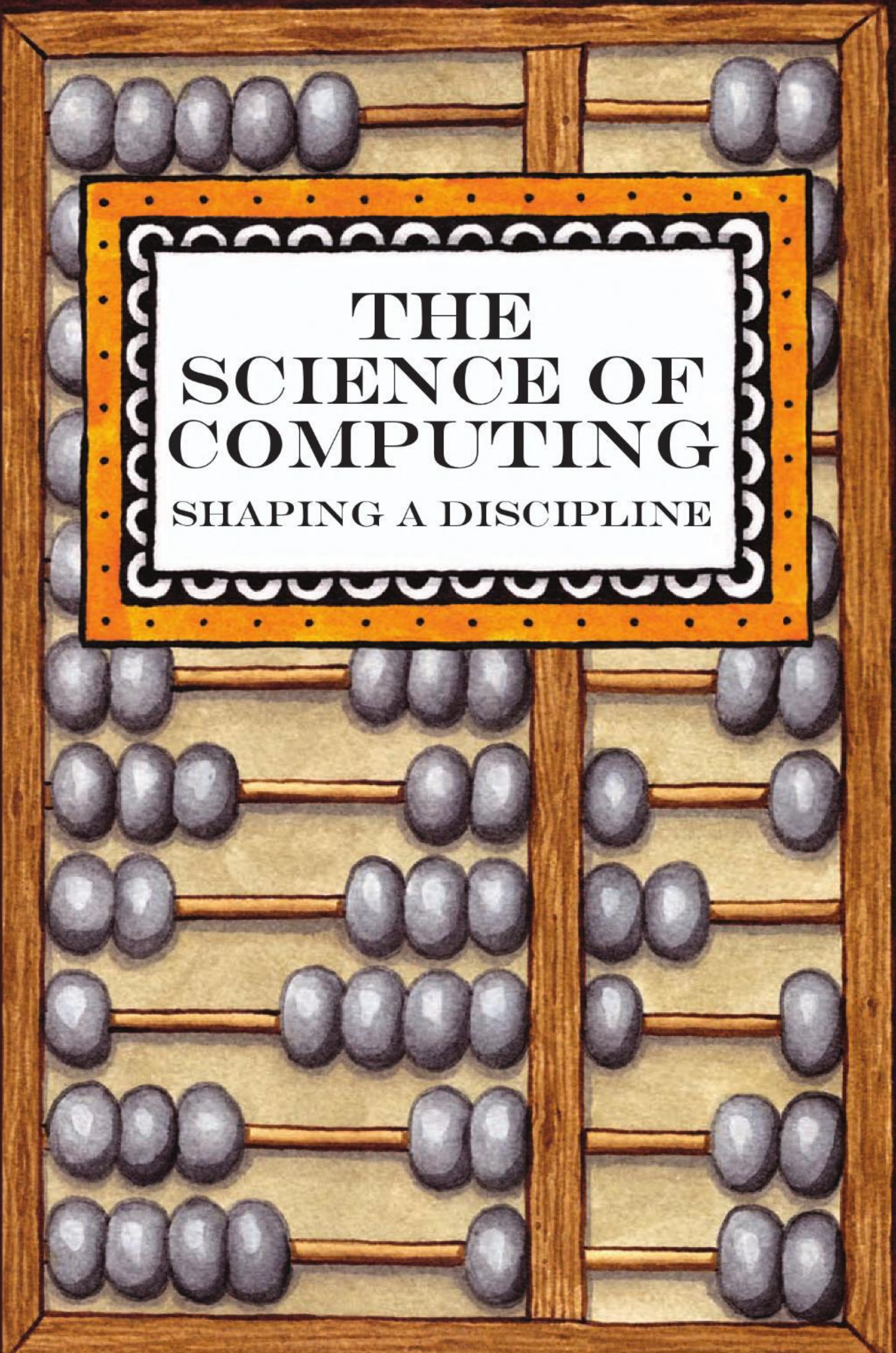
MATTI TEDRE



THE SCIENCE OF COMPUTING

SHAPING A DISCIPLINE

CRC Press
Taylor & Francis Group

A CHAPMAN & HALL BOOK

THE SCIENCE OF COMPUTING

SHAPING A DISCIPLINE

MATTI TEDRE, PH.D.

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20141023

International Standard Book Number-13: 978-1-4822-1769-8 (Paperback)

Library of Congress Cataloging-in-Publication Data

Tedre, Matti.
The science of computing : shaping a discipline / author, Matti Tedre.
pages cm
Includes bibliographical references and index.
ISBN 978-1-4822-1769-8 (pbk.)
1. Computer science. I. Title.

QA76.T4116 2014
004--dc23 2014039070

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

List of Figures

List of Tables

Preface

"That's not computer science," a professor told me when I abandoned the traditional computer science and software engineering study tracks to pursue computing topics that I thought to be more societally valuable. Very quickly I learned that the best way to respond to such remarks was with a series of counter questions about what exactly is computer science and why. The difficulties that many brilliant people had responding to those questions led me to suspect that there's something deeper about the topic, yet the more I read about it, the more confused I got. Over the years I've heard the same reason—"That's not computer science"—used to turn down tenure, to reject doctoral theses, and to decline funding. Eventually I became convinced that the nature of computing as a discipline is something worth studying and writing about.

Fortunately enough, the word "no" doesn't belong to the vocabulary of Professor Erkki Sutinen, who became my supervisor, mentor, colleague, and friend. Throughout my studies in his group I worked on a broad variety of applied computing topics, ranging from unconventional to eccentric, yet in the meanwhile Erkki encouraged me to continue to study computing's disciplinary identity, and I ended up writing, in a great rush, a thesis on the topic. When I moved from the University of Eastern Finland to Asia and then to Africa for the better half of a decade, I kept on writing small practice essays on computing's identity. And as decent journals kept on publishing those essays, I continued to work on the topic. Many years' worth of evenings in the quiet African town of Iringa, one full day's drive away from the bustling Dar es Salaam, gave me the time and mental space to finally read enough about what computing's pioneers over the years have said about computing. I continued that work at Stockholm University, where my department's management encourages the researchers to do whatever they want to do. That, and a nine-month research break in 2013, enabled me to put it all on paper. Eventually, the hardest part about writing this book was putting an end to it: I still have hundreds of bits and pieces that would amend the book in important ways. I guess that, like dissertations, books like this are never finished, but abandoned when stress grows unbearable.

Over the many years that it has taken to finish this book, I have accumulated a great debt of gratitude. During the past year, many people read parts of this manuscript at different stages of its development. I wish to thank Peter Denning for amending the story with his vision and experiences as well as for sharing many behind-the-scenes stories of events described in this book;

Jan van Leeuwen for pages after pages of insights and suggestions on improving the work; Gordana Dodig-Crnkovic for her enthusiastic support and careful reading of the manuscript; Edgar Daylight for pointing out problems and oversimplifications; Johannes Cronjé for research vocabulary; and Taylor & Francis's anonymous reviewer for a great many suggestions for improvement (and what must have been two penfuls of red ink). I wish to thank Jorma Sajaniemi and Viola Schiaffonati for many constructive comments on the manuscript, and Thomas Haigh for pointing out references that I should not miss. As always, although I received a great amount of help, I alone bear responsibility for incorrect interpretations and factual errors.

Over the years, numerous colleagues have shared their insights and ideas; colleagues with whom I worked at University of Eastern Finland, Ajou University in South Korea, Tumaini University in Tanzania, University of Pretoria and Cape Peninsula University of Technology in South Africa, and Stockholm University in Sweden. I apologize that I have included in the above list only those people who had direct influence on this particular manuscript and not those who have influenced my work or thinking in other ways. I particularly apologize to those colleagues whose articles have not received adequate attention from me during the writing of this book (especially Mikko, Henrik, and Jyri). I thank my closest gene pool for my existence and brotherhood. This work received funding from the Academy of Finland grant #132572, the Finnish Association of Non-fiction Writers, the Ella and Georg Ehrnrooth Foundation, and Kauppaneuvos Otto Malmin Lahjoitusrahasto. The artwork is by Lasse Meriläinen.

The greatest gratitude I owe is to Nella for the mornings, for the evenings, and for all the moments between.

Matti Tedre
Stockholm, Sweden

I

Introduction

CHAPTER 1

Introduction

COMPUTING is an ancient activity. There is evidence of computing, starting from counting and calculation, that dates back thousands of years. Many early civilizations developed their own, unique means for storing and processing numerical information, such as the Quipu of Incas and the Chinese counting rods.[1] Various tools and aids for computing—such as analog astronomical computers and mechanical computing machinery—have existed for millennia.[2] The modern computer is the latest addition to that continuum.

Computing as a discipline—whether one prefers to talk about computer science, informatics, algorithmics, or something else—is a much more recent phenomenon than abstract mathematical concepts or the practice of using mechanical aids for calculation. There is, however, no birth date for computing as a discipline. Looking for such a date, one could point out computational or algorithmic-like concepts from the 1800s, the 1600s, or perhaps 1800 BCE. But it would be untrue to say that George Boole in the 1800s, or Blaise Pascal in the 1600s, or the mathematicians of ancient Babylon around 1800 BCE were early computer scientists or that they worked in the discipline of computing. The discipline simply did not exist at the time. The mathematician and logician George Boole, for instance, was a professor of mathematics at Queen's College in Ireland.

The pieces of a new era in computing emerged in the early 1900s from developments on a broad front, most prominently electrical engineering and mathematical logic. In the 1930s, answers to David Hilbert's decision problem led to formal definitions of an algorithm.[3] Also in the 1930s, Boolean logic was connected with the design of digital circuits, and theoretical definitions of computability were presented. The turn of the 1940s saw the crossing of the Newton–Maxwell gap from mechanical and electromechanical computing—governed by Newton's laws of motion—to fully electronic devices—governed by Maxwell's laws of electromagnetism.[4] In and around the 1940s, visions of modern computing started to form from earlier and newer innovations, and later condensed into a hard core of computing knowledge. The now stable hard core of computing includes ideas like the formalization of computable functions, the idea that instructions can be encoded as data, the idea that instructions and data reside in the same memory storage, and the separation of memory, the processing unit(s), control unit, and input/output unit(s).

Indeed, the late 1940s and the 1950s must have been an exciting time to work in computing. The emergence of a new and electrifying field with tangible progress and a whole horizon of open problems, attracted young and talented researchers, as well as significant financial resources. The nascent field brought together people from various backgrounds, such as mathematics, physics, and electrical engineering, and this diversity fueled a rapid series of technological and theoretical breakthroughs. Diversity of viewpoints, immediate applications, and a bold pioneer attitude were among the driving forces of the theoretical and technical revolution in computing.

The disciplinary identity of computing professionals began to emerge around the same time as modern computing machinery, but there is no single birth date of computing as an independent discipline. The field gradually developed attributes of an independent discipline. Theoretical foundations—which were first conceived in terms of mathematical logic, and later as theory of computing, saw great advances starting from the 1930s. A number of central technological advances were made in the 1940s. The 1940s also saw the emergence of today's central academic and professional associations for computing. Conferences for computing machinery were around well before the advent of modern computers. The 1950s saw a number of major computing journals and magazines, and the 1960s computing departments, complete curricula for universities, and the first Ph.D. graduates. In the 1970s, a number of major funding agencies made computing a category of its own. By the 1990s the discipline had a rich and unique body of deep theorems and algorithms.[5] There is no longer doubt about computing researchers' ability to independently set and follow a unique research agenda for their trade.

Today computing is a broad, thriving topic for academic research. Computing fields and branches span from information technology and information systems to software engineering, theoretical computer science, and scientific computation. Technological aspects of computing are studied in computer engineering and electrical engineering. There are a vast array of computing branches that are related to each other to various degrees—take, for instance, branches like computer security, human-computer interaction, artificial intelligence, health informatics, and computability theory.

But over the years computing has grown so large that it is sometimes hard to comprehend the shape and size of the discipline as a whole. Hence, it is no wonder that there is no consensus on what computing as a discipline is actually comprised of. Asking ten computing researchers what computing as a discipline is will yield ten different answers. There are a number of concepts—take for instance, the Turing machine and the stored program concept[6]—whose fundamental significance for the field is now rarely disputed. But aside from some central concepts and innovations, there is considerable disagreement with many aspects of computing as a discipline: for instance, opinions diverge about proper methods, subjects of study, or proper curricula for computing fields.

Starting from the 1950s, computing professionals have presented a dizzying number of arguments concerning the essential features of computing as an academic discipline. After almost 60 years of debates, the field seems to be further away from a consensus than ever before. Moreover, most arguments make a strong case for their cause, and they are often based on an intuitively appealing idea of computing as an activity, body of knowledge, or principles. It is easy to get lost amongst the contradictory, yet compelling, arguments for computing as a discipline made by the field's pioneers. Some argue that computing is primarily a technical field that aims at cost-efficient solutions, and others argue that the field's important contributions are theoretical by nature. Yet another school of thought argues that computing is an empirical science of information processes that are found everywhere—numerous accounts of computing try to combine different aspects of computing into a singular comprehensive package.

So, the field's identity has been fiercely debated throughout its short history. But what are the debates about and what is at stake? Why have the debates not ceased over the 60- or 70-year history of the discipline? Why is it still so difficult to define computing as a discipline? What do people mean when they say that computing is a scientific, mathematical, or engineering discipline? More precisely, what is computing, the academic discipline, about?

This book tries to shed some light on those questions by presenting the reader with arguments and debates about the essence of computing as a discipline. The book puts those debates in a broader disciplinary context, clarifies their background, and analyzes the reasoning behind those debates. By doing so, the book aims at presenting a rich picture of computing as a discipline from the viewpoints of the field's champions. Although this book is written by a computing researcher for other people in computing disciplines, it does not require very deep knowledge of any specific branches of computing.

Viewpoint of This Book

This book emphasizes three viewpoints that are helpful for understanding the debates about computing as a discipline: contextual awareness of those debates, a broad perspective of the field, and tolerance towards different uses of terminology. First, in order to understand debates about computing as a discipline, it is crucial to understand the roots, and the context, of those debates. Many arguments about the nature of computing as a discipline are so deeply rooted in ages-old debates about computing that it is nigh impossible to appreciate them without a contextual frame of reference. For instance, a look behind the still-ongoing debates about the scientific nature of computing reveals a wide range of stimuli. Many computing pioneers were originally natural scientists, and academic and public prestige played an important role in university politics. Attracting students, staff, and funding were affected by the field's image, and there were issues with the burgeoning field's intellectual integrity and progress. Those aspects, and many others like them, shaped the course of discussions about computing's scientific nature.

Contextual understanding is also important because many famous quotes get a different twist in the context where they were first presented. Take, for instance, the oft-cited remarks of Edsger Dijkstra—a computing pioneer and visionary, and a master of metaphors and catchy one-liners. One popular quotation of Dijkstra is his comparison of calling the discipline "computer science" with calling surgery "knife science."[7] Dijkstra's remark was part of a decades-long debate between theoretical, scientific, and engineering approaches to computing. Those debates were fueled by the software crisis, which was manifest in overbudget, poor-quality, unmanageable, and overdue software projects. The crisis with the work force greatly affected academic computing, too. There was a lot to be unhappy about, and a lot of the blame was put on the sloppy practices of software producers. Dijkstra—a recognized practitioner himself—belonged at that time to an influential but at the time already diminishing group of theoretical purists regarding programming, he had his own vision for how computing should develop as a discipline, and his view was that technology is contingent, while formal, abstract theoretical knowledge is of lasting value.

Second, in addition to contextual understanding, it is important to appreciate the breadth of the field. The diversity of the discipline and its dizzying variety of applications have been some of the main driving forces of the field's development. During the past sixty years, computing researchers have brought together a wide variety of scientific disciplines and methodological standpoints. The resulting discipline has its own distinct body of knowledge, but its body of knowledge also intertwines with knowledge from many other fields and it offers a variety of unique means of modeling and simulating phenomena in other fields. The expansion of computational and algorithmic models—"the idiom of modern science"[8]—to all other fields has been dubbed the "algorithmization" of the sciences.[9] The increased investments in research efforts in computing have been paralleled by the growth of the number of branches of computing, such as scientific computation, artificial intelligence, decision support systems, architectural design, and software engineering. Arguments about the content of the field, its methods, and its aims have sometimes been fierce, and the rapid pace of extension of the field has made it even harder to define computing as an intellectual activity faithfully to what happens in the laboratories, offices, and garages.

Although interdisciplinarity made the rapid development of computing possible in the first place, it also gave rise to very real challenges. For example, there never was an agreement over what kinds of topics should be included in the discipline, and it was very difficult to come up with a common understanding of how research in computing should ideally be done. If a generic set of rules for quality research in all of computing were formulated, those rules should cover research in fields such as software engineering, computational complexity theory, usability, the psychology of programming, management information systems, virtual reality, and architectural design. The subjects that computing researchers study can be, for instance, programs, algorithms, logic, programmers, machines, usability, or complex systems. Consequently, there was considerable disagreement about what new generations of computing graduates should be taught. Attempts to describe computing as a discipline have invariably been either so broad that they do not exclude much, or very narrow and applicable to only some subfields of computing. It is debatable whether an overarching, all-inclusive definition of computing as a discipline is even necessary, or if computing stands out as an example of a postdisciplinary era of science.

Interdisciplinarity has also fueled methodological debates that have haunted computing's disciplinary identity from early on. Given the broad variety of computing fields, a single set of methods and approaches cannot be used with the whole variety of subjects that researchers in computing fields study. Mathematical and formal models are precise and unambiguous, yet they are confined to the world of abstractions and they fail to fully capture the unbounded richness of the physical world and meaning and significance in the social world. Narratives and ethnographies are rich in dimensions and sensitive to detail, yet they are equivocal and context-dependent. Narratives have little predictive power, and formal proofs have little explanatory power regarding things like usability preferences and much of human experience in general. Then again, the predictive power of mathematical and computational formulations is uncanny: computational models have a miraculous, "unreasonably effective"[10] capability of accurately predicting things in seemingly unrelated domains. Simulations continue to pervade other academic disciplines and change them. Still today, computing's disciplinary debates abound with arguments about the merits of scientific experiments, formal methods, and engineering methods, to name a few.

Third, a final difficulty in understanding disciplinary debates about computing is that the debaters use the same words to mean different things and different words to mean the same thing. The very phrase "computer science" is a bone of contention as such. Many arguments about how "computer scientists" *should* work have their roots in different conceptions about what "computer science" *is*. Some have used the phrase "computer science" to refer to a specific field of academic computing, while others have used it as an umbrella term for computing's myriad topics. Trying to come to grips with controversies between scholars from different branches of computing sometimes requires open-mindedness toward views that arise from different backgrounds of scholars and their fields of research. Arguments about the essence of computing are rarely uninformed or naïve, yet fully appreciating them often does require insight into the intellectual background from which they are made and meanings they have adopted.

In this book, the terms "computing disciplines" and "computing as a discipline" are used as umbrella terms for the academic fields that belong to the computing family—such as information technology, information systems, scientific computing, and computer science. Also located under that umbrella are computing's engineering fields: computer engineering, which emerged from electrical engineering and deals specifically with computer hardware, and software engineering.[11] "Computer science" refers to one of the computing fields, and it has branches like artificial intelligence, computational complexity theory, and formal languages. These distinctions between computing as a discipline, its fields, and their branches are, however, muddied by historical uses of terminology—many pioneers of computing used "computer science" to refer to all of computing. In many places in this book, the authors' original use of terminology is followed in quotations and their immediate context.

This book presents a broad variety of views on computing as an academic discipline. Those views are presented, in a somewhat chronological fashion, through opinion pieces and academic articles of authority figures, practitioners, and educators in computing fields. The aim of that approach is threefold: to describe the current views of computing as a part of a long continuum, to portray a picture of a rich and living discipline, and to present the recurring bones of contention concerning computing as a discipline.

By focusing on what was publicly said and written, this book excludes, to large extent, what was done in practice. Such account of computing might require an ethnomethodological approach instead. As the book focuses on central debates of computing, the book is not about the development of the great insights and ideas of computing; neither is it a story of people who have contributed to computing. The book's view is also very narrow because it excludes local debates and discussions in languages other than English. Because this book focuses on a few select development lines of computing, the book excludes a large number of crucial developments for computing as a discipline. Perhaps the most important of those excluded topics is the effect of the Internet and the World Wide Web. The web changed the discipline of computing so profoundly that those changes require a book of its own. Similarly, this book only discusses debates about computing's nature as a theoretical, engineering, and scientific field, but excludes many other conceptions of the computing endeavor.

This book is not a history book, either—the author is not a historian, it is not based on archival records, many of the events discussed in this book have not had a proper treatment by professional historians of computing, and "presentism" may haunt the book more than the author would hope. Quotations are presented as they appear in the literature, and they often do not reflect their authors' broader world view, those authors' views of all of computing, or how their views changed during their careers. Instead, quotations are used to exemplify what has been written at single points in time. Biographies, memoirs, and other similar works can provide the reader richer portrayals of each computing pioneer's thinking. Regardless, a book on computing's major debates has to look back to the development of the field, so many research studies by professional historians of computing have been used in writing this book and used as they stand. To help the reader to deeper treatments of the topics, pointers to the literature are presented in footnotes and at the end of each part of the book.

Organization of This Book

This book is organized around three central debates that reflected, and perhaps shaped, the discipline's formation. Many characterizations of computing have been formulated around three different intellectual traditions: the logico-mathematical tradition, the engineering tradition, and the scientific tradition. Although quite some academic computing work is done outside those traditions—following, for instance, historical, anthropological, or social sciences' research traditions—most debates on computing have revolved around those intellectual traditions. The role and relevance of each tradition has, however, been questioned at some point in time. A look into those debates is revealing about what was, and still is, at stake.

After the academic discipline of computing started to form, the first serious debates about computing's place in academia were concerned with the field's independence from other disciplines, especially mathematics and electrical engineering. Disciplinary identity was necessary for a large number of reasons, but the prestige of mathematics, the queen of the sciences, made the relationship between computing and mathematics ambivalent. Considerable effort was spent on describing the relationship between the two fields, and the role of mathematics in computing was a central bone of contention for three decades. The formal verification debate, especially, which had roots in the identity-forming years of the field, characterized the 1970s and 1980s discussions about computing as a discipline.

The first debate that this book portrays is the debate about computing as a formal, theoretical discipline—in many ways similar to mathematics, yet different in crucial ways. While that discussion has not completely petered out, it is no longer central to the field's search for identity. Part II of this book traces the discussions about the theoretical nature of computing, starting from the field's intellectual origins in mathematical logic. The difficult relationship between mathematics and computing is portrayed through arguments by the field's pioneers and through changes in the role of mathematics in computing curricula. Chapters 2 through 4 present characterizations of computing as a discipline of mathematical nature all the way to the end of the formal verification debate, and discuss modern views of the role of theory in computing against that background.

Second, although engineering was central to the birth of modern computing, for decades engineering, with its practical aims, was seriously undervalued in academic computing. In one of the early arguments for the emerging academic discipline of computing, computing education was explicitly distanced from technical considerations.[12] That argument, and many others at the time, made ignoring technology in computing look like a virtue. That spirit lingered in academic computing for decades to come—perhaps best expressed by oft-quoted phrases like "*computer science is not about machines, in the same way that astronomy is not about telescopes*"[13] and "*the computing scientist could not care less about the specific technology that might be used to realize machines, be it electronics, optics, pneumatics, or magic.*"[14]

The second debate that this book portrays is the debate about computing as an engineering discipline, which was brought under scrutiny at the end of the 1960s, when software engineering promised an end to the "software crisis." The engineering debate has also, in many ways, lost its momentum, yet many current arguments about the engineering character of computing can be understood as a continuation of that decades-long debate. Part III traces the engineering debates in computing from various technical innovations in the 1600s, to the conception of the modern technological paradigm of computing in the 1940s and the forming of computing as a new technical profession in the 1940s, to the early 2000s discussions that finally legitimized software engineering as a part of the academic discipline of computing. Chapters 5 and 6 present arguments for and against the view of computing as engineering within the context of software production, and analyze the clash between the theoretical and practical mindsets.

Third, while mathematics and engineering were something that, for various reasons, many computing pioneers wanted to dissociate the field from, science is a different story. Computing was always a tool for other fields of science and engineering, and many early pioneers wished to see computing research to be founded on scientific principles, too. Beginning in the late 1960s, computing publication forums saw a strong movement to liken the discipline of computing with the natural sciences and other empirical sciences. Before the late 1960s, science discussions were often concerned with naming the field—whether the discipline should be called a "science" or not. The "what's in a name?" dispute gave rise to a large number of opinion pieces over the decades. The name "computer science" penetrated the computing parlance so stealthily that it is hard to pinpoint the exact origins of the phrase: the term "computer science" was not mentioned in a naming discussion in 1958 but was found, in a plural form, in a mainstream publication in 1959 and one pioneer traced it to 1956.[15]

The naming issue was soon joined by another branch of "science" debates, this time concerned with the subject matter of computing. Whereas natural sciences study naturally occurring things, the subject matter of computing is in some ways artificial or human-made. The question was whether "sciences of the artificial" can be sciences in the traditional sense of the word. In one of the earliest arguments for the scientific nature of computer science, the authors argued that "phenomena breed sciences," and that computer science is the study of the phenomenon called computers and other phenomena surrounding them.[16] At one point of time, the subject matter debates seemed to have largely disappeared, but the turn towards natural computing breathed new life into the subject matter debate.

Early arguments for computing as an academic discipline often glossed over methodological questions, but over the course of time it became increasingly common to argue that computing is indeed science—not by virtue of its subject matter but by virtue of its method of inquiry. The methodology question, long bubbling under, was brought into the limelight by the "experimental computer science" debate. Driven by various motivations and visions, a campaign for "rejuvenating" experimental computer science started at the turn of the 1980s.

However, the rejuvenation campaign did not make clear what exactly was meant by "experimental computer science." In a nontechnical sense of the word, "experimental" can refer to exploratory work on novel and untested ideas or techniques. In a more specialized sense of the word, "experimental" can refer to a subset of *empirical* work: to the use of controlled experiments for testing hypotheses (perhaps "experiment-based" would be a less ambiguous term). The original "rejuvenating" report[17] teetered between the two meanings of the word but never made it clear what exactly was meant by experimental computer science except that it was desirable and should be funded. What followed was several decades of polemics during which discussants talked past each other, all talking about experimental computer science but meaning different things. The experimental computer science debate has been a notable feature of computing's disciplinary self-image since the 1980s, and that debate shows no signs of fading away. And in the course of computing's triumph in modeling phenomena in an impressive number of disciplines, some came to believe that the old queen of science, mathematics, was dead—long live the new queen, computing. Some even proclaimed the "death of proof."[18]

Part IV portrays the emerging view of computing as a science in its own right—not only as a tool for other sciences. Chapter 7 starts the story by discussing the early naming debates, and Chapter 8 continues to describe various views of computing as a science and some famous characterizations of the field. Chapter 9 continues with a description and analysis of the experimental computer science debate, which started in the 1980s and is still a widely and actively discussed topic. Chapter 9 ends with views to natural computing and the algorithmization of sciences.

The last part, Part V, discusses how computing's disciplinary debates changed over time. That part discusses how tools of rhetoric and narratives were used in the wrangles over computing's disciplinary nature, and how sometimes by "hijacking the narrative,"[19] one school of thinking was able to redefine how computing was conceptualized, while sometimes similar, perhaps better justified, efforts failed. The part describes sticking points that still continue to divide opinions about computing's very nature, and presents that the disciplinary disputes of computing seem trivial in comparison to the changes that computing and computational methods have caused in other sciences. The part ends with a discussion of computing as a discipline and of the importance of disciplinary self-understanding.

1.1 SCIENCE, ENGINEERING, AND MATHEMATICS

This book deals with different traditions of computing—the mathematical tradition, the engineering tradition, and the scientific tradition—but drawing lines between them is very hard. One of the reasons is that the intellectual endeavors they represent are not strictly definable. There is no single, monolithic Mathematics but a large variety of different kinds of mathematics. There is no archetypal example of science, but a broad range of activities, theories, philosophical standpoints, and other elements that together constitute windows to science. Science and mathematics are very tightly connected, too. And similar to science and mathematics, there is no universal agreement on what engineering is, but a spectrum of different views, each emphasizing different aspects of engineering. In addition to its heavy use of scientific knowledge and mathematical tools, engineering also has its own body of knowledge. Add concepts like technology, applied science, and mathematical logic, and the confusion is multiplied. While the problems start with the multiple meanings of those concepts, computing researchers sometimes exacerbate the issues by combining knowledge and methods from different traditions, and by occasionally moving between the traditions as if there was no distinction between them.

Nonetheless, as this book uses the terms mathematics, engineering, and science to draw lines between traditions of computing, some fundamental differences have to be noted between those endeavors. Although such separation between highly debated concepts is an open invitation for numerous angles of well grounded critique, it is nevertheless necessary. All the characterizations below are contentious and highly debated between philosophers of science, mathematics, and engineering—hence, the reader is advised to proceed with caution.

Aims

While some notable scientists have argued that the only valid aims of science are description and prediction of phenomena, many others consider exploration and explanation to be other important aims of science.[20] Exploration refers to developing an initial understanding of a yet uncharted phenomenon. Description refers to the attempt to systematically record and model the phenomenon and its connections to other phenomena. Prediction refers to the attempt to use previous understanding to predict phenomena that have not yet come to pass. And explanation refers to the attempt to clarify the causes, relationships, and consequences of the phenomena at hand. The aims of sciences vary remarkably between natural sciences, social sciences, and life sciences.

Many philosophers of engineering and technology have argued that the essence of engineering lies in its aims. What seems to be common to many different engineering branches is that they are constructive; they aim at *producing* things. For instance, in his analysis of paradigms of computing Peter Wegner wrote that research in engineering is aimed at development of tools that accomplish classes of tasks more efficiently.[21] Carl Mitcham, who is a prominent philosopher of technology, wrote:

> *Engineering as a profession is identified with the systematic knowledge of how to design useful artifacts or processes, a discipline that (as the standard engineering educational curriculum illustrates) includes some pure science and mathematics, the "applied" or "engineering sciences" (e.g., strength of materials, thermodynamics, electronics), and is directed toward some social need or desire. But while engineering involves a relationship to these other elements, artifact design is what constitutes the essence of engineering, because it is design that establishes and orders the unique engineering framework that integrates other elements.*
>
> Mitcham (1994, pp.146–147)

Regarding the aims of mathematics, there is considerable disagreement, suggestions ranging from understanding, to insight, to coherent structures, to creation of abstract beauty.

Certainty

Scientific knowledge, expressed as things like models, theories, constants, or laws, is tentative, and although many scientists claim that their theories or models are very good approximations of how the world works, the door is always wide open to better theories and descriptions of the world. Being approximations, scientific theories—and nowadays increasingly computational models—often compete for which theory gives the most accurate predictions or which one is the most widely applicable.

Contrary to that, mathematical knowledge, expressed as theorems, consists of necessary truths; truths which cannot be otherwise, given the selected set of axioms and rules. Theorems are accepted only if their conclusions are always true in that set of axioms and rules; their conclusions always follow from the axioms. Proofs are chains of substitutions within a formal system of rules, yet most proofs use natural language to increase readability at the expense of introducing some ambiguity. In principle, an informal proof can be expanded to a formal proof—but in practice, that is in most cases not doable.[22] Compared to sciences, in mathematics it is much rarer that significant theorems are first accepted and then shown to be wrong, although the history of mathematics has examples of such cases.[23]

According to philosophers of engineering, engineering knowledge, expressed as things like technical maxims, state-of-the-art solutions, and descriptive laws, is tentative, contextual, and unlike scientific and mathematical knowledge, not concerned with truth but whether that knowledge works.[24] Much of engineering knowledge is prescriptive and tacit, such as technical maxims ("rules of thumb," which offer heuristic strategies for successfully completing tasks) or descriptive laws ("If A then B" kind of experience-based rules).[25]

Methods

"The scientific method" is a catch-all phrase for a cycle of research that consists of systematic observation and collection of measurable data, formulation of hypotheses, testing those hypotheses through experiments, and analysis of results and possible acceptance, modification, or rejection of hypotheses. A broader term "empirical methods" covers all kinds of data collection, their analysis, theoretization, and testing those theories. Data collection and analysis methods are among the dividing elements between academic disciplines. Different scientific disciplines collect empirical data using very different sets of methods ranging from qualitative to quantitative, and data are also analyzed using a broad range of methods of analysis. Statistical analysis is used to generalize findings to broader populations, and nowadays computational models are a common tool at all stages of scientific research.

What makes the methods in mathematics different from those in science is that in pure mathematics, reasoning is deductive—mathematical induction is deductive, too. Generally speaking, new mathematics is strongly linked with old results in mathematics unlike in science, where new empirical knowledge that conflicts with old knowledge can be created. In pure mathematics there is no collection and analysis of empirical data, but the work is based on manipulation of abstract concepts—as well as on intuition and contextualization. George Pólya described in his famous book *How to Solve It*[26] how results in mathematics are presented as rigorous deductive proofs, but in practice, mathematicians use various heuristics, rules of thumb, guessing, and intuition.

In engineering, methods are often constructive and descriptive; they are actions aimed at achieving change in the affairs of the world. Billy Vaughn Koen described the engineering method as "*the use of heuristics to cause the best change in a poorly understood situation within the available resources.*"[27] In addition to scientific knowledge and mathematics, the engineering method usually relies on things like heuristics, technical maxims, and technological theories,[28] which are often encapsulated in *state-of-the-art* engineering practices. Techniques of engineers involve things like parameter variation—repeated measurement of the performance of a device or process, while systematically adjusting the parameters of the device or its conditions of operation,[29] often in search for optimal solutions with various necessary trade-offs. Denning et al. argued that in computing the cycle of engineering work consists of an ever-improving iteration of defining requirements, defining specifications, designing and implementing, and testing.[30] In addition to engineering methods, engineers use a wide range of methods from natural sciences (for studying, e.g., material properties) and social sciences (for studying, e.g., users).

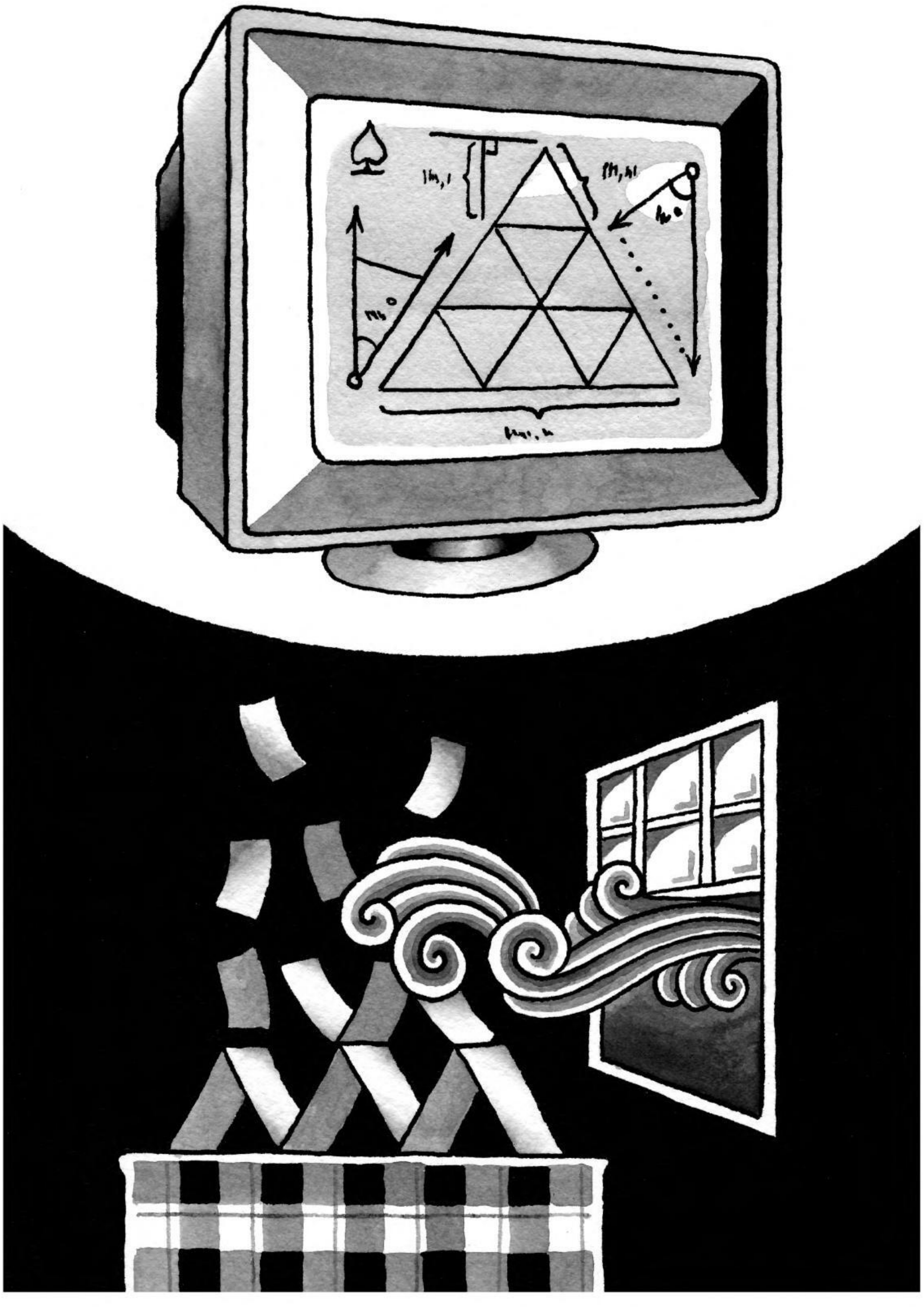
Subjects

Different kinds of science—physical sciences, life sciences, social sciences, and earth and space sciences—deal with very different kinds of subjects. While the subjects of physical science (such as atoms, fields of force, and properties of matter) are mind-independent, the subjects of social sciences (such as economies, societies, and preferences) are mind-dependent. What makes the subjects of engineering different is that unlike natural scientists who deal with naturally occurring phenomena, engineers deal with artifacts, which are created by people. In addition to artifacts, engineers' subjects also include people, the users of artifacts, because value and utility of artifacts arise from the human experience. Much research and design (and design research[31]) in software engineering, for instance, studies people and the artifact at the same time. And, lastly, different from science and engineering, mathematics deals with abstract, intangible objects.[32]

The different subject matters give rise to different claims of value-ladenness of work between mathematics, engineering, and science. Pure mathematics is usually considered to be value-free and basic science often claims to be value-free, but engineering acknowledges its value-ladenness. First, artifacts are created for a purpose and those purposes typically embody some values. Second, for the scientist, natural phenomena are not desirable or undesirable—they "just are," but for engineers natural phenomena can be desirable or undesirable—for instance, in the field of electronic communication thermal noise is an unwanted natural phenomenon.[33]

II

Computer Scientists and Mathematicians



Computer Scientists and Mathematicians

Beginning in ancient Greece, there has been a tight connection between mathematics and many other academic disciplines; the folklore has it that above the entrance to Plato's Academy there was a sign that read "Let none ignorant of geometry enter here."[1] Galileo Galilei's famous methodological stand was that the book of nature is written in the language of mathematics.[2] In a similar manner, it has been argued that mathematics is the quintessential knowledge and skill for computing disciplines. The relationship between mathematics and the discipline of computing seems so seamless that one computing pioneer called computing "the engineering of mathematics."[3]

Disciplinary debates over specific sciences often center around a few sticking points, or pivotal questions—questions that are so foundational that answers to them decide the fate of whole horizons of other questions. One pivotal question concerning scientific disciplines is whether specific sciences are reducible to other sciences. That question in computing asks whether computing, the discipline, is reducible to mathematics or logic.[4]

The reductionist view of the discipline of computing seems compelling. It is hard to know where to start or where to stop. Many forefathers of automatic computing—Pascal, Leibniz, and Babbage, for instance—were known for their contributions to mathematics. Years before the first modern computers were built, mathematicians had developed a definition of what can be computed with any kind of machinery, and many champions of modern computing were trained as mathematicians. The most impressive advancements in computing are frequently proven and presented in the language of mathematics. Mathematical structures—such as matrices, vectors, and graphs—are used to present organization of data in computers. Many branches of computing require sophisticated mathematical tools and techniques. Abstract algorithms can readily be turned into executable programs, and the program text can be formally proven to correspond to the formal specifications. The appeal of a mathematical reductionist view of computing has led some computing pioneers to argue that programming—the actual construction of computer programs—is a form of mathematics, too.

It is, however, one thing to say that the field uses mathematics as a tool and quite another to say that the discipline is reducible to mathematics. There are things that duly recognized computing professionals and theorists do that might not be reducible to pure mathematics, such as eliciting requirements, constructing models, designing and writing programs, testing and debugging programs, and designing user interfaces. Perceptions of the role of mathematics in the field of computing have also changed over the discipline's history and at no point has there been a consensus over that role and its centrality in the field.[5] Hence, it is also important to be clear about the limits of mathematics in computing.

This part of the book starts by describing, in Chapter 2, the mathematical roots of computing: the logical and mathematical ideas that underlie the birth of the discipline. It is important to understand those ideas in their original context, and not simply as "precursors" of modern computing: Pascal certainly did not wake up every day thinking, "Isn't it exciting to be a precursor of computer science here in the Renaissance!" The following chapter, Chapter 3, continues to describe the ambivalent relationship that academic computing had with mathematics once the field started to develop an independent disciplinary identity. Chapter 4 ends this part with a portrayal of how debates about computing's mathematical nature intensified throughout the 1960s and 1970s, coming to a head in the 1980s in an all-out clash between advocates and critics of strong formal verificationism.

CHAPTER 2

Theoretical Roots of Modern Computing

EACH academic discipline has a "hard core" of theoretical ideas that are rarely questioned. Those theoretical ideas underlie each field's *research agenda*: a broad consensus on the field's proper subjects of study, important questions in the field and how to pose them, proper methods and tools for achieving answers, what answers should look like, and valid interpretations of the answers.[1] The ability to independently set research agenda is one of the defining features of an autonomous academic field of research. Hence, it is often a good idea to investigate a scientific discipline by looking at the "hard core" of theoretical ideas, which form the foundations of the field, and which guide some of its research agenda.

Many computing's core concepts and ideas date back a long time and have roots in a variety of disciplines.[2] For instance, having different states of a computing machine stand for, or symbolize, different abstract ideas has roots in the history of symbol systems, yet the idea of abstraction is certainly not an obvious one. Russell noted that "*it must have required many ages to discover that a brace of pheasants and a couple of days were both instances of the number 2.*"[3] Binary representation of numbers—that two symbols is enough—also has a long history in mathematics but also in games, divination systems, and numerous other aspects of life. Discrete mechanisms—that the machine jumps between exact states instead of smoothly and gradually moving between values or states—can be found in various kinds of machinery over centuries. Turing's definition of computability—that five different operations are enough for carrying out any computation—is a much newer insight but it is rooted in centuries of development in mathematical logic.[4]

The history of modern computing is not "a" history but many intertwined histories that are concerned with different motivations, needs, and aims, and that are rooted in different intellectual traditions. One braid of computing histories is concerned with the mathematical and logical roots of modern computing, and has to do with the quest of formalizing human thinking or human problem solving into something that can be reduced to calculation and ultimately mechanized. In the book *The Universal Computer: The Road from Leibniz to Turing*[5] the mathematician and early computing pioneer Martin Davis started the history of modern computing in Germany in the 1600s, where Leibniz, one of history's great polymaths, had a vision of describing all concepts or ideas in our common body of knowledge ("alphabet of human thought") and also presenting the rules for combining them into more complex ideas, for reasoning about them, and for resolving which statements are true.

"Language of Thought"

The extraordinary German intellectual Gottfried Leibniz (1646–1716), "*the last universal genius,*"[6] was a visionary and an incredibly productive man. His significant contributions to multiple fields—law, mathematics, philosophy, natural sciences, and technology, among others—make him one of the true polymaths of history. What makes Leibniz a direct ancestor of modern computing is his envisioned language of rational thought that could be used to formalize human inference. The same vision, in different forms, drove the development of mathematical logic, and similar visions continue to drive numerous branches of computing from the semantic web to artificial intelligence.[7]

Leibniz was a child genius who became proficient in Latin by the age of twelve, and was thus able to read the philosophical and theological works in the library he had inherited from his professor father. As a teenager, Leibniz was introduced to Aristotelian logic, and he got captivated by Aristotle's categorization of objects in the human mind into ten groups based on their function in a proposition.[8] Aristotle's syllogistic logic is the type of logic that works with categories and their relationships, well familiar from elementary philosophy classes: "All men are mortal," "Socrates is a man," "Therefore, Socrates is mortal."

Taking concepts of logic to a higher level, Leibniz's dream was to come up with a general algebra, a sort of calculus that could be used to logically infer the truth value of any proposition. To make his vision come true, Leibniz needed three things. First, Leibniz needed a universal vocabulary or "database" of all concepts, *a characteristica universalis*, encapsulated in a special symbol system. Second, Leibniz needed a script that was able to formally represent the relationships between thoughts. Third, he needed rules of inference, *a calculus ratiocinator*, for determining which propositions written in that language were true and which were false. One does not need much imagination to draw parallels between those three projects and many branches of computing today. For Leibniz, such innovation promised an end to disagreements in areas ranging from metaphysics and morals to geometry and analysis: instead of quarreling, "let us calculate"—"*calculemus!*"[9]

> *If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants.*

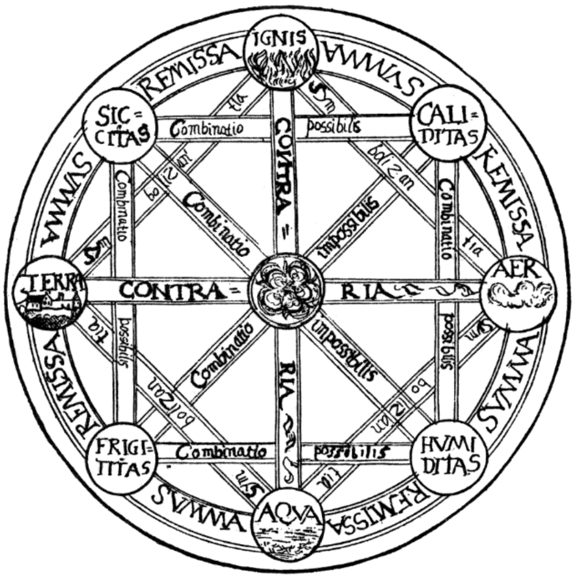


FIGURE 2.1 Leibniz's illustration of concepts and their relationships. Source: Leibniz (1875, B.IV).

For it would suffice to take their pencils in their hands, to sit down to their slates, and to say to each other (with a friend as witness, if they liked): Let us calculate.
Leibniz, translated in Russell (1937, p.200)

Leibniz's idea behind his thinking was simple: He believed that all our ideas can be reduced to a small number of concepts (an alphabet of human thought) and that all complex ideas are rationally deduced and combined from those concepts. In Leibniz's world there was no room for chance: his world was deterministic through and through. Unfortunately, Leibniz presented only few concrete examples of his idea; Figure 2.1 portrays one of Leibniz's diagrams.

In Figure 2.1, earth (*terra*) and air (*aer*) are opposing pairs, as are fire (*ignis*) and water (*aqua*). The four corners represent properties, or qualities, of the elements: dryness (*siccitas*), heat (*caliditas*), humidity (*humiditas*), and coldness (*frigititas*). The elements are formed in combinations of their properties: Fire is possible at the combination of dryness and heat, air at the combination of heat and humidity, water at the combination of coldness and humidity, and earth at the combination of coldness and dryness.

Despite his numerous intellectual contributions to other fields, Leibniz did not get very far with his universal language. He did, however, make pioneering contributions to logic. Leibniz developed a notation and algebra of logic and presented the idea of using logical operators to manipulate concepts in the