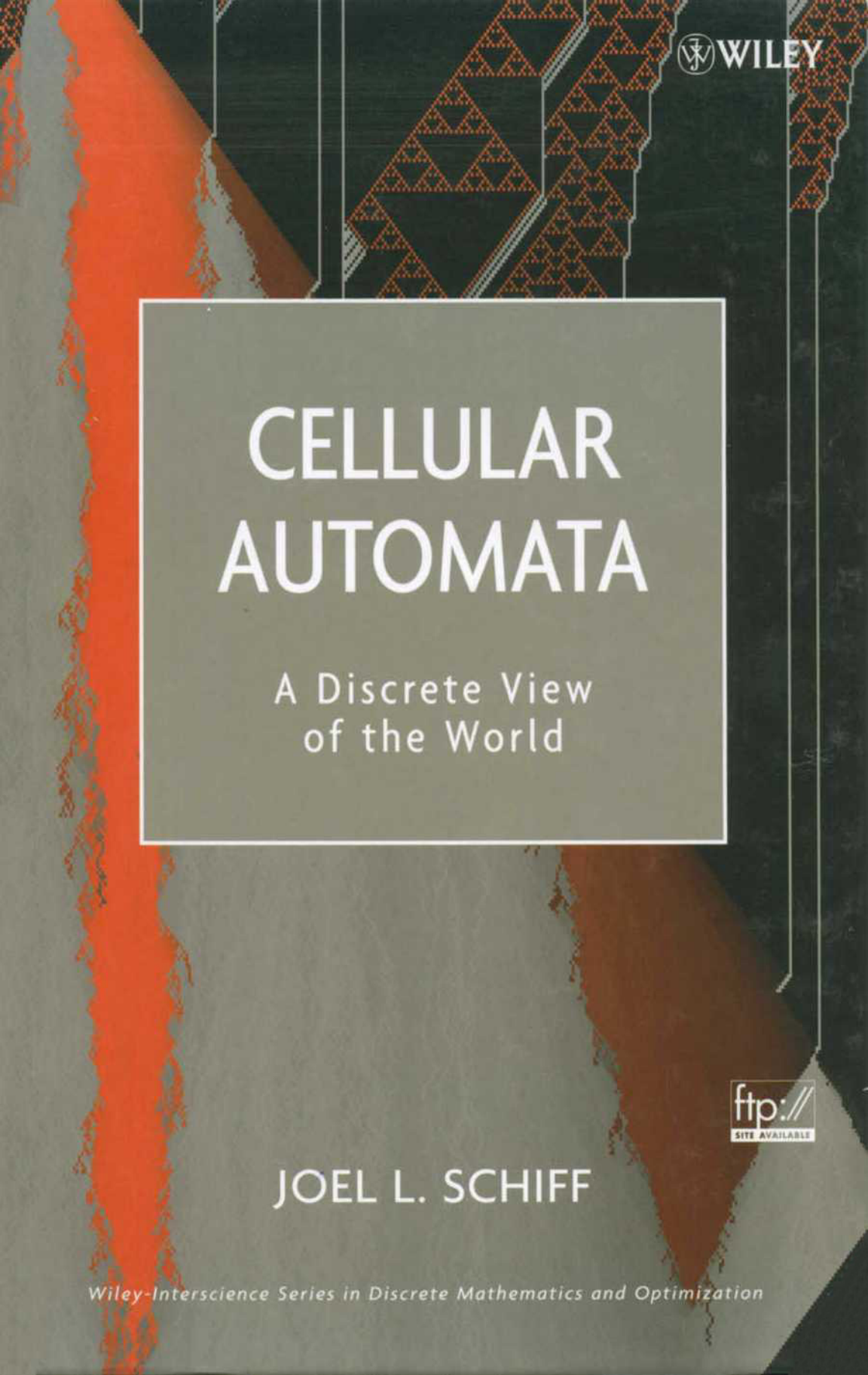
WILEY
CELLULAR AUTOMATA
A Discrete View of the World
ftp://
SITE AVAILABLE
JOEL L. SCHIFF
Wiley-Interscience Series in Discrete Mathematics and Optimization

This page intentionally left blank

CELLULAR AUTOMATA

THE WILEY BICENTENNIAL–KNOWLEDGE FOR GENERATIONS

Each generation has its unique needs and aspirations. When Charles Wiley first opened his small printing shop in lower Manhattan in 1807, it was a generation of boundless potential searching for an identity. And we were there, helping to define a new American literary tradition. Over half a century later, in the midst of the Second Industrial Revolution, it was a generation focused on building the future. Once again, we were there, supplying the critical scientific, technical, and engineering knowledge that helped frame the world. Throughout the 20th Century, and into the new millennium, nations began to reach out beyond their own borders and a new international community was born. Wiley was there, expanding its operations around the world to enable a global exchange of ideas, opinions, and know-how.

For 200 years, Wiley has been an integral part of each generation's journey, enabling the flow of information and understanding necessary to meet their needs and fulfill their aspirations. Today, bold new technologies are changing the way we live and learn. Wiley will be there, providing you the must-have knowledge you need to imagine new worlds, new possibilities, and new opportunities.

Generations come and go, but you can always count on Wiley to provide you the knowledge you need, when and where you need it!

William J. Pesce

WILLIAM J. PESCE
PRESIDENT AND CHIEF EXECUTIVE OFFICER

Peter Booth Wiley

PETER BOOTH WILEY
CHAIRMAN OF THE BOARD

CELLULAR AUTOMATA

A Discrete View of the World

Joel L. Schiff

University of Auckland

WILEY-
INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Wiley Bicentennical Logo: Richard J. Pacifico

Library of Congress Cataloging in Publication Data:

Schiff, Joel L.
Cellular automata : a discrete view of the world / Joel L. Schiff
p. cm. – – (Wiley series in discrete mathematics and optimization)
Includes bibliographical references and index.

ISBN 978-0-470-16879-0 (cloth)

Printed in the United States of America

10 9 8 7 6 5 4 3 2

Dedicated to my beloved wife, Christine

Life is a peephole, a single tiny entry onto a vastness — how can I not dwell on this brief, cramped view I have of things?

— *From* Life of Pi *by Yann Martel*

CONTENTS

This page intentionally left blank

PREFACE

... synthetic universes defined by simple rules...
— *Tommaso Toffoli and Norman Margolus* – Cellular Automata Machines

I expect the children of 50 years from now will learn cellular automata before they learn algebra.
— *Stephen Wolfram* – New Scientist, *November 18, 2006*

The history of cellular automata is only quite recent, coming to life at the hands of two fathers, John von Neumann and Stanislaw Ulam in the early 1950s. Subsequent work in the early 1960s included that of Ulam and his co-workers at Los Alamos and by John Holland at the University of Michigan whose work on adaptation continued for several decades. Early theoretical research was conducted by Hedlund, Moore, and Myhill, among many others, not always under the name of cellular automata (CA), since the concept was still in its formative stages. A big boost to the popularization of the subject came from John Conway's highly addictive Game of Life presented in Martin Gardner's October 1970 column in *Scientific American*. Still the study of CA lacked much depth, analysis, and applicability and could not really be called a scientific discipline.

All that changed in the early 1980s when physicist Stephen Wolfram in a seminal paper, "Statistical mechanics of cellular automata," initiated the first serious study of cellular automata. In this work and in a series of subsequent ones Wolfram began producing some of the images that have now become iconic in the field. Conferences were organized and people from various disciplines were being drawn into the field. It is now very much an established scientific discipline with applications found in a great many areas of science. Wolfram has counted more than 10,000 papers referencing his original works on the subject and the field of cellular automata has taken on a life of its own. Another milestone in the field was the publication in 2002 of *A New Kind of Science*, Wolfram's 1200 page *magnum opus*, a monumental work which brought the subject to the attention of a truly global audience.

The CA paradigm is very appealing and its inherent simplicity belies its potential complexity. Simple local rules govern an array of cells that update the state they are in at each tick of a clock. It has been found that this is an excellent way to analyze a great many natural phenomena, the reason being that most physical processes are themselves local in nature — molecules interact locally with their neighbors, bacteria with their neighbors, ants with theirs, and people likewise. Although natural phenomena are also continuous, examining the system at discrete time steps does not really diminish the power of the analysis. So in the artificial CA world we have an unfolding microcosm of the real world.

One of the things self-evident to everyone is the order that is found in Nature. From an ameoba to plants to animals to the universe itself, we find incredible order everywhere. This begs the obvious questions: Where did this order come from — how could it have originated? One of the fundamental lessons of cellular automata is that they are capable of self-organization. From simple local rules that say nothing whatsoever about global behavior, we find that global order is nonetheless preordained and manifest in so many of the biological and physical systems that we will consider. In the words of theoretical biologist Stuart Kauffman, it is "order for free." It is this order for free that allows us to emulate the order we find in Nature.

Related to the creation of order is the notion of complexity. How can a finite collection of chemicals make up a sentient human being? Clearly the whole is greater than the sum of its parts. How can termites build complex structures when no individual termite who starts a nest

even lives to see its completion? The whole field of complexity has exploded over recent years and here too CA play their part. One of the most endearing creatures that we shall encounter is Langton's ant in Chapter 6, and this little creature will teach us a lot about complexity.

Of course it is no longer possible in a single text to cover every aspect of the subject. The field, as Wolfram's manuscript count shows, has simply grown too large. So this monograph is merely an introduction into the brave new world of cellular automata, hitting the highlights as the author sees them. The mathematics in Chapter 1 serves mainly to relate the notions of fractals, dimension, information, and entropy, which all feature in the science of CA. And in Chapter 2, dynamical systems are mathematical by their very nature. However, in the remainder of the text, outbreaks of mathematics have been deliberately kept to a minimum. On the other hand, a more advanced and mathematical account of cellular automata can be found in the 2002 book by Ilachinski. An excellent pioneering work is, *Cellular Automata Machines,* by Tommaso Toffoli and Norman Margolus, both of whom have made important contributions to the field. Many of the topics mentioned herein, such as dynamical systems, chaos, artificial intelligence, and genetic algorithms, are only touched upon in the context of cellular automata, but many fine books are available on any of these. It is the author's intent that by being exposed to the tip-of-the-iceberg, the curious reader will be inspired to explore the vast wonderland below.

As much as possible, the author has tried to implement Marvin Minsky's dictum that "You don't understand anything until you learn it more than one way." Indeed, several important notions are presented from multiple points of view and throughout the text there are several recurrent themes, such as point attractors, periodic cycles, and chaos, among others.

One *caveat* concerning the applications of cellular automata. We aretex not making any claims that CA models are necessarily superior to other kinds of models or that they are even justified in every case. We are merely presenting them as one way of looking at the world which in some instances can be beneficial to the understanding of natural phenomena. At the very least, I think you will find them interesting. For those who already have a passing knowledge of the subject, I think you will discover some new things you have not seen before. Even if the entire universe is not one monolithic cellular automaton, as at least

one scientist believes, the journey to understanding that point of view is well worth the price of admission.

Finally, I wish to thank Auckland University students Michael Brough, Peter Lane, Malcolm Walsh, as well as Dylan Hirsch-Shell of UCLA, and Nick Dudley Ward of Pattle Delamore Partners Ltd, who produced many of the figures from the CA models given in the text, and Samuel Dillon who produced the Rule 30 data encryption figures. Their assistance has been invaluable as their programming skills far exceed my own. I also wish to thank my daughter-in-law Yuka Schiff for many of the fine graphics and my friend Michael Parish for introducing me to the fascinating world of bees and Maeterlinck's classic monograph. A large debt of gratitude is owed to those who made valuable contributions to the manuscript in one form or another: Eshel Ben-Jacob, Michael Brough, Carlos Gershenson, Mario Giacobini, Dylan Hirsch-Shell, Jacques Mazoyer, Marc Ratkovic, Aaron Schiff, Birgitt Schönfisch, Dror Speiser, Guillaume Theyssier, and Jean-Baptiste Yunès. A very special thanks to Amy Hendrickson of TEXnology, Inc. for doing such an excellent and painstaking job laying up the text and images.

Several of the CA images were produced with the specialized software of Stephen Wolfram's *New Kind of Science Explorer*, which can be purchased from the website `http://www.wolframscience.com/`, and Mirek Wojtowicz's *MCell* program, which can be downloaded at his website `http://www.mirekw.com/ca/`. The latter also permits the real-time viewing of the evolution of some of the CA discussed in the text. Another interesting CA simulator is hosted by Michael Creutz at `http://quark.phy.bnl.gov/www/xtoys/xtoys.html`. Definitely have a look at Andy Wuensche's stunning website, `http://www.ddlab.com/`, and David Griffeath's Primordial Soup Kitchen, `http://psoup.math.wisc.edu/welcome.html`. An excellent CA resource website now edited by Tim Tyer is at `http://cafaq.com/`.

A website specifically for this text has been set up by John Wiley & Sons at ftp://ftp.wiley.com/public/sci_tech_med/cellular_automata/. Here there are further examples that have not been included in the text, Java applets, CA computer code, as well as a venue for interested readers to send in their own experimental contributions to the subject.

JOEL L. SCHIFF
University of Auckland

July 2007

Permissions

Figure 6.10	Reprinted from *Physica A* **284**, B. Luque and R. Solé, Lyapunov exponents in random Boolean networks, 33–45, Copyright © 2000, with permission from Elsevier.
Figure 6.12	Resnick, Mitchel, *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*, Figure 3.3. Copyright © 1994 Massachusetts Institute of Technology, by permission of The MIT Press.
Figure 6.13	Image courtesy Eshel Ben-Jacob, University of Tel Aviv.
Figure 6.16	Image from the website of Anahi Gajardo, `http://www.ing-mat.udec.cl/~anahi/langton/general.html`.
Figure 6.27	Redrawn from *Biosystems* **43**, M. Dorigo and L. M. Gambardella, Ant colonies for the travelling salesman problem, 73–81, Copyright © 1997, with permission from Elsevier.
Figure C.4	Image courtesy of Jacques Mazoyer, ENS-Lyon.
Figure C.6	Image courtesy of Birgit Schönfisch, University of Tübingen.
Figure C.10 and Figure C.12	Reprinted with permission of Christoph Hauert, from the *VirtualLabs* website `http://www.univie.ac.at/virtuallabs/`.
Figure C.11	Reprinted with permission of Serge Helfrich, from the website `http://prisonersdilemma.groenefee.nl/`.
Figure C.13	The code is courtesy of G. W. Flake (*The Computational Beauty of Nature*).
Figure C.14	Image generated by the *Wa-Tor* predator-prey program of Kovach Computing Services, `http://www.kovcomp.co.uk/`.
Figure C.15 and Figure C.16	Reprinted from *Chaos, Solitons & Fractals* **23**, C. Reiter, A local cellular model for snow, 1111–1119, Copyright ©2004, with permission from Elsevier.
Figure C.19	Reprinted with permission of Andy Wuensche, from the website `http://www.ddlab.org/`.
Cover art	Courtesy of Michael Brough, University of Auckland.

CHAPTER 1

PRELIMINARIES

Nothing shocks me. I'm a scientist.
–Harrison Ford (as Indiana Jones*)*

1.1 SELF-REPLICATING MACHINES

The origins of cellular automata can be traced back to mathematician John von Neumann's attempt to create a self-replicating machine. In 1948, von Neumann read a paper at the Hixon Symposium in Pasadena, California ("The general and logical theory of automata"), in which he outlined, among other things, a plan for a self-replicating machine (von Neumann actually refers to such a machine as an *automaton*). The question von Neumann addresses is this: "Can one build an aggregate out of such elements in such a manner that if it is put into a reservoir, in which float all these elements, each of which will at the end turn out to be another automaton exactly like the original one?" He then proceeds

Cellular Automata: A Discrete View of the World. By Joel L. Schiff

to outline the following argument to show that this is entirely feasible in principle.

One starts with a machine (universal constructor) A that has the ability to construct *any* other machine once it is furnished with a set of instructions denoted by I. Machine A is envisaged to float in the reservoir of liquid with all the necessary component parts that it requires for any particular construction. We now attach to our machine A another component called B that can make a copy of any instruction that is supplied to it. A final component, labeled C, von Neumann called the "control mechanism," which has the functions of initiating A to construct a machine as described by the instructions I and then cause B to make a copy of the instructions I and supply the copy of the instructions to the machine newly formed by A. Then the entire apparatus can be denoted by $M = A + B + C$.

To get things rolling, we furnish a machine M with a set of instructions for constructing itself, I_M, and call the resulting system M'. It is this machine, M', that is capable of replicating itself. For, C initiates the construction of M by A, it then has B make a copy of the instructions I_M and these are furnished to M to form the system M' once again. And so on.

It is the multiple use of the instruction set I_M that is crucial here. First, the instructions must be followed by A, second, they must be copied by B, and last, the copy must be attached to the machine constructed by A.

Overall, the copying mechanism is similar to the replication of living cells whereby the DNA (instructions) are first copied by cells preceding cell division. Interestingly, Christopher Langton [1986] comments: "Since he [von Neumann] was able to demonstrate that such a machine can exist, it becomes plausible that many, perhaps all, of the processes upon which life is based are algorithmically describable and that, therefore, life itself is achievable by machines." The study and implementation of these processes have become the domain of the newly emerging subject of *artificial life* and we shall encounter many instances of it throughout this book.

Von Neumann had now shown that a self-replicating machine was established in principle, but at the time of his lecture, he did not suggest how one could be implemented. The technology of the day simply was not capable of such an implementation.

According to received wisdom, it was the Polish-American mathematician Stanislaw Ulam who suggested to von Neumann that he should try constructing his self-replicating automaton using the conceptual framework of what are now known as cellular automata. The resulting system outlined in the early 1950s and later completed after von Neumann's death by Arthur Burks (see von Neumann [1966]) was a universal Turing machine embedded in a two-dimensional cellular lattice that had 29 states for each cell and a 5-cell neighborhood (now known as a von Neumann neighborhood) that required ~200,000 cells. However, it was never actually implemented.

Exactly what is meant by *dimension* and *states* will be dealt with in the sequel.

A simpler 8-state self-replicating cellular automaton was created by Codd [1968] with some computer assistance. Then in 1984, Christopher Langton demonstrated self-reproduction in an 86-cell looped pathway using 8 states with a 5-cell neighborhood, which did not exhibit the feature of universal construction as did the von Neumann and Codd machines, but simply reproduced itself. Langton's loop has a construction arm attached to it and consists of an outer sheath of cells that remain in a fixed state and an inner sequence of "DNA" cells in various states that circulate around the loop (Fig. C.1, see color section). At the junction of the loop and arm, the DNA cells are replicated: One copy goes back around the loop and the other copy travels down the construction arm where it is translated at the tip of the arm spawning new growth.

Once a side of the offspring loop is fully generated, the growth pattern makes a left turn and propagates another side and so on until the offspring loop is complete. Then the connection between parent and offspring is severed (cutting the umbilical cord so to speak) and both parent and offspring propagate separate construction arms to begin the process anew (Fig. C.2, see color section).

Construction continues in this fashion with each new loop generating at least one new offspring. When a loop tries to extend an arm into a region already occupied, it will retract the arm and the DNA of that loop is erased and the loop becomes inert. The Langton loops will continue this replication process indefinitely expanding outward with time and filling the plane (Fig. C.3, see color section).

Although each loop contains the same DNA sequence, the number of times it can replicate itself will depend on the space available in its immediate environment.

A somewhat simpler self-replicating loop that dispenses with the outer sheath but also having eight states was constructed by Reggia *et al.* [1993]. A simple and brief proof of the existence of a self-replicating CA machine capable of universal computation was given by A. R. Smith [1991]. Another approach was taken by Morita and Imai [1997], who devised cellular configurations that were able to reproduce by self-inspection rather than from any stored self-description.

In the realm of actual self-reproducing machines, a primitive form was demonstrated by Roger Penrose (the well-known physicist) and his father Lionel back in 1957 using a set of flat shaped wooden blocks that produced copies of a particular coupling when the blocks were shaken in a box enclosure. In 2001, Greg Chirikjian of Johns Hopkins University developed a LEGO robot that drove around a track and assembled modules to make a copy of itself. Recently, Hod Lipson and colleagues (Zykov *et al.* [2005]) at Cornell University have created a self-replicating robot consisting of a tower of cubes that can swivel around and pick up other cubes and stack them to create another tower identical to itself. According to Lipson, this opens up the possibility of using robotic systems in future space travel that can repair themselves.

An overview of 50 years of research on self-replication can be found in the article by M. Sipper [1998], who also created an interactive self-replicator (Stauffer and Sipper [2002]). Certainly the notion of self-replication has proved enormously popular with the creators of computer viruses.

1.2 GRAND TURING MACHINES

In his seminal 1936 paper on computable numbers ("On computable numbers, with an application to the Entscheidungsproblem"), English genius Alan Turing discussed a very general type of computer that has become known as a Turing machine. This machine is theoretical in nature and today still finds applications in computer science. In the words of computer scientist Edward Fredkin, "It was a way to formalize all the things that a mathematician could do with a pencil and paper."

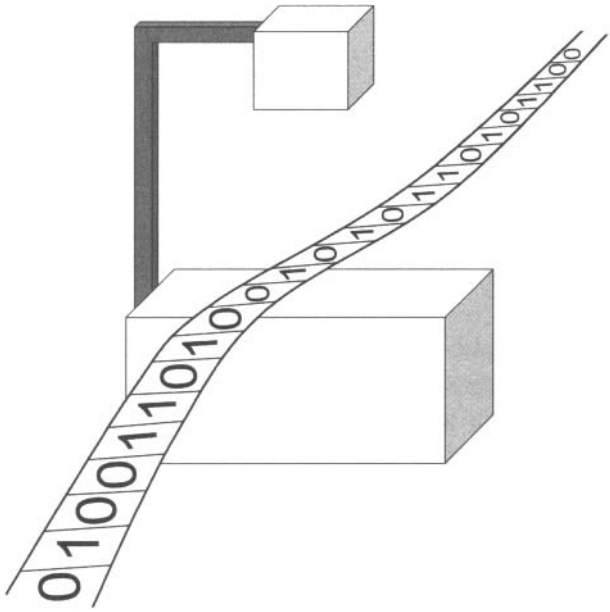

Figure 1.1 An idealized version of a Turing machine with the head reading the input of the tape below. The entire mechanism can move one square to either the right or left.

It can be thought of as a mechanical "head" that has an infinite strip of tape in both directions that lies beneath it. The head can both read and write onto the tape. The head is allowed to exist in a finite number of internal states, say k. The state of the head changes by interacting with the input from the tape beneath it. The tape is divided into an endless succession of square cells in which either a number 1 is written or the cell is blank, which we interpret as being the number 0. While the tape is infinite in length, there can only be a finite number of cells containing the number 1. The remaining cells must all be blank. The head reads just one such cell at a time — the one directly beneath it. Upon the head reading the value 0 or 1 of this cell, it replaces this value either with a 1 or 0 or with the same symbol. The head then moves one square either to the right or to the left (or not at all) and goes into one of its other allowable states. It then repeats the above sequence for each cycle of the machine. The new state of the head, the value the head gives to a particular cell, and the movement of the head left or right

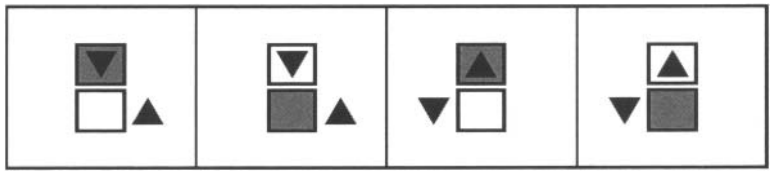

Figure 1.2 Graphical form of the Turing machine state transition function given in the text, denoting Head State 1 = ▼ and Head State 2 = ▲.

are all governed by some underlying set of instructions — the *state transition function.* There are also special starting and halting states. The output is written on a portion of the tape that can be read by an observer after a halt state has been reached (Fig. 1.1).

As an example, let us assume that the head has just two allowable states 1 and 2, with a state transition function given by the format

(**H**ead **S**tate, **C**ell **S**tate) $\longrightarrow$ (**H**ead **S**tate, **C**ell **S**tate, **M**ove)

specifically, say:

$$\begin{aligned} (1,1) &\longrightarrow (2,0,1);\ (1,0) \longrightarrow (2,1,1); \\ (2,1) &\longrightarrow (1,0,-1);\ (2,0) \longrightarrow (1,1,-1). \end{aligned}$$

These four rules could also be depicted in the graphical form of Fig. 1.2, where Head State 1 = ▼ and Head State 2 = ▲.

So, the first cell of Fig. 1.2 illustrates the first rule: $(1,1) \longrightarrow (2,0,1)$; meaning, if HS = 1 (i.e., ▼) and the head is on a cell with CS = 1 (gray), then HS becomes 2 (i.e., ▲), the CS is changed to 0 (white) and the head moves 1 cell to the right along the tape, and so forth with each rule. We can represent the evolution of this Turing machine by letting each step be depicted in a vertical downward direction with moves of $+1$ being to the right and -1 being to the left. Thus the above Turing machine would evolve as in Fig. 1.3 from an initial condition HS = 1, CS = 0.

Via the preceding rules, $(1,0) \longrightarrow (2,1,1)$, so that the HS is changed to 2, the initial cell state is changed to CS = 1, and the head moves one cell to the right. This state of affairs is denoted on the second line down, where we have the initial cell in state 1 (gray), and the head (in HS = 2) has moved over one cell to the right, over a cell in state 0 (white). Next

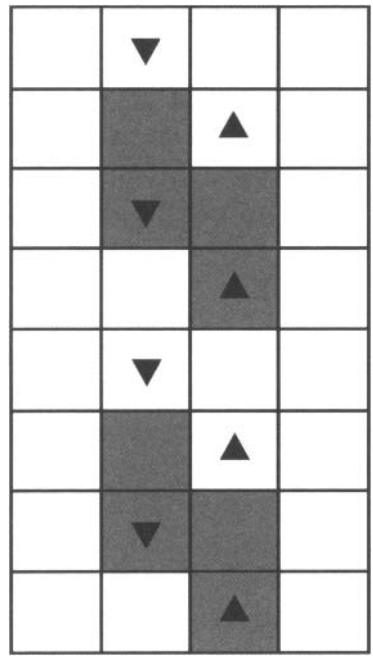

Figure 1.3 The time evolution (top to bottom) of the Turing machine example in the text starting with an initial condition HS = 1, CS = 0.

we find that $(2, 0) \longrightarrow (1, 1, -1)$, meaning that the HS now becomes 1, the CS becomes 1, and the head moves one cell to the left, directly over the initial cell (which was gray from before). Now HS = 1 and CS = 1, so we use $(1, 1) \longrightarrow (2, 0, 1)$, which alters the head to HS = 2, the cell state changes to CS = 0, and the head moves one cell to the right again (which also was gray from before). And merrily on the head goes, in this simple case never moving beyond the first two cells, and continually repeating its actions in a four-step cycle.

One question that one may ask about any particular Turing machine is, given a particular input, will the machine stop? This general question is known as the "halting problem" and it was shown by Turing himself that there is no way in principle to decide whether any particular Turing machine will stop or not. It may be noted that this result has ramifications to the whole field of mathematics itself.

Many mathematical questions can be posed in the language of whether or not a specific Turing machine will halt or not. For example, Opperman's conjecture states that for any integer $n > 1$, between the numbers

n^2 and $(n+1)^2$ one can always find a prime number. For example, between $3^2 = 9$ and $4^2 = 16$ lies a prime, in fact two in this case, 11 and 13. As the integers n start to get larger and larger, the number of primes starts to thin out so there is always a possibility that there will not be sufficient numbers of primes to fit between every n^2 and $(n+1)^2$. Fortunately, n^2 and $(n+1)^2$ spread out too. In spite of the elementary nature of this conjecture, no one has been able to prove or disprove it.

We could set up a Turing machine that would check to see if there was indeed a prime between every pair of numbers n^2 and $(n+1)^2$, and we could instruct the machine to stop if for some pair a prime was not found. If the Turing machine does stop, then we have produced a counterexample to the Opperman conjecture and it is false. If somehow we knew that this Turing machine never stopped, then we would know that the Opperman conjecture was indeed true. But being able to solve the halting problem for this particular Turing machine is equivalent to determining the truth or falsity of the Opperman conjecture.

This means that there can be no general algorithm for deciding the truth or falsity of mathematical problems, which was the Entscheidungsproblem in the title of Turing's paper. This problem was first enunciated by the famous German mathematician, David Hilbert, at the 1900 International Congress of Mathematicians and was included in a list of 23 mathematical problems to be considered over the ensuing new century. This problem asked whether or not there was some general mechanical procedure that would be able to determine the truth or falsity of a large body of well-defined mathematical problems such as the Opperman conjecture. And as we have just seen, Turing showed this was not possible.

Turing machines with different sets of instructions are capable of doing different tasks, while there are some that are capable of emulating the performance of any other Turing machine. These are *universal Turing machines* and are said to be capable of performing *universal computation*. They can in fact do any calculation that is computable. Actually, the ubiquitous personal computer is effectively a universal Turing machine. Although a personal computer does not have an infinite storage capacity, it is essentially large enough to be considered so. Until recently, the simplest universal Turing machine was due to Marvin Minsky, who in 1962 produced a universal Turing machine with seven head states and four cell states. Based on

the universality of Rule 110 (Section 3.5), Wolfram [2002] reduced this to just two head states and five cell states. In May 2007, Wolfram offered a $25,000 prize to anyone demonstrating that a particular two head state/three cell state Turing machine is universal (see `http://www.wolframscience.com/prizes/tm23/` for details).

Other work of Alan Turing's is encountered in Chapter 5 regarding reaction-diffusion equations, but we mention here just one further product of Turing's fertile mind — the *Turing test* — first proposed in 1950, although originally referred to by Turing as the *imitation game*. In this test, there is a human being in one room and a computer in another. Both communicate with those outside the room via some electronic device such as a teletype. An interrogator asks questions of the inhabitants of both rooms and tries to decide which is the human and which is the computer. If the interrogator, after a given period, is unable to determine which is which, the computer is said to have passed the Turing test. This is a very straightforward test of "machine intelligence" and in spite of many criticisms and objections over the years, it has stood the test of time very well. What it is not is a test of "machine consciousness" and this complex issue will be discussed further in Chapter 6.

Turing himself believed that in 50 years time an average interrogator after 5 minutes of questioning would have no more than a 70% chance of deciding which was the computer and which was the person. Nevertheless, as of the present date, no computer has even come close to passing the Turing test. In 1966, Joseph Weizenbaum of MIT created a computer program called ELIZA (after G. B. Shaw's Eliza Doolittle character of *Pygmalion* fame) that simulated the responses of a Rogerian psychotherapist. Some administrative staff members actually consulted ELIZA concerning their everyday problems and even the simple-minded Apple II version had many of this author's friends conversing intelligently with ELIZA for a few minutes. Readers are welcome to discuss their psychological problems with ELIZA at `http://www.manifestation.com/neurotoys/eliza.php3`.

1.3 REGISTER MACHINES

Computers basically perform operations on numbers that are stored in what are called *registers*. A (Minsky) register machine is just an idealization of the basic manipulations that are performed on the num-

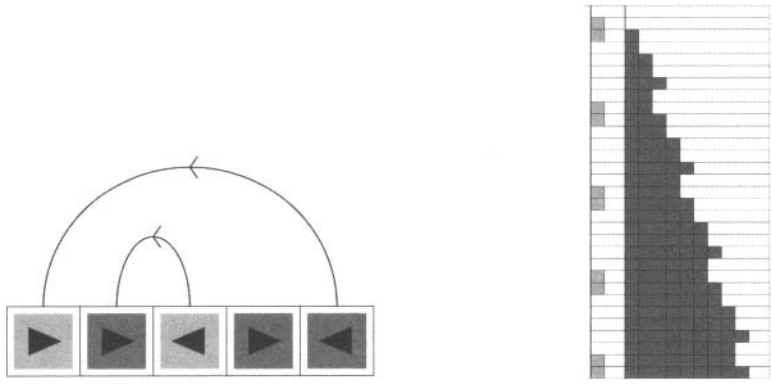

Figure 1.4 An example of a simple register machine with two registers and five instructions. Light gray indicates register 1 and dark gray register 2. An increment instruction is denoted by ▶ and a decrement by ◀ together with the path of the jump (arrowed). The output after each step is at the right with the first column displaying the contents of register 1 and the second column the contents of register 2. Both registers are initially empty (top line right) and the instructions begin with the incrementing of register 1 (first box left). Note the cyclic nature of the contents of register 1.

bers contained within the registers of a real computer. In order for the computer to perform a simple calculation such as the addition of two numbers, it must take the numbers stored in one of the registers and combine it with the number stored in another register. A register machine has three types of instructions: INC(rement), DEC(rement) and HALT. The INC instruction increments by 1 the number stored in a particular register and then the machine proceeds to process the next instruction. The DEC instruction has two components: it decrements by 1 the number stored in a particular register and then it will "jump" to another specifically designated instruction. But there is one *caveat* here. The number in a register cannot be less than zero, so if the value of zero is stored in a register and that register is to be decremented, then the instruction is ignored and the machine proceeds to the next instruction. Finally, the HALT instruction does simply that — it halts the operation of the register machine (Fig. 1.4).

Computer scientist Marvin Minsky [1967] showed that all one needs is just two registers to emulate a universal Turing machine.

The reader can create their own Turing machines and register machines and investigate their behavior using Wolfram's *NKS Explorer*.