SUBRATA DASGUPTA

IT BEGAN WITH BABBAGE

THE GENESIS OF COMPUTER SCIENCE

It Began with Babbage

THE GENESIS OF COMPUTER SCIENCE

Subrata Dasgupta

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

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.

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 Poland Portugal Singapore
South Korea Switzerland Thailand Turkey Ukraine Vietnam

Oxford is a registered trademark of Oxford University Press
in the UK and certain other countries.

Published in the United States of America by
Oxford University Press
198 Madison Avenue, New York, NY 10016

Library of Congress Cataloging-in-Publication Data
Dasgupta, Subrata.
It began with Babbage : the genesis of computer science / Subrata Dasgupta.
pages cm
Includes bibliographical references.
ISBN 978-0-19-930941-2 (alk. paper)
1. Computer science—History—19th century. 2. Computer science—History—20th century.
I. Title. II. Title: Genesis of computer science, 1819–1969.
QA76.17.D36 2014
004.09—dc23
2013023202

9 8 7 6 5 4 3 2 1
Printed in the United States of America
on acid-free paper

IT BEGAN WITH BABBAGE

To Amiya Kumar Bagchi

Acknowledgments

IN RETROSPECT, I now realize that this book on the genesis of computer science had its own genesis somewhere in my subconscious in summer 1977, when I was a visiting scientist in the Cambridge University Computer Laboratory. There, I had my first of many meetings with Maurice Wilkes, and it is also where I met David Wheeler. Twenty-eight years earlier, Wilkes, Wheeler, and their colleagues had designed and built the EDSAC, the world's first fully operational stored-program computer. As a graduate student, I had become seriously interested in the history and cognitive nature of scientific creativity, but computer science seemed different. For one thing, unlike physics, chemistry, or biology, the history of which stretched back centuries, the originators of computer science were, mostly, still living and active. When talking to Wilkes and Wheeler that summer, two pioneers, I was privy to a kind of oral history of our discipline.

Throughout the succeeding decades, my interest in the history and cognitive nature of science meandered in different directions, but I continued to think about computer science. Under the influence of Herbert Simon's *The Sciences of the Artificial*, I came to glimpse something about the nature of the discipline. I remember a conversation with B. Chandrasekaran of Ohio State University sometime during the early 1980s when we were, briefly, colleagues. Chandra said that computer science still had no intellectual tradition in the manner of physics or mathematics or chemistry. This obscurely disturbed me, but other preoccupations both within and outside computer science prevented me from pursuing this issue which, it seemed, required serious inquiry.

In November 2010, Wilkes passed away, and his death prompted me to give a lecture about him titled "The Mind of a Computer Pioneer" in the Center for Advanced Computer Studies at my university. After the talk, some graduate students approached

me to offer a seminar course on the birth of computer science. An early draft of this book formed the basis for the course, which I taught in Fall 2011.

So I am indebted to Duane Huval, Charles LeDoux, Craig Miles, and the late Michael Sharkey for stimulating me in writing this book. They were the "first responders" to its contents. Their feedback was most valuable.

I have had the good fortune of many conversations with Maurice Wilkes, and conversations and lengthy e-mail discussions with the Herbert Simon, a polymath for our times on the origins of artificial intelligence. I was also privileged to know Donald Cardwell, historian of science and technology, when I was teaching at the University of Manchester Institute of Science and Technology during the early 1990s. These encounters have shaped my thinking, in different ways, on the historical, cognitive, and creative aspects of science, especially computer science. I have no doubt that their influence on me has found its way into this book.

I thank Simon Lavington, who made available to me a scarce, historical account of the Manchester University computers and for allowing me to read an unpublished paper by him on the Manchester machines. I thank D. F. Hartley for information on the academic status of computing during its earlier days in Cambridge.

Three anonymous reviewers, who read first drafts of the first few chapters on behalf of the publisher, offered very thoughtful comments. I thank them, as well.

Thank you, Jeremy Lewis, my editor at Oxford University Press who has been with this project from the day he received my first communication. It has been a pleasure working with him.

Thanks to Erik Hane, also of the editorial staff at Oxford University Press for all his help.

Thank you Terry Grow, a young artist who produced the images that appear in this book.

Bharathy Surya Prakash supervised all stages of the physical production of the book. Her professionalism was noteworthy. I thank her and her production team.

My thanks to Oxford University Press for giving me permission to adapt two diagrams (Figure 8.5, p. 135; Figure 8.7, p. 147) from *Technology and Creativity* (1996) authored by me.

Finally, as always, thank you Mithu, Deep, and Shome.

IT BEGAN WITH BABBAGE

Contents

Prologue

1

IN 1819, A young English mathematician named Charles Babbage (1791–1871) began to design a machine, the purpose of which was to compute and produce, fully of its own steam, certain kinds of mathematical tables. Thus came into being the idea of *automatic computation*—performing computations without human intervention—and an intellectual tradition that eventually gave birth to a brand new and very curious scientific discipline that, during the late 1960s, came to be called *computer science*. This book tells the story of its genesis, a long birth process spanning some 150 years, beginning with Babbage and his dream of automatic computation.

The focus of every science (in fact, every intellectual discipline) is a certain kind of reality, a certain class of phenomena. The focus of computer science is the phenomenon called *computation*, which refers both to a concept and an activity that is associated historically with human thinking of a certain kind. The Latin root of the English word *compute* is *computare*—meaning, reckoning, calculating, figuring out. Thus, according to etymology, computation refers to the idea and the act of reckoning or calculating.

Etymologically, then, computation's domain would seem to be the realm of numbers. However, as we will see, we have come a long way from this association. We will see that the domain of computation actually comprises *symbols*—by which I mean *things that represent other things* (for example, a string of alphabetic characters—a word—that represent some object in the world, or a graphical road sign that represents a warning to motorists). The act of computation is, then, *symbol processing*: the manipulation and transformation of symbols. Numbers are just one kind of symbol; calculating is just one kind of symbol processing. And so, the focus of automatic computation, Babbage's original dream,

is whether or how this human mental activity of symbol processing can be performed by (outsourced to) machines with minimal human intervention. Computer science as the science of automatic computation is also the science of automatic symbol processing.

II

However, computer science is not a *natural* science. It is not of the same kind as, say, physics, chemistry, biology, or astronomy. The gazes of these sciences are directed toward the natural world. In contrast, the domain of computer science is the artificial world, the world of made objects, *artifacts*—in particular, ones that perform computations. Let us call these *computational artifacts*.

Now, the natural scientist, when practicing her science, is concerned with the world *as it is*. As a scientist she is not in the business of deliberately changing the world. The astronomer looking through a telescope at the galaxies does not desire to change the universe but to understand it, explain it; the paleontologist examining rock layers in search of fossils is doing so to know more about the history of life on earth, not to change the earth (or life) itself. For the natural scientist, to understand the natural world is an end in itself. The desire is to make nature *intelligible*.[1] The computer scientist also wishes to understand, although not through nature but through computational artifacts; however, that wish is a means to an end, for she wants to *alter* the world in some aspects by creating new computational artifacts as improvements on existing ones, or by creating ones that have never existed before. If the natural scientist is concerned with the world *as it is*, the computer scientist obsesses with the world as she thinks *it ought to be.*

This, of course, highlights the venerable philosophical distinction between *is* and *ought*. We might say that computer science is a science of the *ought* in contrast to a natural science such as evolutionary biology, which is a science of the *is*.

III

Computer science is not unique because of this "oughtness," nor does its curious nature lie in this. In 1969, the polymath scientist and economics Nobel laureate Herbert Simon (1916–2001) pointed out that the main characteristic of artifacts is that they come into existence with a *purpose* and, consequently, the sciences that deal with artifacts—in his term, the "sciences of the artificial[2]"—are concerned with purpose (or goals), and in this sense they stand apart from the natural sciences. The objects of nature have no purpose. They just *are*. We don't ask about the purpose of the moon or the stars, of rocks or fossils, of oxygen or nitrogen. They just exist. It is true that anatomists and physiologists ask questions about the *function* of a particular organ or process in a living organism, but such functions are attributes that belong to some organ or life process as an outcome of natural

evolution. They do not signify some *prior* purpose originating in the mind of a creative being. Artifacts, in contrast, have prior reasons for existence, reasons that were lodged in human minds prior to the beginning of artifact making. Thus, the sciences of the artificial must concern themselves with the characteristics of artifacts as they are related to the purposes *as intended for them by their (earthly) creators.* The structure and behavior of an artifact is meaningful only in respect to its purpose. Artifacts are *imbued* with purpose, reflecting the purposes or goals imagined for them by their human creators.

This is why a material artifact can never be explained solely in terms of natural laws even though the artifact must obey such laws. To explain or understand an artifact, even something as apparently simple as a piece of pottery, one must ask: What is it for? What does it do? What was the potter's intention?

This is why a computational artifact such as one's laptop can never be explained only by the laws of physics, even though the laptop's circuits and hard drive obey such laws. A computational artifact is intended to serve some purpose, and physics has nothing to say about purpose. *Computer science is a science of the artificial.* It must, therefore, embody principles, laws, theories, models, and so forth, that allow an explanation of how its structure and behavior relate to intended goals.

Computer science, then, involves the human mind in two ways. First, as we have noted, it is concerned with how artifacts can perform the mental activity of symbol processing. Second, as a science of the artificial, it must have a place in it for the minds of the human creators of computational artifacts—and how their imagined goals and purposes are transformed into artifactual forms.

IV

Of course, computer science is not the only science of the artificial. There are many disciplines that deal with the world of artifacts, that are concerned with changing the world to a preferred state, with pursuing the ought rather than the is. Some of them are of much earlier vintage than computer science. They include, for example, the traditional engineering disciplines—civil, mechanical, electrical, chemical, and metallurgical; they include architecture and industrial design. Others of more recent vintage include genetic engineering, biotechnology, and digital electronics design. Their concerns are, almost without exception, *material* artifacts: structures, machine tools, internal combustion engines, manufacturing and processing equipment, metals, alloys and plastics, aircraft, electronic systems, drugs, genetically modified organisms, and so forth.

Computer science stands apart because of the *peculiarity* of its artifacts. In fact, they are of three kinds.

They can be *material* objects—the physical computer system. These artifacts clearly resemble the material artifacts just mentioned because, like them, they obey the laws of physics and chemistry. They consume power, they generate heat, there is some physical

motion involved, they decay physically and chemically over time, they have material extension, they occupy physical space, and their activities consume physical time.

Computational artifacts, however, can also be completely *abstract*, existing only as symbol structures (made visible on physical media such as paper or the computer screen). They are intrinsically devoid of any physical meaning. The laws of physical nature do not apply to them. As we will see, things called algorithms and purely mathematical machines called Turing machines exemplify such artifacts. Because they are abstract, once created they exist for ever. "They neither spin nor toil" in the physical world. They occupy no physical space nor do their activities consume physical time; rather, they live in their own *abstract* space–time frame.

Abstract computational artifacts such as algorithms resemble mathematical symbol structures (for example, algebraic equations, geometric objects) except that mathematical artifacts have no space–time characteristics at all, neither physical nor abstract.

The third kind of computational artifact is arguably the most interesting and unique of all. These artifacts are *in between* the material and the abstract. They themselves are abstract, yet their existence and usefulness depend on an underlying material substrate. We will call these *liminal* artifacts.[3] Computer programs and the entities called computer architectures are prime instances of this category; they themselves are abstract, yet they must have underlying material computational artifacts as substrates to make them useful or usable (just as the mind needs the brain for its existence).

Computer science, thus, must deal with computational artifacts that straddle the material, the abstract, and the liminal. Each of these types of artifacts can be studied, analyzed, understood, explained, and created autonomously, just as the mind and the brain can be studied autonomously, but—as in the case of the mind and the brain—only up to a point, because these classes of artifacts form *symbiotic relationships*: the abstract with the liminal, the liminal with the material. In fact, as we will see, automatic computation involves the constant interplay between the abstract, the liminal, and the material.

All of this separates computer science from most other sciences of the artificial—what makes computer science so peculiar, so curious, so distinctive. Because of the abstract and liminal artifacts, the laws governing computational artifacts are, in part only, physical laws. In fact, in a certain sense, the laws of nature are almost marginal in computer science. It is the abstract and the liminal artifacts that have come to dominate computer science, and *their* laws are necessarily of an entirely different nature. This raises the issue: What *is* the nature of the *science* in computer science?

V

The answer, in detail, is the story this book will tell. But, we must pause here on the concept of science itself. The sciences of the artificial differ from the natural sciences because the latter is concerned with natural phenomena and the former with the world of artifacts; they differ because the former *must* factor in purpose into the discourse whereas