less than 8 bits ($= -\log_2[(500 \mu\text{m})/(100 \text{mm})]$), hardly competitive with existing digital computers.

The evaluation of $f(x,y)$ corresponds to a single step of a TM program. If n is the number of TM steps, the optical path length is nL . By Eq. 8.2, the optimal initial aperture setting $\Delta_{x,y} \propto \sqrt{n}$. Each fourfold increase in the execution length of the program, will reduce by 1 bit the accuracy to which the machine can be set to compute.

If we want to make an optical machine more accurate, we have to make it bigger—the growth in the size of astronomical telescopes bears witness to this. For every bit of accuracy we add, we double the linear dimensions. If M is the mass of the machine and b its bit accuracy, then $M \propto 2^{3b}$.

For a conventional digital VLSI machine, $M \propto b$ and the mass of the arithmetic unit grows as $b \log b$. For any but the least accurate calculations, this sort of optical analogue machine will be inferior to conventional machines.

8.2.3 Optical prime factorization

The fame accorded to Shor's prime factorization algorithm encouraged another attempt at the use of interference as a mechanism for prime factorization. Shor relied on quantum interference, whereas this proposal relies upon optical interference. Blakey's design for a prime factorization machine (Blakey, 2008, Blakey) displays considerable geometric ingenuity.

It addresses the problem of finding the prime factors of a large integer. This problem is classically of order \sqrt{n} for an integer n . Basically, we just try dividing all possible factors up to the square root, at which point you will have found all the factorizations \sqrt{n} operations. Blakey observes that if you have a grid in two dimensions as in Fig. 8.4, then a hyperbolic curve $y = N/x$ will pass through the prime factors of N .

He then asks whether we can set up a physical apparatus that will generate the required grid and a hyperbolic curve. His suggested equipment is shown in Fig. 8.5.

A point source of light S along with a group of three mirrors is used to set up a grid pattern of constructive and destructive interference across the plane between the mirrors M_2, M_3 . The hyperbola is then created as the plane intersection between the conic arc P, C , and the mirror plane. Along the circle C that forms the base of the cone, there are photo detectors.

The principle of operation is that:

Diminution of second-source radiation due to its having passed through an integer (that is, maximally active) point in the first-source interference pattern (which models the grid of integer points) is detected at the sensor; the

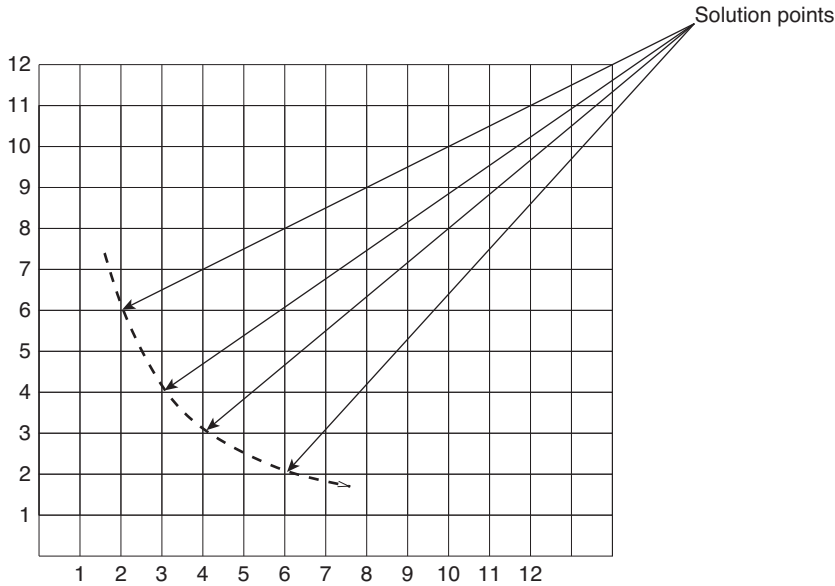


Fig. 8.4 The hyperbolic curve $y = N/x$ passes through the grid points that are prime factors of N , illustrated with $N = 12$.

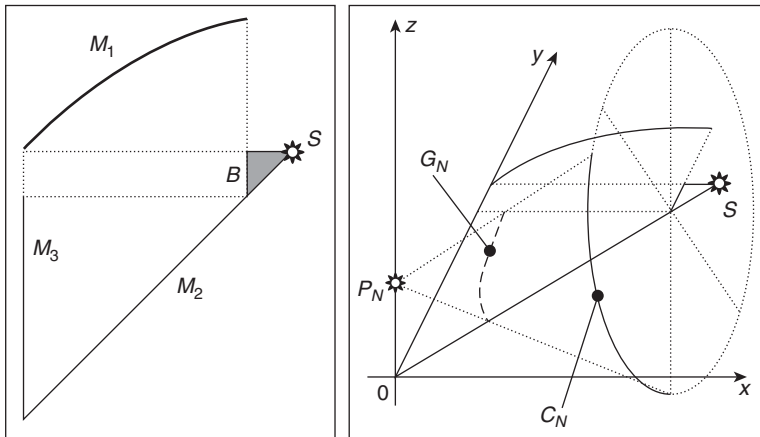


Fig. 8.5 Left, a plan view of the machine, M_i mirrors, S light source. Right, the apparatus to generate the hyperbola as a conic section. P is another light source, and C a circle arc made up of detectors.

coordinates of points of such detection can, Turing-computationally trivially, be converted into the coordinates of the sought integer points on the cone. These latter coordinates are factors of n .

(Blakey, 2009, p. 4).

It is unclear why the interference caused by the standing wave pattern in the mirror plane is supposed to impede the passage of light beams on the path from P to C . Light beams pass through one another without effect. We only observe interference if there is more than one possible route that a photon can have taken between source and destination. The photons arriving at the detectors on C can only have come from P , so there will be no interference between these and the photons emitted from S .

As described, the machine would not work. It can be modified in a sympathetic way to make it work roughly as intended. Blakey does not mention the use of any photosensitive materials. We could imagine a modification of his design so that, during a first phase, an initially transparent photosensitive plate is placed in the mirror plane. The light from S could be used to expose the plate, darkening it at points of constructive interference. There are of course a variety of other ways in which we could lay out such a grid, but Blakey's interference patterns are one possible technique.

Given that we have a grid, is the rest of the design plausible?

The basic design seems to assume that a very fine grid will cast a shadow pattern that will also be a very fine grid. But this is not so. A grid with holes whose size is comparable to a wavelength will act as a two-dimensional diffraction grating, which will split the original light rays from P into a strong principal component and a several diffracted components. There will be no shadows cast by the grid and hence the desired effect on the detector will not be observed, since the principal component will seem to pass directly through though with diminished brightness.

To get a shadow cast, you would require a grid that was large relative to the wavelength of the light and the focal length used. Figure 8.6 illustrates the constraints. One constraint that Blakey fails to take into account is that the sensors will have gaps between them and there is the possibility that the peak in brightness corresponding to a hole in the mask will coincide with a gap. This is an instance the well-known problem of aliasing in digital photography. But ignoring that problem for a moment, let us consider how to overcome the blurring caused by diffraction. We could constrain the separation between the principal and first diffracted components arriving on the sensor array to be no more than half the separation between sensors. This gives us the constraint

$$w/A < \frac{1}{2}B/F \quad (8.3)$$

Let us add the further constraint that $A = cB$ with $c > 1$ in order to have enough sensors to have some hope of compensating for the

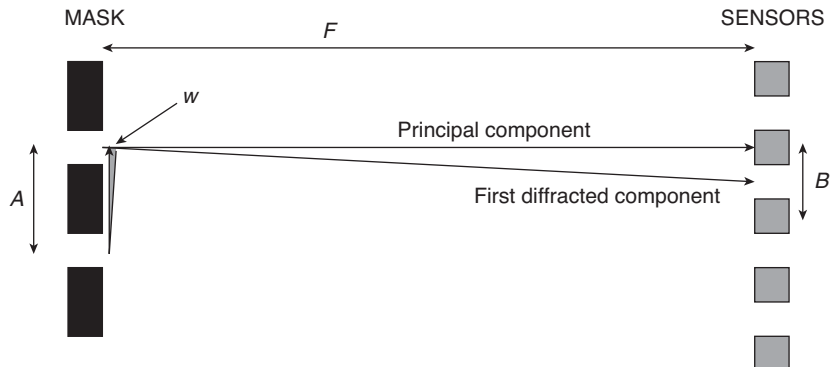


Fig. 8.6 A detailed view of how a mask interacts with a sensor array. The wavelength of the light is w , the separation of the mask and the sensor array is F , and A and B are the separations of the mask holes and the sensors, respectively.

aliasing problem. Let us set $c = 2$. We thus have the derivation

$$\begin{aligned}\frac{w}{2B} &= \frac{B}{2F} \\ 2wF &= 2B^2 \\ B &= \sqrt{wF}\end{aligned}$$

Since we can only currently build sensors of this density of about 4 cm across, we can use Blakey's diagrams to fix a plausible value of the focal length F to be about 2 cm. Let us assume that we use visible light with a wavelength of $0.7 \mu\text{m}$. This gives us a figure for B , the sensor separation, of about 0.1 mm and a grid spacing of twice that. If we assume that the practical size of the grid is set by the sensor dimensions to also be about 4 cm, we see that the optical apparatus could hope to determine the prime factors of numbers up to about 200. Finding prime factors in this range is not a serious problem for digital techniques. Since the prime numbers used in cryptography tend to be of the order of 2^{256} , it is evident that the analogue technique is of far too low a precision to be of use.

The algorithmic complexity would be worse than on an orthodox machine. The problem would be to find the maximum brightness over the sensors, which are of order N , so that the time complexity is N rather than \sqrt{N} on the conventional computer. The space complexity is N^2 , since the area of the grid grows in this manner.

8.3 Wegner and Eberbach's super-Turing computers

Wegner and Eberbach (2004) assert that there are fundamental limitations to the paradigmatic conception of computation that are overcome by more recent 'super-Turing' approaches. We will now summarize their core arguments before exploring them in greater detail. This material draws directly on Cockshott and Michaelson (2007).

Wegner and Eberbach depend strongly on the idea of an algorithm as an essentially closed activity. That is, while the TM realizing an algorithm may manipulate an unbounded memory, the initial memory configuration is pre-given and may only be changed by the action of the machine itself. Furthermore, an effective computation may only consume a finite amount of the unbounded memory and of time, the implication being that an algorithm must terminate to be effective.

They say that the TM model is too weak to describe the Internet, evolution, or robotics. For the Internet, web clients initiate interactions with servers without any knowledge of the server history. The Internet, as a dynamic system of inputs and outputs, parallel processes, and communication nodes, is outside the realm of a static, sequential TM. Furthermore, TMs cannot capture evolution because the solutions and algorithms are changed in each generation and the solution search is

an infinite process. This does not depend on a finite or infinite search space. Because evolutionary algorithms are probabilistic, the search may take an infinite number of steps over a finite domain. Finally, robots interact with non-computable environments that are more complex than the robots themselves.

Wegner and Eberbach claim that there is a class of super-Turing computations (sTCs) that are a superset of TM computations; that is, sTCs include computations that are not realizable by a TM. A super-Turing computer is ‘any system or device which can carry out super-Turing computation’.

Most significantly, Wegner and Eberbach say that it is not possible to describe all computations by algorithms. Thus they do not accept the classic equation of algorithms and effective computations.

They go on to argue that there are three known systems that are capable of sTC: interaction machines (IM), the π -calculus, and the $\$$ -calculus. They give discursive presentations of these systems and explore why they transcend the TM.

8.4 Interaction Machines

Wegner and Eberbach refer to Interaction Machines as a class of computer that is more powerful than the Turing Machine. The latter, they claim, is restricted by requiring all its inputs to appear on the tape prior to the start of computation. Interaction Machines, on the contrary, can perform input output operations to the environment in which they are situated. The difference between Turing Machines and Interaction Machines, they claim, corresponds to the technology shift from mainframes to workstations. Interaction Machines, whose canonical model is the Persistent Turing Machine (PTM) of Goldin *et al.* (2004), are not limited to a pre-given finite input tape, but can handle potentially infinite input streams.

This argument was originally advanced by Wegner in a previous publication (Wegner, 1997*b*), some of whose main arguments have been criticized by Ekdahl (1999). Rather than rehearse Ekdahl’s critique, we shall focus on some additional weaknesses of Wegner and Eberbach’s claims.

8.4.1 Turing’s own views

As is well known, Turing’s contribution to computer science did not stop with the Turing Machine. Besides his work on cryptography, he played a seminal role in the establishment of Artificial Intelligence research. His Turing Test for machine intelligence is probably as well known as his original proposal for the Universal Computer. He proposed, in a very readable paper (Turing, 1950), that a computer could be considered intelligent if it could fool a human observer into thinking that they were interacting with another human being. It is clear that his putative intelligent machine would be an Interaction Machine in Wegner’s sense.

Rather than being cut off from the environment and working on a fixed tape, it receives typed input and sends printed output to a person.

Turing did not, however, find it necessary to introduce a fundamental new class of computing machine for this gedanken experiment. He is quite specific that the machine to be used is a digital computer and he goes on to explain just what he means by such a machine:

The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail. We may suppose that these rules are supplied in a book, which is altered whenever he is put on to a new job. He has also an unlimited supply of paper on which he does his calculations. He may also do his multiplications and additions on a 'desk machine', but this is not important.

(Turing, 1950, p. 436).

This is, of course, a paraphrase of his description of the computing machine in his 1936 paper (Turing, 1937), where he explicitly models his machine on a person doing manual calculations. The states of the machine correspond to the finite number of states of mind of the human mathematician and the tape corresponds to the squared paper that he or she uses. It is clear that Turing is talking about the same general category of machine in 1950 (Turing, 1950) as he had in 1936 (Turing, 1937). With the practical experience of work on the Manchester Mk 1 and the ACE behind him, he speaks in more general terms of the computer as being composed of (i) store, (ii) executive unit, and (iii) control, and says that the store locations are addressable rather than being purely sequential. He says he is concerned with discrete state machines, and that a special property of such digital computers was their universality:

This special property of digital computers, that they can mimic any discrete state machine, is described by saying that they are universal machines. The existence of machines with this property has the important consequence that, considerations of speed apart, it is unnecessary to design various new machines to do various computing processes. They can all be done with one digital computer, suitably programmed for each case. It will be seen that as a consequence of this all digital computers are in a sense equivalent.

(Turing, 1950, p. 442).

This is clearly a recapitulation of the argument in section 6 of his 1936 paper (Turing, 1937), where he introduced the idea of the Universal Computer. Turing argued that such machines were capable of learning and that with a suitable small generalized learning program and enough teaching, then the computer would attain artificial intelligence.

8.4.2 Equivalence of Interaction Machines and Turing Machines

It seems that Turing considered that the class of machines that he had introduced in 1936 (Turing, 1937) had all of the properties

that Wegner and Goldin were later to claim for their Interaction Machines and Persistent Turing Machines. He may of course have been mistaken, but we think that Turing's confidence was well founded. It can be demonstrated that Turing Machines are powerful enough for the task.

Consider first a digital computer interacting in the manner foreseen by Turing in his 1950 paper (Turing, 1950), with teletype input/output. The teletypes of Turing's day had the capability of capturing keystrokes to paper tape, or of accepting paper tape input instead of keystrokes. Suppose, then, that we have a computer initialized with a simple learning program, following which it acquires more sophisticated behaviour as a result of being 'taught'. As the computer is taught, we record every keystroke on to paper tape.

Suppose that we initialize a second identical computer with the same program and, at the end of the first computer's period of interaction, we give to the second machine as an input the tape on which we have recorded the all the data fed to the first machine. With the input channel of the second machine connected to the tape reader, it then evolves through the same set of states and produces the same outputs as the original machine did. The difference between interactive input from a teletype and tape input as used by Turing in 1936 is essentially trivial. By a simple recording mechanism, we can emulate the behaviour of an interactive machine on another tape-input computer. This has been a widely used and practical test procedure.

In Turing (1950), he clearly assumes that his computer has a persistent or long-term memory. Wegner and Goldin Persistent TMs allow a machine to retain data on a tape so that it can be used in subsequent computations. They claim that the idea of a persistent store was absent in TMs. However, persistence only became an issue because the combination of volatile semiconductor storage and first-generation operating systems imposed on programmers a sharp pragmatic distinction between persistent disk store and volatile random access memory (Cockshott, 1982; Cockshott *et al.*, 1984). This distinction had not been present in a first generation of magnetic core based von Neumann machines, and was not included in the basic computational models of Turing and von Neumann.

A small modification to the program of a conventional TM will transform it into a PTM. Like Goldin, we will assume a three-tape TM, M_1 , with one tape T_1 purely for input, one tape T_2 purely for output, and one tape T_3 used for working calculations. We assume that tapes T_1 and T_2 are unidirectional, and T_3 is bidirectional. Such a machine can be emulated on Turing's original proposal by a suitable interleaving scheme on its single tape.

M_1 has a distinguished start state S_0 and a halt state S_h . On being set to work, it either goes into some non-terminating computation or eventually outputs a distinguished termination symbol τ to T_2 , branches to state S_h , and stops. We assume that all branches to S_h are from a state that outputs τ . Once τ has been output, the sequence

of characters on T_2 up to to τ are the number computed by the machine.

We now construct a new machine M_2 from M_1 as follows: replace all branches to S_h with branches to S_0 . From here, it will start reading in further characters from T_1 and may again evolve to a state in which it outputs a further τ on T_2 .

Machine M_2 now behaves as one of Goldin's PTMs. It has available to it the persisting results of previous computation on T_3 and these results will condition subsequent computations. It is still a classic TM, but a non-terminating one. It follows that PTMs, and thus Interaction Machines, of which they are the canonical example, are a sub-class of TM programs and do not represent a new model of computation.

8.4.3 Thermodynamic considerations

For Wegner and Eberbach, there is a fundamental difference between starting a TM over a given tape that is only changed by that TM, and where additional input comes from the environment. This alleged distinction may be further explored using an algorithmic information-theoretic argument.

Chaitin introduced algorithmic information theory (Chaitin, 1987), according to which the entropy of a binary number x is bounded by the number of bits of the shortest TM program that will output x . From this standpoint, there is a pragmatic difference between an isolated TM and one that can accept input from the environment.

A modern computer is initially built with a start-up ROM and a blank disk drive. The ROM typically contains a BIOS, but can be replaced by any other program that will fit on the chip. Let us suppose, realistically, that the ROM chip contains 2^{19} bits. Suppose that instead of a BIOS, we put in a ROM that performs some predefined algorithm, possibly using disk I/O, and in the process of computation outputs a stream of characters from the serial port. Chaitin's result indicates that if the program were to perform no input operations from the keyboard, mouse, CD, and so on, the entropy of the information on disk plus the information output on the serial line could not exceed $2^{19} + 1$, where the additional 1 bit would encode whether the initial blank state of the disk was a 1 or a 0. If the disk was much bigger than the boot chip, it could, for example, never really be randomized by the boot chip.

A practical BIOS chip will attempt to read input from keyboards, communications lines, or CD, starting a process that allows the disk to gain entropy from external sources. We know from experience that disks become cluttered and entropic over time. The external world acts as an entropy source, causing the disk entropy to rise in conformance with thermodynamic laws.

Thus, we can formulate the interaction process within Chaitin's framework, which is in turn grounded in TM theory. This again implies that Interaction Machines are not a new class of computer.

8.5 π -Calculus

Wegner and Eberbach give the π -calculus as another of their three super-Turing computational models. The π -calculus is not a model of computation in the same sense as the TM: there is a difference in level. The TM is a specification of a material apparatus that is buildable. Calculi are rules for the manipulation of strings of symbols and these rules will not do any calculations unless there is some material apparatus to interpret them. Leave a book on the λ -calculus on a shelf along with a sheet of paper containing a formula in the λ -calculus and nothing will happen. Bring along a mathematician, give them the book and the formula and, given enough paper and pencil, the ensemble can compute. Alternatively, feed the suitably expressed λ -calculus formula into a computer with a Lisp interpreter and it will evaluate.

We have to ask if there exists any possible physical apparatus that can implement the π -calculus and, if there does, whether it is a conventional computer such an apparatus. Since it is possible to write a conventional computer program that will apply the formal term rewrite rules of the π -calculus to strings of characters representing terms in the calculus, then it would appear that the π -calculus can have no greater computational power than the von Neumann computer on which the program runs. The language Pict (Turner and Pierce, 2000) is an example of this. Since it is also possible to implement the λ -calculus in the π -calculus (Milner, 1993), we can conclude that the π -calculus is just one more of the growing family of computational models that are Turing Machine equivalent.

A possible source of confusion is the terminology used to describe the π -calculus—channels, processes, evolution—which implies that we are talking about physically separate but communicating entities evolving in space–time. The π -calculus is intended to be used as a language to describe communicating physically mobile computing machines such as cell phones and their underlying communications networks. As a result, there is always a tension between what is strictly laid down as the rules of a calculus and the rather less specific physical system that is suggested by the language used to talk about the calculus.

We have to be very careful before accepting that the existence of the π -calculus as a formal system implies a physically realizable distributed computing apparatus.

Consider two of the primitives: synchronization and mobile channels. We will argue that each of these faces limits to their physical implementation that prohibits the construction of a super-Turing computational engine based on the the calculus.

8.5.1 Difficulties in implementing π -calculus synchronization

It is not clear that π -calculus synchronization is, in its general sense, physically realistic. First, it seems to imply the instantaneous

transmission of information, that is faster than light communication, if the processes are physically separated.

Furthermore, if the processors are in relative motion, relativity theory shows that there can be no unambiguous synchronization shared by the different moving processes. It thus follows that the processors can not be physically mobile if they are to be synchronized with at least three-way synchronization (see Einstein, 1920, pp. 25–26).

Suppose that we have the following π -calculus terms:

$$\alpha \equiv (\bar{a}v.Q) + (by.R[y]) \quad (8.4)$$

$$\beta \equiv (\bar{b}z.S) + (ax.T[x]) \quad (8.5)$$

In the above, α and β are processes. The process α tries to either output the value v on channel a or to read from channel b into the variable y . The $+$ operator means non-deterministic composition, so $A + B$ means that either A occurs or B occurs, but not both. The notation $\bar{a}v$ means output v to a , whilst av would mean input from a into v . If α succeeds in doing an output on channel a , it then evolves into the abstract process Q ; if, alternatively, it succeeds in doing an input from b into y , then it evolves into the process $R[y]$, which uses the value y in some further computation.

We can place the two processes in parallel by using the $|$ operator for parallel process composition to form $\alpha|\beta$, which expands to the following:

$$(\bar{a}v.Q) + (by.R[y]) | (\bar{b}z.S) + (ax.T[x]) \quad (8.6)$$

This should now evolve to

$$(Q|T[v]) \text{ or } (S|R[z]) \quad (8.7)$$

where either Q runs in parallel with $T[v]$ after the communication on channel a or S runs in parallel with $R[z]$ after the value z was transferred along channel b from process β to process α . The key ideas here are processes, channels, synchronization, and parallel and non-deterministic composition.

Suppose further that we attempt to implement the synchronization by a standard three-wire handshake wherein each channel is represented as follows:

rqs request to send
ack acknowledge
d data

The protocol is as follows:

Put:

1. place data on **d** and assert **rqs** then wait for **ack**
2. on getting **ack**, negate **rqs** and remove data from **d**
3. wait for **ack** to be negated

Get:

1. wait for **rqs**, then latch **data**
2. assert **ack**
3. wait for **rqs** to be negated
4. negate **ack**

The two processes are shown in Fig. 8.7, with lines representing the wires used to implement the channel. But instead of wires we could think of these as being radio signals, each at a different frequency. Let us consider how the time evolution of the processes might proceed. We will indicate times as t_0, t_1, \dots :

- t_0 process α places data v on line \mathbf{d}_a and asserts \mathbf{rqs}_a
- t_1 process β places data z on line \mathbf{d}_b and asserts \mathbf{rqs}_b
- t_2 process β gets the \mathbf{rqs}_a
- t_3 process α gets the \mathbf{rqs}_b

What happens now?

At time t_2 , process β has attempted to send on channel b and has got a request to send from channel a , so which of these should it act on?

If it responds to the **rqs** on a with an acknowledge, it will be committing itself to evolving to $T[x]$, but it also has an outstanding **rqs** on b . Suppose that it commits to $T[x]$ by sending **ack** _{a} ; then it can ignore any further communications on channel b . This is fine for process β considered in isolation, but poses problems for the other process. Since we are talking about a general implementation strategy, we have to assume that process α will follow the same rule. Thus after getting the request to send on channel b , it too will acknowledge, which means that it will commit to continuation $R[y]$. The consequence is that we have

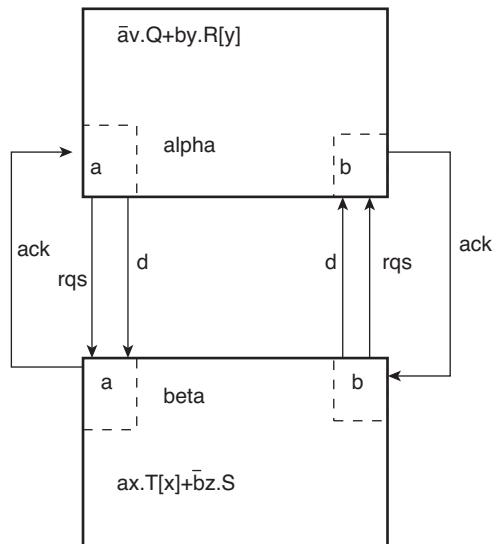


Fig. 8.7 Two paradoxical processes.

evolved to $T[x]|R[y]$, but this is not a permitted transition according to the π -calculus.

Suppose, instead, that β does not send an \mathbf{ack}_a but gives priority to its outstanding request to send on channel a . In this case, we have to assume that process α will likewise postpone transmission of \mathbf{ack}_b , since α is the mirror image of β . It follows that neither process will ever get an \mathbf{ack} , so they will deadlock.

This is not just a reflection of an inadequacy in the two-stage handshake protocol. Since the two processes are identical mirror images of one another, any deterministic rule by which process β commits to communication on one of the channels must cause α to commit to the other channel and hence synchronization must fail. The argument from the processes α, β is a variant of the Liar Paradox, but it is not a paradox within the π -calculus itself. It only emerges as a paradox once you introduce the constraints of relativity theory prohibiting the instantaneous propagation of information. Nor does abandoning determinism help. If the commitment process is non-deterministic, then on some occasions synchronization will succeed, but on other occasions the evolution of both processes will follow the same rule, in which case synchronization will fail.

The arbitration problem is not insoluble. Suppose that there was a global arbitration machine. Each process attempting a guarded non-deterministic fork could inform the arbiter of the channel names and the direction of communication being tried. Then, with knowledge of all outstanding requests for reads and writes, it would probably be possible for the arbiter to apply the reduction rules of the calculus to resolve the synchronization. However, the use of the arbiter machine as a tie-breaker removes the parallelism that we want from a distributed version of the calculus, returning us to a sequential centralized evaluation of a key part of the calculus. A worse loss of parallelism is entailed by broadcast protocols such as Asynchronous Byzantine Agreement (Bracha and Toueg, 1983).

In conclusion, it is not possible to build a reliable mechanism that will implement in a parallel distributed fashion any arbitrary composition of π -calculus processes.

More tractable systems intellectually derived from the π -calculus can be devised. The $A\pi$ -calculus (Sangiorgi and Walker, 2001) has no synchronization, and its processes terminate as soon as they output a message. This should be implementable, if restrictive. The Java library for Grid computing $J\pi$ (Yarmolenko *et al.*, 2004) does away with non-deterministic guards on output, but retains the ability to transmit channels over channels. This provides a practical tool for the parallelization of algorithms in a manner analogous to MPI (Message Passing Interface) (Walker and Dongarra, 1996), PVM (Parallel Virtual Machine) (Ferrari, 1998; Geist, 1994), or *IceT* (Gray and Sunderam, 1997). Parallelization speeds up sequential code, but it does not allow you to solve problems that are TM uncomputable.

8.5.2 Difficulties in implementing channels

As with synchronization, it is not clear how channels may be implemented in terms of physical law. If we have a physical network of processors that do not move physically, then we can connect them by wires or fibre optic cables, but:

1. These only allow fixed point-to-point communication.
2. They are limited in terms of the number of physical wires that can be fed into any given processing unit. Indeed, the number of wires that can be connected to a given chip has always been one of the main limitations on computer chips and remains a substantial technical challenge.

Taking into account (1), it is clear that a system using fixed point-to-point communication can only emulate a system with dynamically created communications channels by using multiplexing and forwarding of messages. From the point of view of computational models, this implies that we would have to emulate the π -calculus on the sort of parallel fixed link network that can be described by CSP. Let us refer to a π -calculus system emulated on a fixed link network as Π_{csp} .

If we do not assume wires or optical fibres, but instead assume a broadcast network using radio waves, like GSM, then we can have physically mobile processes, but at the expense of removing simultaneous overlapping communication. A system such as GSM relies on time multiplexing the radio packets so that, in fact, only a small finite number of the processes can send messages at any one instant—one process per frequency slot. Of course, GSM relies on base stations to forward packets along fixed-point links, but a system such as Aloha used basically similar techniques without the base stations.

It is evident that the π -calculus can be used to reason about the behaviour of, and protocols for, phones and other computing devices using radio networks: such problems were a motivation for its design. It would be reasonable to accept that the behaviour of any physical, wireless-linked, computer network can be described in the π -calculus.

However, it does not follow from this that there can exist a physically constructable wireless network whose evolution will *directly* emulate that of an arbitrary term in the π -calculus. Because of the exponential decay of signal strength with distance and the finite bandwidth of the radio channel, there are limits to the number of mobile agents that can simultaneously engage in direct communication. We can allow *indirect* communication by partitioning both the bandwidth and the machines, setting aside one group of machines to act as relays, and dividing frequency slots between those used by the relay machines and mobile machines. But relying on indirect communication would amount to emulating the π -calculus behaviour in Π_{csp} style. Since the number of directly communicating mobile processes that can operate without relays is modest, and since such a finite network of mobile processes could itself be emulated Π_{csp} style on a finite fixed-link network, the

computational power of physically realizable systems programmed in π -calculus will not exceed that of a formalism such as CSP, which would render it equivalent to other well-known computational models including Turing Machines.

8.5.3 Wegner and Eberbach's argument

Wegner and Eberbach's argument for the super-Turing capacity of the π -calculus rests on there being an implied infinity of channels and an implied infinity of processes. Taking into account the restrictions on physical communications channels, the implied infinity could only be realized if we had an actual infinity of fixed-link computers. At this point we are in the same situation as the Turing Machine tape—a finite but unbounded resource. For any actual calculation a finite resource is used, but the size of this is not specified in advance. Wegner and Eberbach then interpret 'as many times as is needed' in the definition of replication in the calculus as meaning an actual infinity of replication. From this, they deduce that the calculus could implement infinite arrays of cellular automata, for which they cite Garzon (1995) to the effect that the calculi are more powerful than TMs.

We undercut this argument at two points:

1. We have shown that the synchronization primitive of the calculus is not physically realistic. The modelling of cellular automata in the calculus rests on this primitive.
2. The assumption of an infinite number of processes implies an infinity of mobile channels, which are also unimplementable.

We therefore conclude that whilst the π -calculus can be practically implemented on a single computer, infinite distributed implementations of the sort that Wegner and Eberbach rely upon for their argument cannot be implemented.

It is important to emphasize that, just as we are untroubled by an unknown but bounded TM tape, we have no concerns about the deployment of an unknown but bounded number of processes in the π -calculus. However, Wegner and Eberbach are unclear as to whether they mean this or a completed infinity of processes, which we think physically impossible.

8.6 $\$$ -Calculus

8.6.1 Evolution and effective computation

The $\$$ -calculus ('cost' calculus; Eberbach, 2000; E.Eberbach, 2001) is based on a process algebra extended with cost operators. The $\$$ -calculus draws heavily on the π -calculus and on interaction machines: thus the critiques of these in the section below also apply to the $\$$ -calculus.

A central claim is that 'The unique feature of the $\$$ -calculus is that it provides a support for problem-solving by incrementally searching for solutions and using cost to direct its search' (p. 7). Furthermore, 'The

' $\$$ -calculus allows in a natural way to express evolution' (p. 7). Let us first examine the argument that evolutionary computing is not algorithmic before considering the $\$$ -calculus itself in more detail.

Eberbach (2002) characterizes an evolutionary algorithm (EA) as involving the repeated selection from a population under some selection operation for reproduction that introduces variations through mutation and crossover. When some selection threshold is reached, the final population is deemed to be optimal. This characterization thus far corresponds to Kleene's unbounded minimization; that is, general recursion. Eberbach goes on to elaborate a hierarchy of EAs and corresponding TMs. The Universal Evolutionary Algorithm (UEA) consists of all possible pairs of EA TMs and initial population tapes. An Evolutionary Turing Machine (ETM) is a potentially infinite sequence of pairs of EA TMs and population tapes, where each pair was evolved in one generation from the previous pair subject to a common fitness (selection) operator.

Eberbach claims that evolution is an infinite process because the fitness operator is part of the TM and evolves along with it. This seems unremarkable: it is well understood that a Universal TM may execute a TM that modifies its own encoding. Hence, the TM's termination condition may change and may never be achieved.

Eberbach then makes the apparently stronger claim that EAs are a superset of all algorithms, but this is either unremarkable or misleading. EAs are generalized unbounded minimization and so are expressible as general recursion or TMs. Given that any effective computation can be captured as a TM or through general recursion, it seems plausible that any effective computation can be evolved. However, it is not at all clear how we would define a selection operator to decide if a required effective computation had indeed been achieved. As noted above, the equivalence of TM is undecidable, so even if we could specify the required effective computation as another TM, there is no effective method for proving that an arbitrary evolved TM is equivalent to the specification. Furthermore, for EAs to be a superset of all algorithms, there must be something in the set of EAs that is not itself an algorithm. However, Eberbach says that all EAs can themselves be encoded as TMs, so all EAs must be algorithms. Thus, it seems more likely that the set of algorithms is at least as big as the set of all EAs.

Finally, Eberbach introduces the Universal Evolutionary Turing Machine (UETM), which takes an ETM as its initial tape. He states in theorems that the UETM halting problem is unsolvable by the UTM, and that the UTEM cannot solve its own halting problem.

Eberbach speculates that ETM can be understood as a special case of the Persistent Turing Machine. Thus the ETM must answer the critique of Persistent Turing Machines made above. Eberbach goes on to enunciate two more theorems. First, he claims that the UTM halting problem is solvable by the ETM using the 'infinity principle', where 'Fitness is defined to reach optimum for halting instances.' As noted already, we do not think it possible to construct an effective computation

for such a fitness function. He then claims that the UTM halting problem is solvable by ETM using evolution through a TM augmented with an oracle. This argument again seems plausible—though curious, given that Eberbach and Wegner say that the $\$$ -calculus does not gain its power from an oracle (p. 7). However, as discussed above, if an ETM is a TM with an oracle, then ETMs are not effectively computable and in general are not materially realizable.

8.6.2 $\$$ -calculus and expressiveness

In Eberbach (2000), Eberbach explores the expressiveness of the $\$$ -calculus. First, he shows that the λ -calculus and the π -calculus may both be simulated in the $\$$ -calculus. He claims that ‘the λ -calculus is a subclass of the $\$$ -calculus, because of the one-to-one correspondence between reductions in λ -terms and in their corresponding $\$$ -calculus terms’ (p. 5); and that ‘ π -calculus could be claimed to be a subclass of the $\$$ -calculus, because each operator of π -calculus is simulated by a corresponding operator(s) from $\$$ -calculus’ (p. 5). We dispute the claimed subclass relationship, but note that this is not expressed in (1) below and do not consider this further.

Next, citing Milner’s proof that λ -calculus may be simulated by π calculus (Milner, 1992), and drawing implicitly on the Church–Turing equivalence of λ -calculus and TMs, he enunciates a hierarchy:

$$(1) \quad TM \subseteq \pi C \subseteq \$C$$

We note the use of \subseteq rather than \subset in $\pi C \subseteq \$C$. Certainly, Eberbach does not substantiate a strong hierarchy (\subset) at this stage by demonstrating a $\$$ -calculus term that cannot be expressed as a π -calculus term.

Next, Eberbach says that Sequential Interaction Machines (SIMs) have less expressive power than Multi-stream Interaction Machines (MIMs), and cites Wegner’s claim (Wegner, 1997a) that π -calculus has only the same expressive power as SIMs, giving the additional hierarchy:

$$(2) \quad \pi C \subseteq SIM \subset MIM$$

Finally, he sketches how MIMs may be simulated by $\$$ -calculus ($MIM \subseteq \$C$), which combined with (1) and (2) gives:

$$(3) \quad TM \subseteq \pi C \subseteq SIM \subset MIM \subseteq \$C$$

Note that (3) greatly strengthens the $\$$ -calculus’s position in the hierarchy relative to Church–Turing systems: (1) says that $\pi C \subseteq \$C$, but (3) now implies that $\pi C \subset \$C$ through transitivity of \subseteq and \subset . Thus, the $\$$ -calculus term that is not expressible in π -calculus may come from an instance of MIM, but this is not displayed.

8.6.3 $\$$ -calculus and costs

As noted above, the $\$$ -calculus (E.Eberbach, 2001) is characterized by the integration of process algebra with cost functions derived from

von Neumann/Morgenstern utility theory. As well as sequential and parallel composition, and inter-\$-expression communication, cost choice and mutating send constructs are also provided. Rather than having a unitary expression form, the \$-calculus distinguishes between composite (interruptible) expressions, and simple (contract) expressions that are considered to be executed in one atomic indivisible step. While cost choices are made in composite expressions and costs communicated in simple expressions, they may be defined and evaluated in both layers.

Composition and choice are over countably infinite sets of \$-expressions, which Eberbach claims as one locus of the \$-calculus's increased expressive power. However, λ -calculus is also capable of expressing dynamically changing, arbitrary width, and depth nesting of functions; the Y fixed-point finder being a classic example. \$-calculus is also supposed to support true parallelism; the implications for effective computation are discussed below.

Costs are asserted as a central locus of the \$-calculus's strength. While it does not prescribe a base set of cost functions, crisp (i.e. algorithmic), probabilistic and fuzzy functions have all been developed. Users may also define their own cost functions. However, it is not clear how costs actually extend the \$-calculus's expressive power. Without costs, in particular mutating send, the \$-calculus is reminiscent of a higher-order process algebra, which as Eberbach (2000) notes, is no more powerful than a first-order system.

To add power to the \$-calculus, user-defined costs must be crafted in some formalism other than the \$-calculus itself, or built from base cost functions, where the other formalism or the base functions are themselves more powerful than anything expressible by either a Church–Turing system or cost-less \$-calculus. Either way, cost choice over any one bounded set of cost values and mutation over a bounded set of \$-expressions are both Church–Turing.

Choice over an infinite set of cost values seems deeply problematic, where even an approximation ultimately involves the expansion of all possible execution traces of the invoking program.

Finally, the operational semantics for \$-calculus is defined 'in a traditional way for process algebras' (Eberbach (2000), Section 3) using inference rules and a labelled transition system (LTS), where the LTS always looks for a least-cost action. However, inference rules and LTS are no more powerful than Church–Turing systems, so either the \$-calculus is a Church–Turing system or the semantics does not capture the full expressive power of the \$-calculus. In the latter case, it is not clear how the meaning of \$-calculus programs may actually be formalized or implemented.

8.7 Conclusions

We have reviewed a number of proposals for trans-Turing machines. In each case, we have seen that the machine does not live up to its

promise. Now in one sense this is not surprising, since many designs of conventional computers have faults in them initially. A computer is something very complicated and it is hard to get the design right initially. Bugs are uncovered in the initial designs as you try to fill in details. Some bugs survive until the first prototypes are built and tested.

Unless and until a hypercomputer is built, we have to reserve engineering judgement on whether it is possible for one to exist. The current proposals all seem to hit snags well before the prototype stage.

But suppose that a machine could be built that would solve some hypercomputational problem. What use would it be?

Suppose that it gave us answers about some uncomputable function. We would not be in a position to verify whether the answer was correct unless there was an independent way of verifying it by algorithmic means. We would need multiple different designs of hypercomputer, working in different ways so that they could check each other for consistency.

But in what sense would this hypercomputer be a computer?

It would not be a general-purpose computer. If it had general-purpose computing abilities, up to the level of self-emulation, then it would be possible to construct a self-referential problem analogous to the paradoxical problem that Turing constructed to demonstrate the halting problem.

Instead, it would be a special-purpose measuring device rather than a general-purpose computer. A Turing Machine can't answer the question 'How much do I weigh?' From the standpoint of TM's, this is an uncomputable number. But a set of bathroom scales can answer it, and we can check such scales for mutual consistency. In that sense, we already have non-algorithmic devices that give us numerical answers—we just don't call them hypercomputers.

The key property of general-purpose computers is that they are general purpose. We can use them to deterministically model any physical system, of which they are not themselves a part, to an arbitrary degree of accuracy. Their logical limits arise when we try to get them to model a part of the world that includes themselves.

Bibliography

- Adkins, C. J. (1983). *Equilibrium thermodynamics*. Cambridge University Press, Cambridge.
- Aharonov, D., van Dam, W., Kempe, J., Landau, Z., and Lloyd, S. and Regev, O. (2007). Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation. *SIAM Journal of Computing*, **37**(1), 166-194.
- Aristotle (1983). *Aristotle's Physics*. Clarendon Press, Oxford.
- Aristotle (1994–2000). *Metaphysics*. Internet Classics Archive by Daniel C. Stevenson, Web Atomics. <http://classics.mit.edu/Aristotle/metaphysics.mb.txt>.
- Aristotle, R. H. (1960). *Metaphysics*. University of Michigan Press, Ann Arbor, Michigan.
- Barrow, J. (2005). *The Infinite Book*. Vintage, London.
- Bashe, C. J., Johnson, L. R., Palmer, J. H., and Pugh, E. W. (1986). *IBM's Early Computers*. The MIT Press, Cambridge, MA.
- Beckenstein, J. (1981). Universal Upper Bound on the Entropy-to-Energy Ratio for Bounded Systems. *Physical Review D*, **23**, 287–298.
- Beggs, E. J. and Tucker, J. V. (2006). Embedding Infinitely Parallel Computation in Newtonian Kinematics. *Applied Mathematics and Computation*, **178**(1), 25–43.
- Bergman, G. D. (1955). A New Electronic Analogue Storage Device. In *Proceedings of the International Analogy Computation Meeting*, Brussels, pp. 92–95. Presses Académiques Européennes, Brussels.
- Blakey, E. (2011). Computational Complexity in Non-Turing Models of Computation. *Electronic Notes in Theoretical Computer Science*, **270**, 17–28.
- Blakey, E. (2008). *A Model-Independent Theory of Computational Complexity. Price: From Patience to Precision (and Beyond)*. Dissertation, Computing Laboratory, University of Oxford.
- Blakey, E. (2009). Factorizing RSA Keys, an Improved Analogue Solution. *New Generation Computing*, **27**(2), 159–176.
- Bohm, D. (1952a). A Suggested Interpretation of the Quantum Theory in Terms of 'Hidden' Variables. i. *Physical Review*, **85**(2), 166–179.
- Bohm, D. (1952b). A Suggested Interpretation of the Quantum Theory in Terms of 'Hidden' Variables. ii. *Physical Review*, **85**(2), 180–193.
- Boltzmann, L. (1995). *Lectures on Gas Theory*. Dover Publications, New York.

- Bournez, O. and Cosnard, M. (1995). On the Computational Power and Super-Turing Capabilities of Dynamical Systems. Technical Report 95-30, Ecole Normale Supérieure de Lyon.
- Bouso, R. (2002). The Holographic Principle. *Reviews of Modern Physics*, **74**(3), 825.
- Bracha, G. and Toueg, S. (1983). Asynchronous Consensus and Byzantine Protocol in a Faulty Environment. Technical Report TR-83-559, Computer Science Department, Cornell University, Ithaca, New York.
- Brooks, J. (2005). *Dreadnought Gunnery and the Battle of Jutland: The Question of Fire Control*. RoutledgeFalmer, London.
- Bunimovich, L. A. and Khlabytova, M. A. (2002). Lorentz Lattice Gases and Many-Dimensional Turing Machines. In *Collision-Based Computing* (ed. A. Adamatzky), pp. 443–468. Springer-Verlag, London.
- Burghartz, J. N., Jenkins, K. A., and Soyuer, M. (1996). Multilevel-Spiral Inductors using VLSI Interconnect Technology. *IEEE Electron Device Letters*, **17**(9), 428–430.
- Ceruzzi, P. E. (2003). *A History of Modern Computing*. The MIT Press, Cambridge, MA.
- Chaitin, G. (1987). *Information, Randomness and Incompleteness*. World Scientific, Singapore.
- Chaitin, G. (1999). Information and Randomness: A Survey of Algorithmic Information Theory. In *The Unknowable*. Springer-Verlag, Singapore.
- Chaitin, G. J. (1998). *The Limits of Mathematics*. Springer-Verlag, Singapore.
- Chomsky, N. (1956). Three Models for the Description of Language. *IRE Transactions on Information Theory*, **2**(3), 113–124.
- Church, A. (1936). An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, **58**, 345–363.
- Cockshott, P. (1982). *Orthogonal Persistence*. Ph.D. thesis, Department of Computer Science, University of Edinburgh.
- Cockshott, P., Atkinson, M., Chisholm, K., Bailey, P., and Morrison, R. (1984). POMS—A Persistent Object Management System. *Software Practice and Experience*, **14**(1), 49–71.
- Cockshott, P. and Michaelson, G. (2007). Are There New Models of Computation? Reply to Wegner and Eberbach. *The Computer Journal*, **50**(2), 232.
- Cohen-Tannoudji, C., Diu, B., Laloë, F., and Hemley, S. R. (1977). *Quantum Mechanics, Vol. 1*. Wiley Interscience, New York.
- Copeland, J. (2002). Accelerated Turing Machines. *Minds and Machines*, **12**, 281–301.
- Cray, S. R., Jr (1986). Immersion Cooled High Density Electronic Assembly. US Patent 4,590,538, 20 May.
- Crozier, W. D. and Hume, W. (1957). High-Velocity, Light-Gas Gun. *Journal of Applied Physics*, **28**(8), 892–894.
- da Costa, N. C. A. and Doria, F. A. (2009). How to Build a Hypercomputer. *Applied Mathematics and Computation*, **215**(4), 1361–1367.

- Davies, P. C. W. and Brown, J. R. (1993). *The Ghost in the Atom: A Discussion of the Mysteries of Quantum Physics*. Canto Series. Cambridge University Press, Cambridge.
- Davis, M. (1965). *The Undecidable*. Raven Press, Hewlett, NY.
- Davis, M. (1973). Hilbert's Tenth Problem is Unsolvable. *The American Mathematical Monthly*, **80**(3), 233–269.
- Dawson, T. (1909). Range Keeper For Guns. US Patent 941,626, 30 November.
- de Solla Price, D. (1959). An Ancient Greek Computer. *Scientific American*, **201**, 60–67.
- de Solla Price, D. (1974). Gears from the Greeks. The Antikythera Mechanism: A Calendar Computer from ca. 80 BC. *Transactions of the American Philosophical Society*, **64**(7), 1–70.
- d’Espagnat, B. (1976). *Conceptual Foundations of Quantum Mechanics*. Addison-Wesley, Reading, MA.
- d’Espagnat, B. (2003). *Veiled Reality: An Analysis of Present-Day Quantum Mechanical Concepts*. Frontiers in Physics. Westview Press, Boulder, CO.
- d’Espagnat, B. (2006). *On Physics and Philosophy*. Princeton University Press, Princeton, NJ.
- Deutsch, D. (1985). Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London A*, 97–117.
- Deutsch, D. (1989). Quantum Computational Networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, **425**(1868), 73–90.
- DiVincenzo, D. P. (2000). The Physical Implementation of Quantum Computation. *Arxiv preprint quant-ph/0002077*.
- Dumaresq, J. S. (1905). A Rate of Change of Range and Deflection Finder to Facilitate the Accurate Shooting of Guns. GB patent GBD190417719 19040815.
- Dunn, P. D. and Reay, D. A. (1973). The Heat Pipe. *Physics in Technology*, **4**, 187–201.
- Eberbach, E. (2000). Expressiveness of $\$$ -Calculus: What Matters? In *Advances in Soft Computing*, (eds M. Klopotek, M. Michalewicz, and S. T. Wierzchon), pp. 145–157. Physica-Verlag, Heidelberg.
- Eberbach, E. (2001). Process Algebra + Anytime Algorithms. In *Applicable Mathematics: Its Perspectives and Challenges* (ed. J. C. Misra), pp. 213–220. Narosa Publishing House, Mumbai.
- Eberbach, E. (2002). On Expressiveness of Evolutionary Computation: Is EC Algorithmic? In *Proceedings of the 2002 World Congress on Computational Intelligence WCCI’2002*, pp. 564–569.
- Einstein, A. (1920). *Relativity*. Methuen, London.
- Einstein, A. (1965). Concerning an Heuristic Point of View toward the Emission and Transformation of Light (English translation). *American Journal of Physics*, **33**(5), 367.

- Ekdahl, B. (1999). Interactive Computing Does Not Supersede Church's Thesis. In *Proceedings, Computer Science*, 17th International Conference, San Diego, pp. 261–265. Association of Management and the International Association of Management.
- Englund, R. K. (1996). The proto-Elamite Script. In *The World's Writing Systems* (eds P. T. Daniels and W. Bright), pp. 160–164. Oxford University Press, New York.
- Etesi, G. and Németi, I. (2002). Non-Turing Computations via Malament–Hogarth Space–Times. *International Journal of Theoretical Physics*, **41**(2), 341–370.
- Farhi, E., Goldstone, J., Gutmann, S., and Sipser, M. (2000). Quantum computation by adiabatic evolution. *Arxiv preprint quant-ph/0001106*. Arxiv, USA.
- Ferrari, A. (1998). JPVM: Network Parallel Computing in Java. *Concurrency Practice and Experience*, **10**(11–13), 985–992.
- Feynman, R. P. (1986). Quantum Mechanical Computers. *Foundations of Physics*, **16**(6), 507–531.
- Feynman, R. P., Leighton, R. B., and Sands, M. L. (1963). *The Feynman Lectures on Physics: Quantum Mechanics*. The Feynman Lectures on Physics. Pearson/Addison-Wesley, Reading, MA.
- Feynman, R. P. (1999). Simulating Physics with Computers. In *Feynman and Computation: Exploring the Limits of Computers* (ed. A. Hey), pp. 133–153. Perseus Books, Cambridge, MA.
- Fredkin, E. and Toffoli, T. (1982). Conservative Logic. *International Journal of Theoretical Physics*, **21**(3), 219–253.
- Freedman, M. H., Kitaev, A., Larsen, M. J., and Wang, Z. (2003). Topological Quantum Computation. *Bulletin of the American Mathematical Society*, **40**(1), 31–38.
- Freeth, T., Bitsakis, Y., Moussas, X., Seiradakis, J. H., Tselikas, A., Mangou, H., Zafeiropoulou, M., Hadland, R., Bate, D., Ramsey, A., et al. (2006). Decoding the Ancient Greek Astronomical Calculator Known as the Antikythera Mechanism. *Nature-London*, **444**(7119), 587.
- Gannon, P. (2006). *Colossus: Bletchley Park's Greatest Secret*. Atlantic Books, London.
- Garzon, M. (1995). *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin.
- Gea-Banacloche, J. (2002). Minimum Energy Requirements for Quantum Computation. *Physical Review Letters*, **89**(21), 217901.
- Gea-Banacloche, J. and Kish, L. B. (2003). Comparison of Energy Requirements for Classical and Quantum Information Processing. *Fluctuation and Noise Letters*, **3**, C3–C7.
- Geist, A. (1994). *PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing*. The MIT Press.
- George, D. F. J. (2005). *Reconfigurable Cellular Automata Computing for Complex Systems on the SPACE Machine*. Ph.D. thesis, University of South Australia.

- Gödel, K. (1962). *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. Oliver and Boyd, Edinburgh.
- Goldin, D. Q., Smolka, S. A., Attie, P. C., and Sonderegger, E. L. (2004). Turing Machines, Transition Systems, and Interaction. *Information and Computation*, **194**(2), 101–128.
- Gordon, P. (2004). Numerical Cognition without Words: Evidence from Amazonia. *Science*, **306**(5695), 496.
- Gray, P. A. and Sunderam, V. S. (1997). IceT: Distributed Computing and Java. *Concurrency—Practice and Experience*, **9**(11), 1161–1167.
- Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, New York, pp. 212–219. ACM.
- Halmos, P. R. (1960). *Naive Set Theory*. Van Nostrand, Princeton, NJ.
- Hamkins, J. D. (2002). Infinite Time Turing Machines. *Minds and Machines*, **12**(4), 521–539.
- Hamming, R. W. (1980). The Unreasonable Effectiveness of Mathematics. *American Mathematical Monthly*, 81–90.
- Hartree, D. R. (1938). The Mechanical Integration of Differential Equations. *The Mathematical Gazette*, **22**(251), 342–364.
- Hawking, S. W. (1974). Black Hole Explosions. *Nature*, **248**(5443), 30–31.
- Heyting, A. (1974). *Mathematische Grundlagenforschung, Intuitionismus, Beweistheorie*. (German) Reprint. Springer-Verlag, Berlin.
- Hodges, A. (1983). *Alan Turing: The Enigma*. Touchstone edition. Simon and Schuster, New York.
- Hodges, W. (1977). *Logic*. A Pelican original. Penguin Books, Harmondsworth.
- Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, MA.
- Hunter, G. (1971). *Metalogic: An Introduction to the Metatheory of Standard Logic*. Macmillan, London.
- Ifrah, G. (1995). *Histoire universelle des chiffres: l'intelligence des hommes racontée par les nombres et le calcul*. R. Laffont, Paris.
- Isham, C. J. (1995). *Lectures on quantum theory: mathematical and structural foundations*. Imperial College Press, London.
- Joos, E. and Zeh, H. D. (1985). The Emergence of Classical Properties through Interaction with the Environment. *Zeitschrift für Physik B: Condensed Matter*, **59**, 223–243. 10.1007/BF01725541.
- Kieu, T. D. (2003). Quantum Algorithm for Hilbert's Tenth Problem. *International Journal of Theoretical Physics*, **42**, 1461–1478.
- Kim, Y.-H., Yu, R., Kulik, S. P., Shih, Y., and Scully, M. O. (2000). Delayed 'Choice' Quantum Eraser. *Physical Review Letters*, **84**(1), 1–5.
- Kish, L. B. (2004). Moore's Law and the Energy Requirement of Computing versus Performance. *IEE Proceedings on Circuits, Devices, and Systems*, **151**(2), 190–194.

- Kistermann, F. W. (1998). Blaise Pascal's Adding Machine: New Findings and Conclusions. *IEEE Annals of the History of Computing*, **20**(1), 69–76.
- Kleene, S. (1952). *Introduction to Metamathematics*. Van Nostrand, Princeton, NJ.
- Kneebone, G. (1963). *Mathematical Logic and the Foundations of Mathematics*. Van Nostrand, London.
- Knight, T. F., Jr and Younis, S. G. (1994). Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic. Technical report. Massachusetts Institute of Technology, Cambridge, MA.
- Knuth, D. E. (1976). Big Omicron and Big Omega and Big Theta. *SIGACT News*, **8**(2), 18–24.
- Koiran, P. and Moore, C. (1999). Closed-Form Analytic Maps in One and Two Dimensions can Simulate Turing Machines. *Theoretical Computer Science*, **210**(1), 217–223.
- Kornai, A. (2003). Explicit Finitism. *International Journal of Theoretical Physics*, **42**(2), 301–307.
- Lakoff, G. and Nunez, R. (2001). *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. Basic Books, New York.
- Landauer, R. (1961). Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, **5**, 183–191.
- Landauer, R. (1991). Information is Physical. *Physics Today*, May, 23–29.
- Landauer, R. (1996). The Physical Nature of Information. *Physics Letters A*, **217**(4–5), 188–193.
- Landauer, R. (2002). Information is Inevitably Physical. In *Feynmann and Computing* (ed. A. Hey). Westview Press, Oxford.
- Landin, P. J. (1964). The Mechanical Evaluation of Expressions. *Computer Journal*, **6**(4), 308–320.
- Lavington, S. (1975). *A History of Manchester Computers*. NCC Publications, Manchester.
- Lavington, S. H. (1980). *Early British Computers: The Story of Vintage Computers and the People Who Built Them*. Manchester University Press, Manchester.
- Lavington, S. H. (1978). The Manchester Mark I and Atlas: A Historical Perspective. *Communications of the ACM*, **21**(1), 4–12.
- Lloyd, S. (2002). Computational Capacity of the Universe. *Physical Review Letters*, **88**(23), 237901.
- MacKay, D. and Fisher, M. (1962). *Analogue Computing at Ultra High Speed*. Chapman and Hall, London.
- Mandler, G. and Shebo, B. J. (1982). Subitizing: An Analysis of its Component Processes. *Journal of Experimental Psychology: General*, **111**(1), 1–22.
- Marlow, S. (ed.) (2010). *Haskell 2010 Language Report*. <http://www.haskell.org/onlinereport/haskell2010/>.
- McNamara, B. (2010). *F# Language Reference*. <http://msdn.microsoft.com/en-us/library/dd233181.aspx>.

- Menicucci, N. C. and Caves, C. M. (2002). Local Realistic Model for the Dynamics of Bulk-Ensemble NMR Information Processing. *Physical Review Letters*, **88**(16), 167901.
- Menninger, K. (1992). *Number Words and Number Symbols: A Cultural History of Numbers*. Dover Publications, New York.
- Mermin, N. D. (2007). *Quantum Computer Science: An Introduction*. Cambridge University Press, Cambridge.
- Mermin, N. D. (2009). Whats Bad About This Habit. *Physics Today*, **62**(May), 8.
- Michaelson, G. (1993). *Interpreter Prototypes from Formal Language Definitions*. Ph.D. thesis, Department of Computing and Electrical Engineering, Heriot-Watt University.
- Milne, G., Cockshott, P., McCaskill, G., and Barrie, P. (1993). Realising Massively Concurrent Systems on the SPACE Machine. In *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, 5–7 April 1993, Napa, CA (eds D. A. Buell and K. L. Pocek), pp. 26–32. IEEE Computer Society Press, Los Alamitos, CA.
- Milner, R. (1992). The Polyadic π -Calculus: A Tutorial. In *Logic and Algebra of Specification* (ed. F. L. Bauer and W. Brauer). Springer-Verlag, Berlin.
- Milner, R. (1993). Elements of Interaction: Turing Award Lecture. *Communications of the ACM*, **36**(1).
- Minsky, M. (1972). *Computation: Finite and Infinite Machines*. Open University Press, Milton Keynes.
- Mizel, A., Lidar, D. A. and Mitchell, M. (2007). Simple proof of equivalence between adiabatic quantum computation and the circuit model. *Physical review letters*, **99**(7), 70502.
- Moore, E. F. (1956). Gedanken-Experiments on Sequential Machines. *Automata Studies*, **34**, 129–153.
- Mott, N. (1929). The Wave Mechanics of Alpha-Ray Tracks. *Proceedings of the Royal Society*, **A126**, 79–84.
- Nagel, E. and Newman, J. R. (1959). *Gödel's Proof*. Routledge and Kegan Paul, London.
- Neugebauer, O. (1955). Apollonius' Planetary Theory. *Communications on Pure and Applied Mathematics*, **8**(4), 641–648.
- Nidditch, P. H. (1962). *Propositional Calculus*. Routledge and Kegan Paul, London.
- Nissen, H. J., Damerow, P., and Englund, R. K. (1993). *Archaic Book-keeping: Early Writing and Techniques of Economic Administration in the Ancient Near East*, pp. 11–12. University of Chicago Press, Chicago, IL.
- Omnes, R. (1994). *The Interpretation of Quantum Mechanics*. Princeton University Press, Princeton, NJ.
- Peter, R. (1967). *Recursive Functions*. Academic Press, New York.
- Polcari, M. R. (2005). Collaboration: The Semiconductor Industry's Path to Survival and Growth. In *AIP Conference Proceedings*, Volume 788, p. 3. IOP Institute of Physics Publishing, Bristol.
- Post, E. L. (1936). Finite Combinatory Processes. Formulation 1. *Journal of Symbolic Logic*, **1**, 103–105.

- Power, R. J. D. and Longuet-Higgins, H. C. (1978). Learning to Count: A Computational Model of Language Acquisition. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, **200**(1141), 391.
- Pratt, V. (1987). *Thinking Machines: The Evolution of Artificial Intelligence*. Blackwell, Oxford.
- Primas, H. (1998). Emergence in Exact Natural Sciences. *Acta Polytechnica Scandinavica*, **91**(1998), 5–83.
- Pugh, E. W., Johnson, L. R., and Palmer, J. H. (1991). *IBM's 360 and Early 370 Systems*. The MIT Press, Cambridge, MA.
- Quinn, T. J. (1999). International Report: Practical Realization of the Definition of the Metre (1997). *Metrologia*, **36**(3), 211–244.
- Rashleigh, S. C. and Marshall, R. A. (1978). Electromagnetic Acceleration of Macroparticles to High Velocities. *Journal of Applied Physics*, **49**(4), 2540–2542.
- Rucker, R. (2004). *Infinity and the Mind: the Science and Philosophy of Infinity*. Princeton University Press, Princeton, NJ.
- Russell, G., Cowie, A., McInnes, J., Bate, M., and Milne, G. (1994). Simulating Vehicular Traffic Flows Using the Circal System. Technical report RR-94-157 (formally aiscia-1-94), University of Strathclyde.
- Sangiori, D. and Walker, D. (2001). *The π -Calculus*. Cambridge University Press, Cambridge.
- Sazonov, V. Yu. (1995). On Feasible Numbers. In *Logic and Computational Complexity* (ed. D. Leivant), pp. 30–51. Lecture Notes in Computer Science, Volume 960. Springer-Verlag, Berlin.
- Scardigli, F. (1999). Generalised Uncertainty Principle in Quantum Gravity from Micro-Black Hole Gedanken Experiment. *Physics Letters B*, **452**, 39–44.
- Schenck, T. O. E. C. (1883). Manufacture of Gunpowder. US Patent 273,209, 27 February.
- Sears, F. W. and Salinger, G. L. (1975). *Thermodynamics, kinetic theory, and statistical thermodynamics*. Addison-Wesley, Boston, MA.
- Shannon, C. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, **27**, 379–423 and 623–656.
- Shaw, P., Cockshott, P., and Barrie, P. (1996). Implementation of Lattice Gases Using FPGAs. *Journal of VLSI Signal Processing*, 1251–1256.
- Shor, P. W. (1999). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, **41**(2), 303–332.
- Shor, P. W. (2004). Progress in Quantum Algorithms. *Quantum Information Processing*, **3**(1), 5–13.
- Shor, P. W. (1995). Scheme for Reducing Decoherence in Quantum Computer Memory. *Physical Review A*, **52**(4), R2493–R2496.
- Smith, W. D. (2006a). Church's Thesis Meets the N -Body Problem. *Applied Mathematics and Computation*, **178**(1), 154–183.
- Smith, W. D. (2006b). Three Counterexamples Refuting Kieu's Plan for 'Quantum Adiabatic Hypercomputation' and Some Uncomputable Quantum Mechanical Tasks. *Journal of Applied Mathematics and Computation*, **187**(1), 184–193.

- Spinellis, D. (2008). The Antikythera Mechanism: A Computer Science Perspective. *Computer*, **41**(5), 22–27.
- Stapp, H. P. (1972). The Copenhagen Interpretation. *American Journal of Physics*, **40**, 1098–1116.
- Steane, A. (1996). Multiple-Particle Interference and Quantum Error Correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, **452**(1954), 2551–2577.
- Suzuki, S. and Matsuzawa, T. (1997). Choice between Two Discrimination Tasks in Chimpanzees (*Pan troglodytes*). *Japanese Psychological Research*, **39**(3), 226–235.
- Swinbourne, A. (1875). *Picture Logic or the Grave Made Gay*. Longmans, Green, London.
- Temple, R. K. G. (2000). *The Crystal Sun: Rediscovering a Lost Technology of the Ancient World*. Century, London.
- Thomson, W. (1876a). Mechanical Integration of the Linear Differential Equations of the Second Order with Variable Coefficients. *Proceedings of the Royal Society of London*, **24**, 269–271.
- Thomson, W. (1876b). On an Instrument for Calculating $(\int \varphi(x) \psi(x) dx)$, the Integral of the Product of Two Given Functions. *Proceedings of the Royal Society of London*, **24**, 266–268.
- Tiles, M. (1989). *The Philosophy of Set Theory: An Introduction to Cantor's Paradise*. Blackwell, Oxford.
- Tripp, J. L., Mortveit, H. S., Hansson, A. A., and Gokhale, M. (2005). Metropolitan Road Traffic Simulation on FPGAs. In *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2005 (FCCM 2005)*, 18–20 April 2005, pp. 117–126.
- Turing, A. (1937). On Computable Numbers, With an Application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, **42**, 230–265.
- Turing, A. (1939). Systems of Logic Based on Ordinals. *Proceedings of the London Mathematical Society*, Series 2, **45**, 161–228.
- Turing, A. (1950). Computing Machinery and Intelligence. *Mind* (49), 433–460.
- Turing, A. (2004). Lecture on the Automatic Computing Engine, 1947. In *The Essential Turing* (ed. B. J. Copeland). Oxford University Press, Oxford.
- Turner, D. and Pierce, B. (2000). Pict: A Programming Language Based on the pi-Calculus. In *Proof, Language and Interaction: Essays in Honour of Robin Milner* (eds G. Plotkin, C. Stirling, and M. Tofte), pp. 455–494. The MIT Press, Cambridge, MA.
- von Neumann, J. (1945, 30). *First Draft of a Report on the EDVAC, Contract No. W-670-ORD-4926, Between the United States Army Ordnance Department and the University of Pennsylvania Moore School of Electrical Engineering*. University of Pennsylvania.
- von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ.

- Walker, D. and Dongarra, J. (1996). MPI: A Standard Message Passing Interface. *Supercomputer*, **12**(1), 56–68.
- Wass, C. A. A. (1955). *Introduction to Electronic Analogue Computers*. Pergamon Press, London.
- Wegner, P. (1997a). Interactive Software Technology. In *The Computer Science and Engineering Handbook* (ed. A. B. Tucker, Jr). CRC Press, Boca Raton, FL.
- Wegner, P. (1997b). Why Interaction is More Powerful than Algorithms. *Communications of the ACM*, **40**(5), 80–91.
- Wegner, P. and Eberbach, E. (2004). New Models of Computation. *Computer Journal*, **47**, 4–9.
- Wigner, E. (1960). The Unreasonable Effectiveness of Mathematics in the Natural Sciences. *Communications in Pure and Applied Mathematics*, **13**(1), 1–14.
- Wilczek, F. (1982). Magnetic Flux, Angular Momentum, and Statistics. *Physical Review Letters*, **48**(17), 1144–1146.
- Wilder, R. L. (1978). *Evolution of Mathematical Concepts*. Open University Press, Milton Keynes.
- Williams, F. C. (1948). A Cathode Ray Tube Digit Store. *Proceedings of the Royal Society of London*, **195A**, 279–284.
- Wisdom, J. O. (1953). Berkeley’s Criticism of the Infinitesimal. *The British Journal for the Philosophy of Science*, **4**(13), 22–25.
- Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media, Inc., Champaign, IL.
- Yarmolenko, V., Cockshott, P., Borland, E., and Mackenzie, L. (2004). $\mathcal{J}\pi$ INTERFACE: A Java Implementation of the π -Calculus for Grid Computing. In *Proceedings of the Middleware Grid Conference*, October.
- Younis, S. G. (1994). *Asymptotically Zero Energy Computing Split-Level Charge Recovery Logic*. Ph.D. thesis, Ph.D. Dissertation, MIT, Cambridge, MA.
- Zalka, Ch. (1999). Grover’s Quantum Searching Algorithm is Optimal. *Physical Review A*, **60**(4), 2746–2751.

Index

- * (multiplication), 56
- + (addition), 56
- − (subtraction), 57
- / (division), 57
- $\langle\langle e \rangle\rangle$ (Gödel number), 62
- SUCC* (successor), 56
- \Rightarrow (implication), 48
- \Rightarrow_β (β reduction), 76
- \aleph_0 (aleph null), 65
- \aleph_1 (aleph one), 66
- β (beta) reduction, 76
- \cap (intersection), 52
- \cup (union), 52
- \exists (exists), 53
- \forall (all), 53
- \in (member), 51
- λ (lambda) calculus, 75
- λ calculus, 81
- λ expression, 75
- \wedge (and), 48
- \vee (or), 48
- \neg (not), 48
- \subset (subset), 52
- $\{\}$ (empty set), 51
- mod* (remainder), 57
- \setminus (difference), 52

- Fredkin
 - and Tofoli, 119

- abacus, 25, 44
- abakion, 24, 25
- absolute, 97
 - temperature, 103
- absolute infinity, 179
- abstract, 6, 10, 12, 24, 85, 191
 - maths, 191
- abstract machine, 81
- abstraction, 11, 12, 84
- accelerating, 187, 188, 194, 195
 - Turing, 188
- acceleration, 127, 188, 194
- accuracy, 30, 35, 41, 83, 167, 168,
 - 170, 171, 181, 184, 190,
 - 197, 198
- accurate, 10, 24, 32, 35, 36, 42, 83,
 - 182, 198
- Ackerman's function, 185

- actions, 17, 23
- active, 83, 196, 198
 - gate, 116
- actualized infinity, 178
- add, 30, 35, 43–45, 86, 198, 200
- adding, 23–25, 27, 42, 83
- addition, 23, 24, 30, 35, 37, 39, 43, 45,
 - 56, 84, 196
- address, 20, 41, 86, 88
- adiabatic, 101, 121, 162
 - chip, 122
 - circuit, 122
 - system, 101
- adiabatic chip, 122
- adjoint, 141
- advanced, 12, 23, 28, 34, 41, 180
- aerodynamic, 83
- aft, 37, 38
- ahead, 126
- aleph null, 65
- aleph one, 66
- algebra, 10, 34
 - associative, 143
- algorithm, 45, 68, 172, 192, 198
- algorithmic, 191, 201, 215
- algorithmically, 193
- algorithms, 136
- aliasing, 200
- aligned, 37, 38
- alignment, 41
- amber, 18, 19
- AMD, 92
- AMD K10, 94
- amplified, 118
- amplifier, 84
- amplitude, 157
- amplitudes, 157
- analogous, 34, 84, 195, 215
- analogue, 9, 34, 35, 39, 41, 83,
 - 167–171, 187, 190, 196,
 - 198, 201
 - computer, 84, 195
 - device, 84
 - machine, 84
 - system, 196
- analogy, 34, 84, 187
- analysis, 7, 17
- Analytical Engine, 82

- ancient, 24, 28, 32, 34
 - Greek, 30
- ancillary qubit, 161
- and, 48
 - gate, 88
 - truth table, 49
- angle, 38, 180, 197
 - between, 38
- angular, 30, 32, 35, 192
- angular momentum, 137, 138, 143
 - z -component, 144
- animals, 11, 13
- anti-realism, 163
- Antikythera, 32, 34–36, 42
 - device, 33, 35
- antor diagonalization, 185
- aperture, 197, 198
- Apollonius, 32, 33
- Apollonius'
 - model, 33
- apparatus, 84, 191, 197–199,
 - 201
- application, 7, 15, 127, 169, 192
- apply, 85
- applying, 194
- appropriately, 119
- approximate, 32, 93, 167,
 - 170, 171
- approximation, 32, 33, 106, 168
- arbitrarily, 170, 181, 192–194
 - high, 193
 - small, 192, 194
- arbitrary, 31, 105, 167, 182, 192,
 - 195, 215
- architecture, 26, 123
- Aristotle, 177
- arithmetic, 23, 35, 37, 42, 45, 56,
 - 125, 198
 - human, 68
 - Peano, 55
- array, 89, 200
- assertion, 105
- assigned, 25, 86, 98
- associate, 14
- associated, 9, 98, 171
- associative, 143
- astronomers, 33, 34
- astronomy, 32, 42

- Athlon, 92
- atoms, 169, 170, 192
- automata, 123, 196
- automatic, 7, 42, 86
- automaton, 17, 20–22, 127, 196
- average, 6, 87
- Avogadro, 106
- axiomatic, 193
- axioms, 167
 - Peano arithmetic, 57
- axis, 32, 33, 37, 40, 90, 183, 197
- axiom, 50

- Babbage, C., 82
- ball, 119, 194, 195
 - logic, 119
- ballistic
 - engineering, 194
- barrel, 45, 195
- base case, 55
 - inductive proof, 57
 - recursive definition, 56
- basis, 143, 144, 146, 147, 165
 - change, 146
 - computational, 143, 152
 - standard, 145, 153
- basis states, 160, 161
- beads, 24, 25, 44
- beams, 199
- bear, 34, 40, 85, 198
- bearing, 35, 37
- bee, 9
- Beggs, 193–195
- behaves, 105
- behaviour, 9, 19, 20, 96, 97
- Bell Laboratories, 112
- below, 15, 21, 25, 168, 172
- Bergman, 83
- beta barium borate, 131
- big O notation, 175
- billiard, 119, 195
 - ball, 119, 195
 - ball logic, 119
- binary, 184, 194, 196
 - switching, 91
- bits, 88, 170, 180, 184, 198
- black, 171, 183, 184, 187, 188, 190
 - hole, 183, 190
- Blakey, 198, 200
- body, of λ function, 75
- Bohm, 135, 164
- Bohr, 128, 133, 134, 163
- Bohrian view, 163
- Boltzmann, 96, 108, 111, 114, 115
 - Boltzmann's
 - relation, 110
 - constant, 108, 115
 - boolean, 85
 - boolean algebra, 10
 - Born Rule, 141, 157
 - bound, 116, 169
 - bound variable, of λ function, 75
 - bounded, 85
 - box, 8, 109
 - bra vector, 141
 - braid, 162
 - brain, 10, 17, 20, 165
 - brightness, 200
 - British, 86
 - Brouwer, L. E. J., 179
 - Busicom, 92

 - Calabria, 35
 - calculate, 39, 42
 - calculated, 25, 34
 - calculating, 26
 - calculation, 6, 9, 15, 26–27, 30, 35, 36, 41, 42, 84, 192, 193, 198
 - calculator, 25, 43, 45, 46, 90
 - calculi, 25
 - calculus, 9, 10, 191, 193
 - calendar, 28
 - Callipic, 30, 35
 - cannon, 194, 195
 - cannonball, 194, 195
 - canonical, 106
 - Cantor, 185
 - diagonalisation, 66
 - Cantor diagonalization, 178
 - Cantor, G., 52, 64, 179
 - capacitance, 93
 - of gates, 93
 - of the gate, 94
 - capacitor, 85
 - capacity, 17, 184
 - car, 126
 - cardinal number, 65
 - cardinality
 - of reals, 67
 - cartesian, 37, 38
 - cathode, 87
 - cause, 88, 184
 - caused, 107, 199, 200
 - causing, 188
 - cell, 88
 - current, 69
 - of Turing Machine tape, 69
 - Central processing Unit, 80
 - Chaitin, G., 79
 - change
 - of state, 99
 - of the system, 101
 - charge, 10, 43, 85, 121, 195
 - on, 87
 - on the gate, 92
 - Charge Recovery Logic, 121
 - charged, 121
 - charging, 93
 - and discharging, 120
 - cheapening, 90
 - cheaper, 89
 - children, 24, 27
 - chip, 20, 88, 114, 122
 - Chomsky, 17, 20
 - class, 17
 - Church, 187
 - Church–Turing thesis, 176
 - circle, 6, 14, 33, 197–199
 - circle free, 72
 - circles, 7
 - circuit, 17, 90, 172
 - board, 123
 - circular, 30, 32, 33, 37, 42, 197
 - circumference, 6
 - class, 10, 17, 20, 117, 196
 - classes, 17
 - of grammars, 17
 - classical, 95, 167, 169, 171, 172, 192, 193, 198
 - thermodynamics, 96
 - classical words
 - appearance of, 149
 - classified, 17, 20
 - clause, 18
 - Clausius, 102, 109
 - entropy, 111
 - Clausius'
 - Theorem, 104
 - theorem, 104
 - clay, 12, 13, 15
 - clock, 20, 28, 34, 35, 39, 40, 42, 85, 172, 189, 194
 - speed, 115
 - cycle, 93, 189
 - dial, 41
 - speed, 94, 116
 - ticks, 189
 - cloud chamber, 134
 - CMOS, 91–93, 114, 115, 121, 172
 - gate, 121
 - chip, 93
 - circuit, 120
 - gate, 93, 115
 - CNOT, 151
 - gate, 151, 161
 - matrix, 151
 - operator, 151
 - code, 86, 192

- coded, 86
- cog, 46
- coherence time, 159
- coins, 24
- cold, 95
 - reservoir, 102
- colder, 101
- collection, 93, 194
 - of messages, 113
- collections, 24
- collision, 119
- colours, 19
- combination, 19, 21, 89
- combine, 91
- combined, 37, 41
- commercial, 42, 46
- commutative, 143
- compact, 41, 46
- compilation, 81
- complete, 59
- completeness
 - topological, 139
- Completeness Hypothesis, 164, 166
 - Weak, 134
- complex
 - conjugate, 140
 - numbers, 139, 140
- complexity, 116, 158, 172, 174, 201
 - constant time, 175
 - exponential time, 175
 - linear time, 175
 - non-deterministic polynomial time, 175
 - polynomial time, 175
- complicated, 28, 215
- component, 17, 22, 33, 37, 39, 99, 169, 188, 200
 - of angular momentum, 143
- composed, 104
- composite, 104
 - system, 110
- computability, 78, 187
- computable, 183, 184, 187, 192
 - real, 184
- computation, 6, 7, 14, 27, 31, 41, 59, 118, 170, 187, 188
- computational, 14, 15, 17, 35, 184, 194, 196
 - aids, 15
 - basis, 150
 - power, 10
- compute, 8, 26, 34, 118, 184, 188, 189, 191, 192
- computed, 183, 188
- computer, 7–10, 15, 17–19, 28, 34, 39, 41–43, 80, 83, 171, 184, 187–189, 191, 193–195, 215
 - science, 9, 184
 - technology, 116
- computing, 6, 7, 9–11, 14, 23, 26, 28, 32, 34, 37, 39, 83, 171, 172, 184, 188, 190, 191, 193, 215
 - machine, 123
 - with real numbers, 84
 - work, 122
- conception
 - of entropy, 111
- conceptual, 16
- conclusion, 50
- configuration, 100
- configurative, 100
 - work, 104
- confusion, 197
- connection, 8–10, 93
 - to ground, 93, 121
- consciousness, 165
- conservative
 - logic, 117
- conservative logic, 117
- conserved, 99
- consistent, 59
- constant, 15, 39, 83, 169, 172, 182, 183, 194
 - of proportionality, 108
- constant time complexity, 175
- constrain, 168, 197, 200
- constraint, 101, 171, 188, 197, 200
- constraints, 97, 191, 193, 194
- constructive methods, 179
- constructivism, 179
- consumption, 171, 172
 - per square centimetre, 94
- context, 17, 20–22
 - sensitive, 22
- continuous, 34, 35, 84, 167, 192
- continuously, 85, 167, 193
- continuum, 192
- contradiction, 59, 167
 - proof by, 179
- contradictions, 9
- control, 7, 17, 20, 23, 42
 - computers, 42, 83
- controlled NOT, 151
- cooling, 94, 172
- coordinates, 38, 97, 195, 199
- Copeland, 187, 188
- Copenhagen interpretation, 163, 164, 166
- Copernican revolution, 164
- Copernicus, 166
- core, 87
 - memory, 88
- cores, 89
- corpuscular, 96
- Costa, 191
- count, 11, 13, 15, 17, 20, 22, 23, 24, 27, 45, 90
- countable, 65
- counted, 20, 26
- counters, 23–26
- counting, 12, 14, 16–20, 22–25, 27, 108, 181
 - aloud, 20, 23, 27
 - as, 22
- CPU, *see* Central processing Unit, 123
- Cray, 124
- current, 18, 32, 41, 91, 171, 172, 215
 - flow, 93
 - state, 69, 123
 - symbol, 69
- currently, 114, 201
- curve, 199
- cycle, 20, 29, 34, 35, 85, 189
 - from, 102
 - time, 87
- cycles, 19
- cyclical, 87
- d’Espagnat, 136, 163–165
- D-time, 162
- Da Costa, 190, 191
- Davis, 191
- Davisson–Germer experiment, 133
- de Broglie, 133, 135, 164
- decidability, 176
- decidable, 59, 176
- decide, 112
- decimal, 41, 84, 183, 187, 188
 - fraction, 65
- decision procedure, 176
- declarative language, 57
- decode, 88
 - logic, 89
 - tree, 89
- decoded, 89
- decoding, 126
- decoherence, 148, 149, 152, 158, 160, 162
 - time, 158, 160
- deduce, 193
- deductive proof, 50
- deep, 29, 94
- define, 18, 21, 103, 109, 168, 184, 191
 - a
 - different state, 109
 - random bit, 184
 - real, 191
- defined, 19, 20, 98, 109, 113, 183

- defining, 103
- definite, 97, 171, 180, 188
- definition, 86, 184
 - of heat, 101
 - recursive, 56
 - expression, 62
- deflection, 36, 39
- degenerate, 107
- degree, 43, 84, 85, 167, 182, 183, 189, 191, 192, 215
 - of accuracy, 184
 - of freedom, 97, 120
 - of precision, 167
- delay lines, 86
- delayed choice quantum eraser, 132
- delayed choice quantum eraser experiment, 132
- demultiplexer, 89
- denumerable, 65
- depletion, 92
 - zone, 92
- detector, 199
- determination, 106, 192
- determine, 12, 26, 169, 191, 193, 201
- determined, 107, 168, 192
- determining, 112
- deterministic, 120, 136, 137, 145, 152, 167, 174
- Deutsch, 135, 162, 165, 166
- deviation, 32, 85
- device, 10, 20, 25, 32, 34–37, 39, 42, 84, 168, 171, 191, 215
- DeWitt, 165
- diagonalisation, 66
- diagonalization
 - Cantor, 178, 185
- dial, 28, 29, 35, 40
- diameter, 31
- difference, 52
- differential
 - analyser, 40, 84
 - equations, 193
 - form, 104
 - gearing, 32
- diffracted, 200
- diffraction, 129–131, 197, 200
- digit, 43, 84, 184, 188
- digital, 10, 15, 34, 35, 41–44, 170, 171, 198, 200
 - computer, 15, 83
 - computers, 10, 85
 - machine, 83
- digits, 22, 30, 35, 41, 84, 192, 197
 - encode, 113
 - leading, 35
- digitword, 21
- dimension, 90, 167, 197, 198, 201
 - of tensor product space, 147
- dimensional, 169, 170, 200
- Diophantine equation, 158
- Dirac, 140
 - notation, 140, 141, 152
- direction
 - of Turing Machine tape, 69
- disc, 25
- discharge, 93, 121
- discharged, 121
- discharging, 93
- disk, 38, 42, 122
- display, 30, 198
- dissipate, 91
- dissipated, 100, 172
- dissipation, 91
- dissipative, 100, 110, 188
 - work, 100
- distance, 32, 88, 168, 169, 181, 183, 194
 - between, 94
- divided, 84, 104, 169
 - into, 123
- DiVincenzo, 158
- divisible, 181, 183
- division, 57
- doping, 91
- double
 - slit experiment, 130, 132, 133
 - well potential, 159
- double-slit experiment, 130
- drachma, 24, 25
- drain, 91
- DRAM, 89
- draw, 21
- drive, 96
- dual, 140, 141
- Dumaresq, 37–39, 41, 42, 152
- dynamical variables, 137, 139

- earth, 32, 33, 180
- earthquake, 112
- East, 37
- eccentric, 32, 33
 - model, 33
- eclipse, 30, 34
- economic, 12, 122
- economy, 89, 171
- effective, 6, 7, 114
 - unreasonably, 6
- effective calculability, 78
- effective procedure, 68, 174
- effectively, 108
- effectiveness, 7, 11
 - of maths, 11
- effects, 32, 84
- efficiency, 103, 112

- efficient, 113
- effort, 16, 122
- eigenspace, 141, 152
- eigenstate, 165
 - normalised, 142
- eigenvalue, 141, 152
 - degenerate, 141
 - of unitary operator, 144
- eigenvector, 141, 144, 152
- Einstein, 7, 129, 133, 148, 192
- electric, 87, 192
- electrical, 17, 20, 41, 100
 - energy, 102
- electro-magnetic, 192
- electromagnetic, 90, 195
 - theory, 129
- electron, 85, 133, 134, 137, 138, 145, 159, 192
 - magnetic properties, 145
 - orbital, 133
 - spin, 139, 143, 145, 160
- electronic, 17, 46, 83
 - calculator, 90
 - computer, 122
- ellipse, 33
- elliptical, 32
- elongated, 125
- empty set, 51
- emulate, 118, 196
- encode, 112
 - in binary, 125
- encoded, 112, 190
- encoding, 113, 184, 190, 191, 193
- enemy, 38, 39, 42
 - course, 38
- energies, 99
- energy, 96, 111, 115, 136, 142, 172, 182, 184, 189, 192
 - kinetic, 136
 - level, 107, 159
 - levels, 110
 - of the particles, 100
 - of the system, 99
 - potential, 136
- engine, 86
- engineering, 6, 7, 112, 191, 194, 215
- English, 20, 21, 23
- Enlightenment, 164, 166
- ensemble, 159
- entangled, 131, 132, 147, 149, 159
 - photons, 148
 - state, 149
 - system, 147–149
- entanglement, 146–152, 154, 158, 160, 165
 - state, 147
- entanglement

- state, 147
- entropy, 95, 104, 105, 110, 111, 184
 - of the environment, 115
 - reduction, 115
 - to decrease, 105
- entropy , 184
- enumerable, 65, 188
- enumeration, 12
- environment, 96, 148, 149, 158, 167
- epicycle, 32, 33
- epistemology, 163, 164
- EPR paradox, 148
- equal, 32, 34, 103, 171, 180, 181
- equality, 103
- equally, 97
 - likely, 112
 - probable, 108
- equation, 31, 34, 107, 167, 168, 183, 193
 - of state, 97
- equations, 7, 10
- equilibrium, 96, 99, 110, 184
 - state, 109
 - state, 97
- equipment, 37, 84, 198
- error, 23, 28, 34, 41, 83, 84, 158, 160, 169, 171, 172, 192, 193, 195–197
 - bit flip, 160, 161
 - correcting code, 160
 - correction, 162
 - detection and correction, 160
 - due, 197
 - due to uncertainty, 84
 - phase, 161
- Esti, 188
- evaluate, 191
- evaluation, 198
- event, 120, 184, 188, 189
 - horizon, 188, 189
- events, 128
- Everett, 164
- exact, 42, 107, 171, 180
- exchanged
 - with the surroundings, 104
- excited state, 159
- excluded middle
 - law of, 179
- execute, 102, 188
 - a cycle, 102
- execution, 198
- Exeligemos, 30
- existence proof, 179
- existential quantification, 53
- exitonic state, 160
- expand, 18, 100
- expansion, 183, 196
- experimental, 6, 95
- explain, 7, 33
- explained, 86
- explanation, 119
- explicit, 34
 - representations, 34
- explicitly, 187
- explosion, 98, 195
- explosive, 195
- exponential, 90, 172
- exponential time complexity, 175
- exponentially, 84, 91, 170, 187, 194
- expression
 - recursive definition, 62
- expressive power, 78, 176
- fact, 32, 192
- factor, 180, 198, 199, 201
 - of 10, 84
 - of four, 170
 - prime, 198
- factoring, 158
- factorization, 136, 198
- faculty, 14
- fairly
 - hall, 190
 - queen, 190
 - stuff, 190
- false, 20, 48, 85, 188
- falsity, 112
- feature, 90
 - size, 114
- Feynman, 130, 135
- field
 - magnetic, 145
- fields, 128
- finger, 16, 17
 - counting, 16, 17
- fingers, 17, 27
- finite, 17, 19, 20, 95, 183, 184, 187–191, 193–195
 - macroscopic, 96
 - number, 17, 192
 - state, 17, 19, 20, 124
 - automaton, 123
 - machine, 123
- finitism, 179
- fire, 42, 180, 194
- fring, 35, 36
- first, 35, 37, 41, 43, 87, 109, 119, 168, 180, 187, 197, 198, 200, 215
 - digital, 43
 - Law, 101
- fit
 - the, 123
- fitted, 28, 41
- five, 20, 22
- fixed, 19, 37, 86
 - at one mole, 97
 - internal, 106
 - mass, 97
 - stars, 32
- floating point, 185
- flux qubit, 159
- focal, 197, 200
 - length, 201
- focus
 - on, 96
- formal, 17, 24, 192
- formalism, 168, 178
- formally, 17
- formula, 106
- Fortran, 34
- fraction, 10, 11, 94, 180
 - decimal, 65
- frame, 28, 37, 38, 119, 189
 - of reference, 31, 38
- Fredkin, 117, 195
 - gate, 118
 - and Tofoli, 119
 - gate, 118
- freedom, 85
- frequency, 116, 172
- friction, 40, 43, 100
- function, 28, 31, 83, 101, 172, 184, 191, 192
 - composition, 146
 - of state, 99, 105
 - square integrable, 139
- functional, 196
- fundamental, 96, 167, 171, 182, 187, 192
- Gödel incompleteness theorems, 61
- Gödel number, 62
- Gödel, K., 61, 173
- Gödelisation, 61
- garbage, 118
 - bits, 118
- gas, 105, 192, 195
 - constant, 97
 - is compressed, 101
 - law, 97
- gate, 85, 171, 172
 - 2-qubit, 151
 - count, 90
- efficiency, 116
- insulation, 94
- one-qubit, 151
- two-qubit, 151
- Gaussian, 6, 168
 - distribution, 6

- gear, 30, 43, 45
- gearing, 34, 37, 39, 40, 43
- gearwheel, 30, 34, 46
- general
 - purpose, 34, 41, 215
- general recursion, 74
- geometric, 7, 11, 37, 198
 - entities, 7, 11
- geometry, 7, 34, 38, 183, 197
- German, 86
- GHz, 85, 172
- gods, 190
- Gover operator, 157
- grammar, 18–23
- grammatical, 18, 21, 23
- graph, 42
- gravitational, 171, 182, 183, 188, 193
- gravity, 171, 195
- Greek, 24, 25, 30, 32, 33, 180
 - astronomers, 33
- ground, 93
 - state, 109
- ground state, 159
- group, 15, 26, 28, 198
- Grover operator, 156
- Grover's algorithm, 155–157
- grow, 169, 170, 198, 201
 - as the square, 125
 - exponentially, 170
- growth, 90, 198
 - in processor power, 116
- gun, 35–37, 195
 - control, 83
 - gas, 195
 - rail, 195
- gunners, 35, 36, 39
- gunnery, 36, 39, 41
- gunpowder, 194

- Hadamard
 - operator, 145, 154, 156
 - superposition, 166
- halting, 188, 191, 215
 - function, 191
- halting problem, 158, 176
- Halting Universal Turing Machine, 73
- halts, 191
- Hamilton, 136
- Hamiltonian, 136, 137, 142, 145, 148, 153, 162
 - time-independent, 145
- Hamkins, 188
- Hamming, 7, 11, 12
- Hamming distance, 160
- hand, 15–17, 25, 26, 90, 194
- Harvard machine, 82

- Haskell, 57
- head
 - of Turing Machine tape, 69
- heat, 91, 95, 114, 188
 - dissipation, 91
 - limit, 94
 - bath, 106
 - capacity, 105
 - dissipation, 91
 - engines, 95
 - entering, 99
 - is exchanged, 100
 - reversibly, 103
 - transfer, 104
 - transferred, 101
- Heisenberg, 133, 197
- hermitian, 152
 - matrix, 144
 - operator, 141, 144, 145
- hermitian operator, 150
- high, 35, 43, 171, 193, 194
 - entropy, 108
 - speed, 86
- higher, 171, 195
 - entropy, 105
- highly, 41, 195
- Hilbert space, 139–142, 144, 147, 148, 150, 152
 - dimension, 139
 - infinite-dimensional, 139
- Hilbert's programme, 59, 61, 67
- Hilbert's Tenth Problem, 158
- Hilbert, D., 59, 60, 178
- hole, 162
- holographic principle, 185
- hot reservoir, 102
- hotter, 101
 - body, 101
- hour, 35, 90
- human, 7, 9, 14, 18, 28, 86, 187
 - computer, 86, 187
- HUTM, *see* Halting Universal Turing Machine
- Huygens, 129
- hydrogen, 133, 137
- hyperbolic, 198, 199
 - curve, 198, 199
- hypercomputation, 158
- hypercomputational, 176
- hypercomputer, 158, 191, 195, 215
- hypercomputing, 162, 187, 191
- hypothetical, 188, 193

- idea, 181, 188, 190, 191
 - of being, 167
- ideal, 97
 - gas, 97
 - law, 97
- idealized, 101, 191
- idempotent, 152
- identity element, 143
- identity operator, 154
- Interpretation
 - Many Worlds, 165
- immortals, 190
- implement, 30, 45, 120
- implementation, 33, 37, 87
- implemented, 34, 84, 196
- implication, 48, 181
 - truth table, 49
- implications, 10, 115, 170
- impurity ion, 159
- in
 - equilibrium, 96
- incompleteness theorems, 61
- indicate, 22, 24, 25, 32, 116, 182, 194
- indicator, 97
 - diagram, 97
- induces, 91
- induction, 89
- induction case, 57
- inductive proof, 57
- inductors, 121
- inference rule, 50
- infinite, 191, 193
- infinite, 181, 187–191, 193, 194
 - computation, 187
 - energy, 189
 - number, 194
 - residual, 189
 - time, 191
- infinite list, 60
- infinite tape, 178
- infinitely, 181, 183, 187, 188, 194
 - fast, 188
- infinitesimal, 100
 - change, 100
- infinities, 187, 193
- infinity, 64, 177
 - absolute, 179
 - actualized, 178
 - mathematical, 179
 - physical, 179
 - potential, 178
- infinitely, 193
- information, 11, 41, 86, 87, 108, 163, 184, 188, 190
 - and entropy, 114, 115
 - between, 41, 123, 188
 - content, 11, 112
 - content of stream, 113
 - required, 112
 - theory, 111
- inner product, 139, 140

- inner product space, 139
- input, 20, 85, 191
 - output, 118
 - variables, 118
- inputs, 20, 84
- instruction, 86
- instructions, 86
- instrumentalism, 134, 135, 163
- insulating, 91
- insulation, 94
- insulator, 93
- integer, 30, 42, 109, 180, 181, 184, 198
 - points, 198
 - positive, 56
- integers, 11
- integral, 34, 85, 191
- integrate, 121
- integrated, 41
- integrated circuit, 91
- integrator, 39, 40, 42
- Intel, 91, 92
- inter-subjective, 163
- interference, 129–131, 149
- internal, 30, 34, 98
 - energy, 98, 99
- internally, 17, 99
- interpretation, 30, 34, 106
 - Many Minds, 165
 - ontological, 165
 - quantum theory, 133, 135, 163
 - quantum theory, 163, 166
 - relative states, 165
- interpreted, 96, 193
- interpreter, 80
- intersection, 52, 198
- interval, 18, 168, 169, 188, 194
- intractable, 175
- intuitionism, 179
- invertible, 152
 - function, 153
- ion, 159
 - confined, 159
- irrational number, 66
- irreversible, 152, 165
- isomorphic, 142, 143, 149

- James Thomson, 39
- Joos, 149
- Josephson junction, 159
- joules, 115, 116, 172
- Jupiter, 192

- Kant, 163
- Kantian, 163
- Kelvin, 39, 102, 103
- Kepler, 32, 33
- Kerr, 188

- ket vector, 140, 150
- kinematic, 34, 119
- kinetic, 99
 - theory, 120
- Knight, 121

- labour, 90
- Landauer, 115
 - energy, 116
 - efficiency, 117
 - limit, 116, 117
- Landin, P., 81
- language, 10, 12–14, 17, 18, 22, 192, 194
- laser, 159, 160
- law, 95, 191
 - of physics, 114
 - of Thermodynamics, 102
- law of excluded middle, 179
- leakage, 93
 - current, 93
- Leibniz, 9, 45
- length, 46, 104, 168–171, 180, 181, 183, 184, 194, 197, 198, 200
 - of code, 113
 - of device, 168
 - of meter, 181
 - of programme, 198
- light, 18, 19, 169, 181–184, 188, 189, 192, 193, 195, 197–200
 - from point source, 192
 - in state, 18
 - red, 19
 - signal, 189
 - wavelength, 201
- limit, 13–15, 18, 23, 27, 28, 94, 96, 98, 115, 168, 171, 172, 182, 190, 194, 195, 197
 - logical, 215
- limit to computation, 59
- limitation, 87, 168, 170, 193
- Limitations, 169, 187
- limitations, 169, 187
- limited, 35, 115, 167, 195, 197
- line, 25, 181, 183, 194
 - of development, 42
 - of sight, 39
- linear operator, 141, 143
- linear time complexity, 175
- liquid state, 158
- list, 37
- local, 110
- logarithm, 108, 124, 183
- logic, 85, 125, 171, 192
 - gate, 121, 171
 - operations, 116
 - per second, 116
 - predicate, 52
 - propositional, 47
- logical, 89, 215
- logically, 101, 193
- logicism, 178
- longer, 35
- LOQC, 160
- Lord Kelvin, 39, 102, 128, 166
- low, 86, 201
 - temperature, 105
- Lukasiewicz, 50
- luminiferous aether, 129, 135
- lunar, 28, 29, 32–34
 - and solar calendars, 29
 - months, 29
 - orbit, 33
 - position, 32

- machine, 7, 9, 10, 19–22, 28, 32, 34, 41, 44, 45, 170, 184, 187–191, 193, 194, 196–201, 214, 215
 - abstract, 81
 - finite state, 19
 - Harvard, 82
 - Post, 178
 - virtual, 81
 - von Neumann, 81
- machine code, 80
- macroscopic, 95, 149, 164, 165, 167
 - description, 110
 - measurements, 109
 - system, 107
- macrostate, 107, 108, 110
- macrostates, 108, 109
- magnetic, 84, 195
 - field, 84, 159, 160
 - flux, 159
- Manchester, 87
 - Mk1, 116
- mantissa, 185
- manual, 41, 83
- manufactured, 90
- manufacturing, 84, 196
 - errors, 84, 196
 - process, 90
 - steps, 90
- Many Worlds Interpretation, 165
- mask, 197, 200
- mass, 15, 25, 46, 169, 170, 182, 183, 190, 194, 198
 - of photon, 183
 - of system, 97
 - produced, 90, 94
 - production, 91
- master
 - calculus, 10
- matching, 31, 35
- material, 7, 93, 129, 167, 184, 187, 193

- materialism
 - classical, 166
- mathematical, 7, 9–11, 16, 84, 191, 193
 - concepts, 10, 11
 - construct, 118
 - techniques, 11
- mathematical infinity, 179
- mathematician, 9, 10, 102, 187, 192
 - does, 9, 10
- mathematicians, 7
- mathematics, 6, 7, 9–11, 34, 180, 184
- maths, 6–8, 11, 27, 191, 193
- Matlab, 34
- matrix, 37, 39, 89, 140, 143, 146
 - components, 146
 - inverse, 144
 - multiplication, 143
 - row, 139
 - self adjoint, 144
- matter, 10, 95, 183, 191
- maximal, 103
- maximum, 109, 168, 169, 201
 - entropy, 110
 - entropy, 109
- Maxwell, 129
- Maxwellian, 192
 - theory, 192
- mean, 34, 169, 193
 - number, 85
- mean value, 142
- meaning, 6, 18, 109
- measurable, 85, 167, 171
- measure, 107, 180, 181, 184
 - of disorder, 110
- measured, 38, 97, 180
- measurement, 107, 134, 137, 139, 141, 147–149, 152, 154, 156, 158, 168, 169, 171, 180, 182, 190, 192
 - probabilities, 136
- measurement gate, 152, 154, 165
- measurements, 85, 183
- measures, 85, 183
- mechanical, 16, 17, 23, 28, 32, 33, 35–37, 39–43, 46, 96, 168, 188, 190, 196
 - computer, 28, 39, 41, 42
- mechanical procedure, 59
- mechanically, 45
- mechanics, 96, 167, 168, 182, 191, 193, 194
 - Newtonian, 133
- mechanism, 28, 29, 32–35, 41, 43, 120, 188, 198
- mechanisms, 34
- member, 51
- memory, 20, 23, 86, 167–171, 187
- Menninger, 12, 25, 26
- mental, 10, 16, 23, 45
 - arithmetic, 45
- mercury, 86
- Mermin, 163, 166
- message, 112, 189
- metaphysical
 - model, 148
- metaphysics, 134, 163, 166
- meters
 - per
 - second, 37
- Metonic, 29, 34
 - cycle, 29, 34
- microcanonical, 106
- microscopic, 95, 167
- microstate, 107, 108
 - compatible, 108
 - corresponding, 108
- microstates
 - are occupied, 108
 - available, 109
 - of system, 108
 - of which, 108
- mirror, 182, 196–200
- mode, 92
- model, 10, 14, 26, 32–34, 36, 39, 45, 46, 93, 183, 193, 195, 215
 - a real system, 10
 - is represented, 33
 - of CMOS, 121
- modelled, 10, 26, 33, 38, 88, 106, 180
 - as, 106
- modelling, 9, 10, 26
- models, 12, 33, 42, 119, 167, 187, 191, 198
- modernism, 166
- Modus Ponens, 50
- mole, 97
- molecule, 158
- molecules, 106
- moment, 18, 188, 197, 200
- momentum, 109, 136, 137, 139, 169, 171, 197
- months, 29
- moon, 28, 32–34
- Moore, 17
- motor, 16, 17, 20, 23
 - control, 17, 20, 23
 - schema, 16, 17
- Mott, 134
- multiiverse, 166
 - branch, 165
- multiple, 10, 11, 14, 89, 215
 - multiple tape head Turing Machine, 173
 - multiple tape Turing Machine, 173
 - multiplication, 26, 31, 35, 37, 39, 40, 44, 56, 84, 196
 - tables, 26
 - multiplications, 26, 34
 - multiplied, 26, 44
 - multiply, 26, 38, 41, 113
 - multiplying, 45, 83
 - multiverse, 165
- Nagel, E., 63, 173
- NAND, 93, 121
- natural, 7, 17, 22, 35, 168
 - languages, 22
 - logarithm, 115
 - selection, 9
- natural number, 57
- nature, 10, 96, 167, 180
 - of charge, 114
- naval, 35, 37, 39, 41
- navies, 35, 41
- negation
 - simplification, 50
- negative, 35, 91
 - charge, 91
- Nemeti, 188, 189
- nervous, 17
- nested, 17, 21, 22, 44
- new state, 69
- new symbol, 69
- Newman, J. R., 63, 173
- Newton, 7–10, 129, 191, 194
- Newton's
 - laws, 9, 192
- Newtonian, 188, 191, 193–195
 - mechanics, 191–194
- NMOS, 91, 92
- NMR
 - liquid, 158, 159
 - solid state, 159
- no
 - process, 102
- no-cloning theorem, 154, 161
- noise, 84
 - level, 116
 - voltage, 114
- non-deterministic, 174
- non-deterministic polynomial time
 - complexity, 175
- non-terminals, 18
- norm, 144
- normal, 23, 38, 39, 46, 85, 90, 197
- normal form, of λ expression, 76
- not
 - truth table, 49

- noumenal, 163
- NP, 158
- NP (non-deterministic polynomial time complexity), 175
- number, 10–15, 17, 18, 21–24, 28, 34, 35, 37, 38, 42, 43, 45, 84, 168, 170, 180, 181, 183, 184, 187–190, 192–194, 197, 198, 201, 214
 - as property of set, 58
 - cardinal, 65
 - Gödel, 62
 - irrational, 66
 - natural, 57
 - of AND gates, 125
 - of bits, 113
 - of cogs, 46
 - of degrees, 85
 - of digits, 113, 192
 - of electrons, 85
 - of layers, 91
 - of microstates, 108
 - of particles, 106
 - of possible, 110
 - of properties, 97
 - of quanta, 85
 - of reconstructions, 28
 - of states, 125
 - of theorems, 189
 - of transistors, 90
 - of wavelengths, 181
 - of ways, 109
 - of wires, 124
 - ordinal, 65
 - rational, 65
 - real, 66
 - system, 13, 15
- numbers, 12–16, 21, 23, 24, 25, 26, 167, 171, 180, 181
 - and sets, 58
- numberword, 21
- numerator, 35
- numerical, 24, 34, 192, 215
 - precision, 192
- objective, 149
- objectivity
 - strong, 136, 163, 166
 - weak, 163, 164
- obol, 25
- observable, 137, 139, 141, 142, 144, 150, 165
 - compatible, 139
- observation, 41, 100, 136, 192
- observational prediction, 163
- observer, 134, 136, 149, 163–165, 189
- observer independence, 165
- ontological
 - model, 148
- ontology, 134, 163
- operate, 121, 172, 187–189
- operated, 25, 94
- operating, 38, 103, 172, 188, 189
 - time, 188
 - voltage, 115
- operation, 22, 23, 34, 85, 118, 168, 169, 172, 187, 188, 198
- operations, 27, 34, 35, 83
 - per
 - second, 116
- operator, 27, 84, 167, 196
 - single-qubit, 151
- operators, 169
- optical, 32, 85, 181, 197, 198, 201
 - gates, 160
 - system, 197
- optimal, 88, 198
 - size, 88
- optimise, 125
- or, 48
 - truth table, 49
- oracle, 174
- orbit, 32, 33, 190, 192
- ordinal number, 65
- orthogonal
 - vectors, 144
- orthonormal basis, 141, 142, 144, 157
- oxide, 94
- P (polynomial time complexity), 175
- packages, 11
- packets, 86
- paper, 86, 187, 191–193
- papyrus, 27, 28
- parabolic, 194–196
 - mirrors, 196
- paradigm, 125
- paradox, 58, 192
 - Russell's, 60
- paradoxical, 215
- parallel, 33, 37, 38, 42, 90, 193
- parallel universe, 166
- parallelism, 122
 - quantum, 135, 154, 166
- parameter, 103, 192, 196
 - of system, 196
- parameters, 170
- parametric, 197
- particle, 96, 110, 131, 134, 193–195, 197
 - energy, 109
 - levels, 110
 - state, 109
- particles, 106, 128
- particles , 99
- Pascal, 43, 45
- path, 33, 88, 197–199
 - length, 197, 198
- pattern, 20, 23, 26, 198–200
- patterns, 13, 15, 20
- Pauli matrices, 143
- Pauli matrix, 144
- Peano arithmetic axioms, 57
- Peano arithmetic
 - as Turing Machine, 71
- Peano, G., 55
- pebbles, 11, 12, 23, 25, 26
- Perga, 32, 33
- perturbation, 120
- perturbed, 120
- phase, 29, 114, 155, 200
- phases, 26
 - of the moon, 29
- phenomena, 192
- phenomenal, 163
- philosophical, 96, 191
- phonon, 162
- photo, 192, 198
- photography, 28
- photon, 129–131, 134, 137, 159, 160, 182, 183, 193, 195, 197, 199
 - arriving, 197, 199
 - idler, 132
 - signal, 132
- photons, 181, 183
- photosensitive, 200
- physical, 6, 9, 17, 23, 25, 26, 28, 29, 34, 84, 168, 170, 171, 183, 188, 191–193, 196, 198, 215
 - models, 191
 - system, 111, 167
- physical infinity, 179
- physically, 33, 39, 85, 167, 171, 187
- physicist, 10, 102, 192
- physicist's, 10
- physicists, 7
- physics, 6, 7, 10, 11, 99, 128, 167, 192, 193
- Pieris, 163
- pilot waves, 164
- pin, 28, 32
 - and slot, 32
- pinwheel, 44–46
- planar, 90, 196
- Planck, 102, 129, 133, 169, 171, 172, 182, 183
- Planck's constant, 129

- reduced, 138
- Platonic realism, 177
- plus, 10, 11, 106
 - external, 10
- PMOS, 91, 92
- pocket, 25
 - calculator, 25
- Podolsky, 148
- point, 86, 168, 169, 183, 188, 192, 193, 197–200
 - masses, 193
 - of view, 99
 - on, 199
 - diagram, 98
 - ruler, 38
 - source, 192
- pointer, 29, 34, 35, 41
- pointing, 192
- points, 11, 183
- Poisson, 85
 - distribution, 85
- polarization, 160
- polynomial, 191
- polynomial time complexity, 175
- positive, 91
- positive integer, 56
- Post machine, 178
- postulate, 11, 108, 168
- potential, 99, 168, 169
 - energy, 99
- potential infinity, 178
- potentially, 20, 123, 191
 - infinite, 191
- powder, 195
- power, 9, 19, 35, 94, 115, 171, 172, 191, 196
 - loss, 122
 - consumption, 172
 - dissipation, 121
 - supply, 121
 - used, 91
 - would, 122
- power set, 59, 66
- powerful, 21, 117
- powers, 25
- practical, 95, 181, 183, 191, 201
 - application, 6, 7
 - engineering, 112
- practically, 94
- precision, 29, 35, 41–43, 167, 168, 170, 171, 183, 185, 192, 201
- predicate
 - as set, 53
- predicate logic, 52
- predicting, 30, 34
- predictions, 6, 7, 32, 96, 182, 191, 192
 - about, 191
 - of lunar position, 32
 - of quantum theory, 182
 - of statistical mechanics, 96
- predictive, 191
- premise, 50
- pressure, 9, 41, 96
- price, 26, 28, 90
 - of cargo, 26
- prime, 180, 198, 201
 - factor, 198, 199, 201
 - factorization, 198
 - number, 201
- primitive recursion, 74
- prior, 15, 90, 168, 189, 195
- probabilistic, 156
- probabilities, 160
- probability, 108, 113, 134, 139, 141, 148, 152, 168, 169
 - distribution, 111
 - distribution, 108
 - of failing, 168
 - of switching, 85
- probable, 108, 112
 - message, 112
- procedure, 12, 24, 26, 191
 - decision, 176
 - effective, 174
- procedure, effective, 68
- process, 83, 168
 - as, 8
 - of reckoning, 28
- processes, 192
- processing, 117, 189
- processor, 94, 194
 - chip, 122
- produce, 23, 34, 42, 90, 184, 193, 195
 - infinities, 193
- produced, 20, 21, 25, 45, 46
- product, 94
 - of human mind, 7
 - state, 150
- product space, 157
- product state, 147, 150
- product states, 146, 147
- production, 19, 21, 46, 91
 - of calculators, 91
- productions, 18–20
- profound, 192
- program, 80, 184, 188, 198
- programmed, 9, 83, 188
 - by plug boards, 83
- programming, 7, 9, 17, 192
 - language, 192
 - languages, 17
- programs, 9, 11
- projection operator, 152
- projector, 152
- Prolog, 57
- proof, 13, 33
 - as number, 62
 - by contradiction, 179
 - deductive, 50
 - existence, 179
 - inductive, 57
 - base case, 57
 - induction case, 57
 - of equivalence, 33
- proofs, 180
- propagate, 43, 88, 188
 - along, 88
 - halfway, 88
- propagation
 - distance, 88
 - time, 88
- proportional, 94, 108, 183, 195, 197
 - to
 - the square, 122
- proposition, 112, 187
- propositional logic, 47
- proved, 32, 191
- proven, 105
- proves, 6
- punched, 86
- Pythagoras
 - theorem, 181
- Pythagorean, 11
 - theorem, 11
- quanta, 85
- quantification
 - existential, 53
 - universal, 53
- quantify, 112
- quantised, 109
- quantization, 129, 133
- quantized, 193
- quantum, 85, 167–169, 171, 172, 182, 190, 192–194, 198
 - algorithm, 136, 155–157
 - assembly, 106
 - circuit, 145, 146, 149, 151, 153, 162
 - circuit diagram, 145
 - computer, 135, 136, 145, 152–155, 157–160
 - computing, 128–167
 - error correcting code, 162
 - error correction, 161, 162
 - gate, 135, 136, 145, 149, 151, 153
 - gates, 158, 160
 - mechanical, 96, 168
 - mechanics, 120, 128–168, 182, 193
 - NOT gate, 145
 - potential, 135, 164

- register, 149, 151, 153, 155
- rules, 136
- search, 155
- splits, 165
- state, 139, 140, 159
- superposition, 106, 108
- system, 148
- theory, 96, 128–167, 193
- wire, 145
- wires, 160
- quantum adiabatic computer, 162
- quantum circuit model,
 - 153, 162
- quantum dot, 159
- quantum register, 152
- quasi, 169
- quasi-particles, 162
- quantifier
 - recursive definition, 55
- qubit, 135–137, 139, 142–162
- qubit , 146
- query, 7
- qunatisation, 119
- radiation, 90, 184, 198
- radius, 119, 171, 183, 197
- RAM, 89
- random, 84, 184
 - access, 87
- randomly, 13, 86
- range, 22, 27, 35, 36, 39–42, 170, 172,
 - 181, 191, 192, 201
 - and bearing, 36
 - clock, 39
 - of position, 170
 - of the target, 35
- range , 96
- ratio, 6, 31, 34, 43, 180, 197
 - as, 35
- rational, 31, 34
 - number, 31, 34
- rational number, 65
- rationality, 184
- rations, 13, 26
- ratios, 180
- real, 83, 168–171, 181, 183, 184, 187,
 - 191, 193, 195, 198
 - number, 181, 187
 - numbers, 167, 171, 180
 - physical, 193
 - time, 83
 - valued, 196
- real number, 66
- realism, 149, 163, 177
 - classical, 130
 - far, 149
 - near, 149
 - physical, 128
- realist, 148
- realistic, 94
- reality, 8, 9, 110, 133, 149, 192
 - objective, 128, 134, 135, 163, 164
 - observer-independent, 163
 - physical, 148
- reals, 171, 184
- reasoning, 9, 10, 110, 184
 - power, 9
- reckoning, 24–26, 28, 34, 44
 - table, 26
 - tables, 27
- recursion, 55
 - base case, 55
 - general, 74
 - primitive, 74
 - recursion case, 55
- recursion case, 55
 - recursive definition, 56
- recursive, 15, 27
- recursive definition, 56
 - expression, 62
- reduction, 115
- redundancy bits, 160
- recursive function theory, 81
- reference, 31, 38, 119, 189
- referential, 215
- reflector, 119
- register, 20, 25, 26, 45, 118, 189
- reification, 166
- relative, 28, 30, 32, 35–40, 42, 89,
 - 188, 200
 - velocity, 36, 38
- relative states, 164
- relatively, 32, 99, 192
- relativistic, 193
- Relativity
 - Special Theory of, 129
- relativity, 128, 133, 148
- relaxed, 101
- reliable, 190
- remainder, 57
- replace
 - Turing Machine, 71
- replacement, 63
- resistance, 100
 - of circuit, 121
- resistor, 93, 121
 - to ground, 93
- result, 31, 34, 45, 86, 94, 168–170, 183,
 - 188, 191–194, 197
 - of multiplying, 45
 - of plotting, 183
- reverse, 33, 40, 45, 104
 - direction, 33, 118
- reversed, 99
- reversibility, 100
- reversible, 99
 - change, 104
 - engine, 103
- reversibly, 101
- reversing, 100
- ring
 - non-commutative, 143
- role, 21
- Roman, 15, 16, 25
- Rosen, 148
- rote, 7, 27
- round, 40, 43, 192
- rule
 - inference, 50
 - Modus Ponens, 50
 - of Turing Machine, 69
- ruler, 38, 180, 181
- ruler e
 - had, 38
- rules, 10, 19, 21, 86, 187
 - of grammar, 21
 - plus, 10
- run, 122
- running, 43, 117, 188
- Russell's Paradox, 60
- Russell, B., 60, 178
- sailing, 37
- Salamis, 24, 25
- Saros, 29, 30
 - cycle, 29, 30
- Saturn, 192
- scalar product, 139, 147
- scaling, 158, 159
- schema, 16
- schematic, 36, 41, 125
- scheme, 85
- Schickard, 43
- Schrödinger, 107, 109, 133, 139,
 - 167, 168
 - equation, 107
- Schrödinger equation, 136, 145, 147,
 - 148, 153, 165
 - time dependent, 142
- Schwarzschild radius, 190
- select, 88, 189
 - the, 88
- selected, 87
- selective, 9
 - pressure, 9
- self, 215
 - referential, 215
- self-reference, 60, 62
 - in *λcalculus*, 78
- semi-decidable, 176
- semiconductor, 159
- separable, 150

- sequence, 16, 19, 20, 43, 112, 184
 - of bytes, 184
 - of colours, 19
 - of random real, 184
 - of tosses, 112
 - of wheels, 43
 - sequences, 23
 - of actions, 23
 - sequencing, 17, 18, 20
 - sequentially, 86
 - set
 - definition by property, 60
 - difference, 52
 - element, 51
 - empty, 51
 - intersection, 52
 - member, 51
 - of sets, 59
 - power, 66
 - size, 58
 - union, 52
 - set theory, 51
 - set:subset, 52
 - sets
 - and numbers, 58
 - Shannon, 111–113, 115
 - Shannon's
 - information, 111
 - ship, 35–39, 41
 - sailing, 37
 - Shor, 136, 157, 162, 198
 - Shor's algorithm, 136, 155, 157
 - shot, 35, 36, 42, 85
 - noise, 85
 - signal, 85, 88, 188, 189
 - significance, 167
 - significant, 35, 41–43, 84, 197
 - digits, 35, 197
 - figures, 43
 - significantly, 83, 171, 183
 - signs, 13
 - silicon, 91
 - area, 125
 - dioxide, 93
 - silicon 28, 159
 - simple, 7, 10–12, 17, 20, 22, 23, 26, 27, 37, 93, 197
 - abstraction, 11, 12
 - model, 93
 - simplest, 7, 97, 180
 - such, 119
 - simplicity, 7, 85, 168
 - simplified, 111
 - simply, 35, 102
 - simulate, 188, 193, 196
 - simulated, 32, 126
 - simulation, 126
 - simultaneously, 90
 - single, 41, 44, 45, 88, 172, 197, 198
 - particle, 109
 - substance, 106
 - size
 - of set, 58
 - Smith, 193
 - social, 9
 - behaviour, 9
 - society, 12, 26
 - software, 10, 11, 122
 - development, 11
 - packages, 11
 - solid-state physics, 162
 - solution, 34, 36, 38, 106, 191, 193
 - solve, 107, 215
 - solved, 46, 83, 168
 - solving, 109
 - special, 14, 215
 - specific, 9, 10, 12
 - specification, 84, 191
 - specified, 109, 181, 188
 - specify, 110, 192, 197
 - exactly, 110
 - specifying, 17, 96
 - spectral expansion, 142
 - spectral theorem, 141
 - speech, 18, 21
 - speed, 32, 35, 37, 38, 83, 117, 169, 170, 182, 183, 188, 194, 195
 - of light, 169, 182, 183
 - of operation, 83
 - of sound, 195
 - of the moon, 32
 - spend, 122
 - sphere, 119
 - of radius, 119
 - spin, 138, 144, 159
 - nuclear, 158, 159
 - spiral, 29
 - spontaneous parametric
 - down-conversion, 131
 - spread, 25, 169, 193
 - sq, 94
 - square, 88, 169, 180, 198
 - of voltage, 115
 - root, 88, 198
 - squares, 180
 - SQUID, 159
 - stable, 97
 - stack, 22, 196
 - automata, 196
 - automaton, 196
 - machine, 22
 - standard
 - basis, 150
 - Stapp, 134
 - state, 12, 17–21, 24, 90, 98, 107, 125, 167, 168, 192
 - vector, 125
 - automata, 123
 - automaton, 127
 - bit, 126
 - current, 69
 - machine, 20, 125
 - new, 69
 - of Turing Machine, 69
 - variable, 105
 - vector, 123, 167
 - word, 22
- state transition diagram, 70
 - state vector, 140, 142, 148, 152, 153
 - statement, 18, 167
 - states, 17, 20
 - static, 93, 100
 - stationary, 107
 - microstate, 108
 - state, 107
 - statistical, 96
 - mechanical, 106
 - mechanics, 96
 - statistician, 6
 - Steane, 162
 - step, 12–14, 17, 25, 45, 90, 188, 198
 - stepped, 46
 - wheel, 46
 - steps, 12, 17
 - Stern–Gerlach experiment, 138
 - Stirling's formula, 111
 - storage, 83, 171
 - store, 84, 86, 168–170, 190
 - stored, 12, 86, 167, 170, 171
 - storing, 88, 167
 - stylus, 13, 25
 - sub, 35
 - subitizing, 13, 15, 23, 27
 - ability, 13, 15
 - faculty, 14
 - subject, 171, 196
 - subset, 52, 168
 - subspace, 140
 - substance, 96
 - substitute, 12, 191
 - subtract, 37, 44
 - subtraction, 35, 44, 57
 - success, 96, 168, 192
 - successive, 25, 90
 - successor, 56
 - sun, 28, 32
 - super, 187
 - super-Turing, 176, 195
 - superconducting, 159
 - superluminal signalling, 148

- superposition, 108, 153, 154, 161, 165
 - of states, 108
- superposition, 165
- switch, 91, 196
 - each, 94
- switching, 85, 172
- symbol, 6, 18, 21, 93
 - current, 69
 - new, 69
- symbols, 6, 13–15, 18
 - of language, 18
- syntax, 18, 19
- system, 12, 37, 38, 96–98, 167–170, 172, 184, 187, 189, 192, 193, 196, 197, 215
 - at
 - equilibrium, 108
 - at equilibrium, 97
 - based, 196
 - being, 108
 - can, 97, 184
 - could, 108
 - energy, 110
 - from, 197
 - has, 97, 172, 197
 - in
 - equilibrium, 99
 - is tallying, 15
 - of Cartesian coordinates, 38
- systems, 169, 170

- table, 20, 24, 26–27, 41, 44, 113, 170
- tables, 25, 27
- tallies, 13, 23
- Tam Lynne, 190
- tape, 86, 187, 196
 - Turing Machine, 69
 - cell, 69
 - direction, 69
 - infinite, 178
- tape head, 69
- technical, 20, 108, 194
- technique, 26, 27, 90, 187, 195, 201
- technologies, 27
- technology, 14, 15, 32, 41, 88, 117, 172
 - has, 91
 - of 1961, 88
 - of Dumaresq, 41
- teeth, 31, 34
- telephone, 83
- temperature, 97, 100, 101, 104, 172, 184, 188, 196
 - of system, 104
- temple, 12, 13

- temporal, 7, 27
- tensor product, 146–152
 - operator, 150
 - space, 142, 148–150
- term, 7, 23, 85, 190, 191, 194, 195
- terminal, 18, 21
 - symbols, 18
- terminals, 18, 21
- termination, 74
- terms, 18, 168, 171, 181
 - of wavelengths, 181
- Thaller, 42
- the
 - original, 40, 41, 200
- Theorem, 11, 183
- theorem, 11, 50, 115, 169, 180, 183, 188, 189
 - of Boltzmann, 115
 - of Pythagoras, 180, 183
- theorems, 11
- theoretical, 119, 169, 187, 192
- theories, 171
- theory, 22, 95, 167, 182, 183, 188, 192–194
- thermal, 84, 184
 - noise, 114
 - contact, 101
 - energy, 99, 184
 - equilibrium, 100, 184
 - noise, 114
- thermal noise, 159
- thermally, 99
- thermodynamic, 96, 171
 - equilibrium, 96
 - system, 96
- thermodynamics, 89, 95, 96, 184
- think, 12, 37
- thinking, 7
- Thomson, 39, 102
- threshold, 10
- time, 35, 88, 90, 167–169, 188–191, 193–195, 201
 - interval, 169, 194
 - taken, 15, 84
 - to
 - propagate, 88
- time evolution, 142, 145, 152, 153
- times, 13, 168–171, 183
- TM, *see* Turing Machine, 187–189, 191, 196, 198, 215
 - head, 196
 - tape, 196
- TMs, 187
- Tofoli, 117
- token, 12–14, 24
- tokens, 12–15, 24, 26, 27
 - into, 14, 24

- topological quantum computing, 162
- total, 24, 26, 88, 184, 195
 - energy, 107
 - number, 24, 88
 - work, 90
- tractability, 174
- traffic, 18, 19, 126
 - light, 18, 19
 - simulation, 126
- trajectories, 84, 194, 195
- transcend, 187
 - the, 187
- transfer, 41, 45, 102, 188
 - of energy, 99
- transferred, 40
- transferred, 90, 96
- transferring, 99
 - heat, 102
- transistor, 85, 122
- transition, 43, 127
- transitions, 17, 20, 22, 181
- translated, 34, 36
- translation, 196
- transmission, 41, 112, 194
 - of information, 112
- transmit, 112
 - information, 123
- transmitted, 41, 90
- triode, 116
- TRUE, 85
- true, 48, 85
- truth, 112, 194
- truth table, 49
 - implication, 49
 - not, 49
 - predicate, 52
 - quantifier, 54
 - size, 50
- Tsushima, 35
- tube, 83
- Tucker, 193–195
- Turing, 10, 86, 187, 188, 191, 193, 194, 199, 214, 215
 - had, 194
 - Machine, 10, 135, 188, 191, 193, 215
- Turing Machine, 67, 176
 - multiple tape, 173
 - multiple tape heads, 173
 - tape, 69
 - Universal, 71
- Turing's, 184
- Turing, A. M., 67, 174
- turntable, 40, 42

- Umbriel, 190
- unary, 27

- unbounded, 196
- uncertainty, 182
- undecidability, 71
 - of oracle machine termination, 175
- undecidable, 176
- under, 17, 193, 196
- uniform, 32, 194
- union, 52
- uniquely, 113
- unit, 14, 24, 43, 83, 180, 182, 198
 - of measure, 182
 - of measurement, 180
 - of study, 96
 - sheep, 14
 - tokens, 14, 24
 - wheel, 43
- unitary, 152
 - evolution, 164
 - matrix, 144
 - operator, 144–146, 151–153, 155, 166
 - operators, 150, 151
 - transformation, 144, 145, 149, 150
- units, 13, 14, 24, 25
 - and tens, 13
- universal, 10, 83, 167, 168, 187, 191
 - mathematical, 10
- universal quantification, 53
- Universal Turing Machine, 71, 80
- universe, 11, 12, 90, 180, 184, 188, 194
 - quantum state of, 164
- unreasonable, 9, 11
 - effectiveness, 11
- UTM, *see* Universal Turing Machine
- value, 15, 24, 25, 84, 167–170, 193, 197, 201
- valued, 24, 191, 195
- values, 24, 167–170
 - for parameters, 170
 - of counters, 24
- valve, 8, 91
 - computer, 8, 91
- valves, 91
- variable, 32, 40, 41, 44, 46, 48, 84, 167, 195, 196
- vector, 37, 38, 123, 124, 167, 168
 - column, 146
 - component, 146
 - components, 150
 - length, 140, 144
 - of quantum system, 167
 - space, 140, 147
- vector space, 139, 140
 - finite-dimensional, 139
- velocity, 32, 35–38, 40, 109, 119, 169, 192–195
 - vector, 37, 38
- virtual machine, 81
- visible, 201
- visual, 14, 16
 - process, 16
- volt, 85
- voltage, 84, 91
 - across, 121
 - and capacitance, 94
- volume, 96, 97, 109, 184, 192
- von Neumann, 123, 124, 127
 - computer, 123
- von Neumann machine, 81
- von Neumann, J., 81
- war, 41, 86
- warfare, 35
- watts, 94
- wave, 87, 130, 168, 169, 193, 197, 199
- wave function, 139
- wave mechanics, 139
- wave–particle duality, 130, 133
- wavelength, 181–183, 189, 197, 200
 - of light, 182, 200
 - of photon, 182, 183, 197
- waves, 134
 - electromagnetic, 129
 - matter, 133
 - pilot, 135
- wheel, 30, 32, 34, 35, 40, 43, 46
- Wheeler, 167
- Wigner, 6, 7, 11
- Wigner's, 11
 - question, 6, 7
- William, 39, 102
 - Thomson, 39, 102
- Williams, 83
 - tubes, 84
- wires, 123
- word, 21, 22
- work
 - is configurative, 104
 - is done, 100
- world-lines, 162
- worth, 90
- write, 13, 19, 21, 83, 188, 191, 192
 - down, 13, 21
- writing, 7, 188
- written, 87, 169, 192
 - down, 27
 - number, 13
- Young, 130
- Younis, 121
- Zeh, 149
- Zermelo-Fraenkel, 52
- zero point, 171
- Zeus, 190
- ZFC, 188, 189
 - set theory, 188
- zodiac, 28
- zone, 92