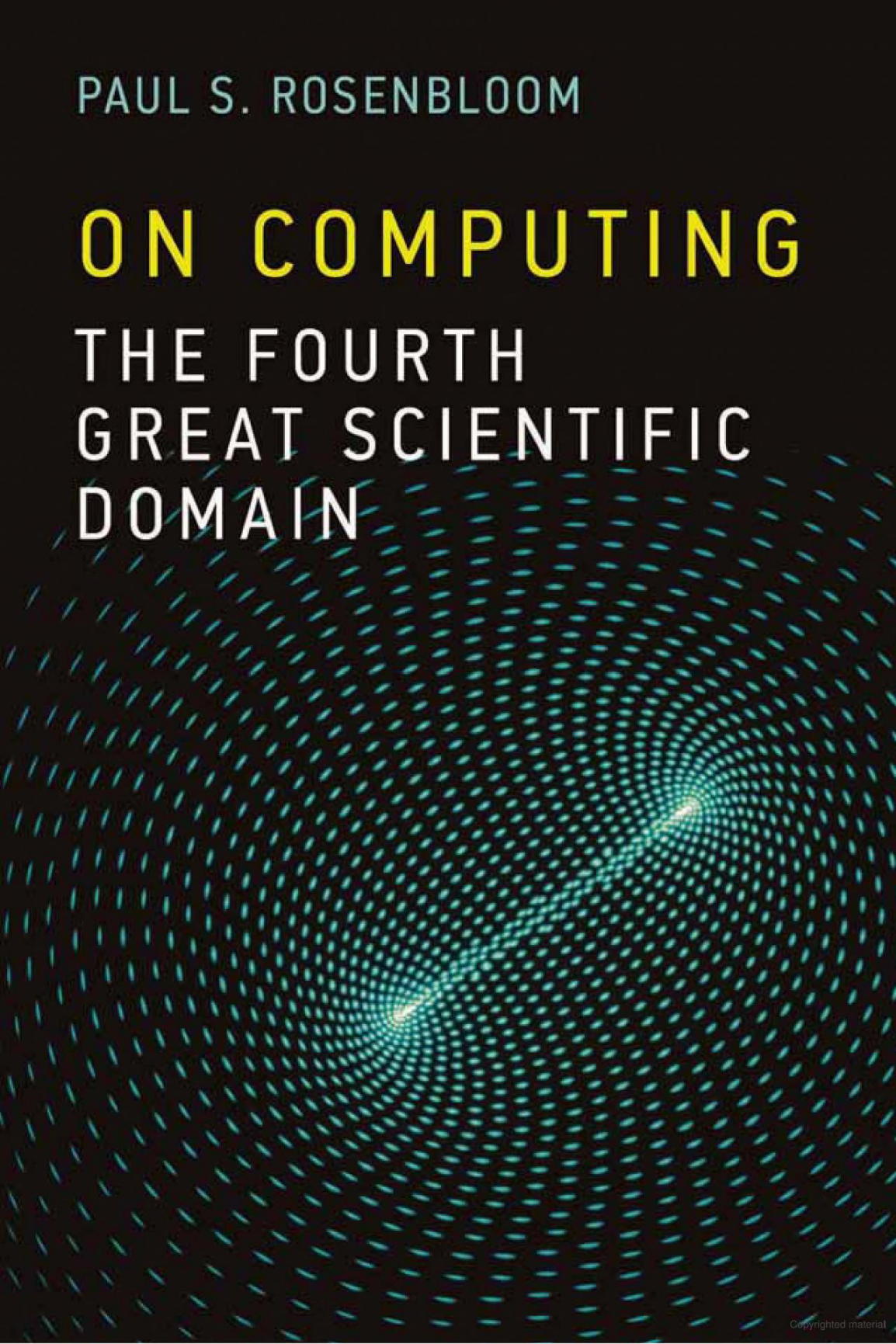
PAUL S. ROSENBLOOM

ON COMPUTING

THE FOURTH GREAT SCIENTIFIC DOMAIN



On Computing

The Fourth Great Scientific Domain

Paul S. Rosenbloom

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Stone Sans and Stone Serif by Toppan Best-set Premedia Limited. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Rosenbloom, Paul S.
On computing : the fourth great scientific domain / Paul S. Rosenbloom.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-262-01832-6 (hardcover : alk. paper)
1. Computer science. 2. Computer science—Research. 3. Computer science—Philosophy. I. Title.
QA76.R657 2013
004—dc23
2012013852

10 9 8 7 6 5 4 3 2 1

Contents

Preface

Computing isn't simply about computers (hardware) or programming (software). Nor is it merely about the act of determining the results of numerical expressions (calculation) or the development of tools (applications) for use by others. It is an exciting and diverse, yet remarkably coherent, scientific enterprise that is highly multidisciplinary while maintaining a unique core of its own. It has a future that promises to be as rich and momentous as its revolutionary past has already been.

This book is about the *computing sciences*, a broad take on computing that can be thought of for now as in analogy to the physical sciences, the life sciences, and the social sciences, although the argument will ultimately be made that this is much more than just an analogy, with computing deserving to be recognized as the equal of these three existing, great scientific domains—and thus the *fourth great scientific domain*. In eschewing the common phrase *computer science* and using instead a relatively unfamiliar phrase,[1] the goal is to transcend the normal boundaries and modes of thinking about both computing and its study, capturing each in its broadest sense. This then becomes the basis for a new perspective on computing that is open and expansive yet fundamentally coherent and that has major implications for how computing is viewed both in and of itself and in the larger context of the global scientific enterprise.

At one level down, a multiplicity of interrelated purposes have shaped this work, making for a potentially complex narrative and the likelihood that some readers will resonate with only particular segments of the overall story. Yet the story remains most compelling when told all of a piece, so that is how the reader will find it here. To help tame the complexity, five primary purposes have been articulated and associated with classes of readers for whom the purposes may be of most interest. This then yields a map of the overall structure and content of the story while also guiding those with more focused interests toward what is likely to be of most relevance to them.

The five primary purposes are as follows:

1. Introducing the *relational approach*—comprising a *relational architecture* and the accompanying *metascience expression (ME) language*—for structuring and characterizing the computing sciences, yielding a novel way of looking at them and how they relate to the other sciences.
2. Conveying the excitement of the present for the computing sciences and the revolutionary potential they continue to possess for the future.
3. Providing a broad multidisciplinary perspective on the computing sciences as a counter to the narrow and frequently self-limiting views of computing that are too often encountered.
4. Exploring three forms of *macrostructure* afforded by the relational approach for structuring and understanding complex multidisciplinary topics and organizations within computing.
5. Defining a dynamic, inclusive notion of a great scientific domain and arguing for the computing sciences providing the fourth such domain.

As listed, the first of these primary purposes is to introduce a new approach to understanding computing in terms of how it relates to both itself and the rest of science; in particular, the physical, life, and social sciences. The architecture and language derived from this approach form the core technical contribution of this book. The term *architecture* is most familiar colloquially in the world of buildings, where it refers to their design or structure. Within computing, the term is most familiar in the context of computer architecture, where it denotes the design and structure of hardware that computes. However, it also plays a role in software engineering—in the form of software architecture—and in cognitive science, where cognitive architectures model the fixed structure underlying intelligent behavior. As used here, the focus is on structure rather than design and on the structure of the field of computing rather than on the structure of computers.

The relational architecture provides an organization over the field of computing in terms of two relationships—*interaction* and *implementation*—that reveal hidden structures and connections among its diverse disciplines. It also exposes the domain's natural role in the larger world of science and engineering while highlighting an inherent symmetry between what are traditionally viewed as central versus peripheral—or core versus multidisciplinary—aspects of computing. The unidisciplinary[2] aspects of computing end up accounting for only a rather limited subspace of its overall coverage.

Architectures within computing typically induce useful languages. A computer architecture, for example, induces a machine language that enables programming of the computers based on the architecture. Likewise, software architectures may support architecture description languages, and cognitive architectures may yield languages for building intelligent systems. The relational architecture does not induce an executable language in which programs can be run, but it does yield a language—the ME language—that supports the analysis and description of computing disciplines and organizations in terms of the domains they comprise and the relationships that are implicated among these domains.[3]

The primary audience for such an architecture and language is likely to be both computing professionals—whether in industry or academia—and students who are interested in a new perspective on the domain of computing, its component parts, and how all of it can be seen to fit together into a coherent whole with a well-defined place in the overall scheme of the sciences. Although not written explicitly to be easily accessible to a broader audience, this perspective may also be of interest to people outside the field who seek a new way of understanding computing and as well to anyone interested in the structure of science as a whole. The architecture has predominantly been developed and explored in the context of computing, but it reflects a symmetry across the sciences that suggests the possibility of broader applicability and utility.

The relational approach arose from a computer (and cognitive) scientist's reflections on computing. Although it is natural and appropriate for an expert in a field to reflect on his or her field in such a manner and to want to share the results of such reflections with the broader community should they appear useful, my reflections by necessity led me deeper and deeper into philosophy of science in order to consider such topics as what science itself is and what makes a great scientific domain. I am an amateur in philosophy, so there is an inherent risk of philosophical naïveté in such an endeavor, yet such excursions became necessary to fill in the whole story. These latter topics are at the core of the fifth purpose, but they also impact aspects of this first one. For example, the idea of symmetrically relating computing to the physical, life, and social sciences assumes that there is at least some form of equivalence among them, while the appropriateness of the term *metascience* for the accompanying language depends on taking a broad view of science.

Earlier drafts of this book placed this philosophical material before the architecture to set the stage for its introduction. However, this led to two problems. First, the strong early dose of philosophy proved to be a

deterrent for many people within computing—the primary audience for this book—making it difficult for them to get on to the more accessible and relevant material to come. Second, this ordering may have been construed to imply that the utility of the architecture depended on the validity of these excursions into more controversial aspects of philosophy of science. If either the notion of a great scientific domain or the proposal that computing is the fourth such domain ultimately fails to pan out, some of the sense of fundamental rightness about the architecture may go away, but its usefulness as a tool for understanding computing may remain. Similarly, if the attempt to broaden the standard notion of science should prove unworkable, the ME language could still be a useful tool even if the term *metascience* in its name becomes inappropriate. In consequence, the architecture is now presented up front, in chapter 2, with most of the more philosophical material delayed until chapter 6. Only as much of the philosophy as is necessary to understand the architectural material is included early on, such as the discussion of the nature and scope of science, which can be found as part of the introduction to the computing sciences in chapter 1.

The second listed purpose—of conveying the excitement and potential of computing—has a situational aspect. Computing has been in the doldrums in many ways since the dot-com bust. In combination with this bust, a focus on the mundane in computing—what can be considered the *trees* of its currently successful computing products and applications rather than the *forest* of its full scope and potential—has led to a growing concern that the excitement is gone from the field, or at least that the world perceives this to be so. In academia, this has been manifested most dramatically in a drop in undergraduate enrollment, but it also is evident in everyday interactions with people from both within and without the field. In response, various individuals and groups have attempted to reconceptualize and rearticulate the nature of computing, along with why it remains a vital and exciting intellectual endeavor with broad societal implications. One prime example is an effort to identify the *great principles of computing* as a vehicle for refocusing the field on its fundamental scientific principles.[4] Another is the explication of *computational thinking* and its pervasive impact on the world at large.[5]

Part of the motivation for the second purpose is to contribute to this ongoing discussion about the nature and vitality of computing by surveying, in chapters 3 and 4, much of what is exciting in the present and future of computing. But the survey isn't solely a response to the doldrums within computing, which as of mid-2011 we are already beginning to see turn

around, at least with respect to undergraduate enrollment and high-flying valuations for computing companies. To illustrate the application of the relational architecture, help better understand the field of computing, and infuse additional novelty and interest into the survey, it is structured according to the relationships used within the architecture, with chapter 3 covering implementation and chapter 4 covering interaction.

The survey concentrates on what can be called *dyadic computing*, the combination of computing with one other domain; as is seen for example in computing's combination with the physical sciences in computer hardware and with the social sciences in human computer interaction.[6] Much of what is most familiar in computing over the past few decades fits into dyadic computing, but there is also a range of newer and less familiar topics that show up here, such as brain–computer interfaces and the use of DNA for biocomputing. Beyond dyadic computing is *polyadic computing*, which includes the full breadth and richness of computing that arises from its combination with an arbitrary number of other domains. Examples here include intelligent robots and cyborgs and worlds that tightly intermingle the real and the computational/virtual. Much of the current leading edge in computing, and even more of its future, falls within the scope of polyadic computing. Aspects of polyadic computing are included in the survey when there is a single dominant relationship by which we can classify the area at the top level. Other pieces are included in chapters 5 and 6 in support of later purposes.

Because so much of the future of computing seems to lie in multidisciplinary areas, and because such topics are too often given short shrift within the field, the survey concentrates more on such aspects of computing than on traditional core topics. Within this overall focus, my own background in artificial intelligence (AI) may have led to AI and robotics being somewhat overrepresented in both examples and descriptions—as my home base at the University of Southern California may have led to examples from there also being overrepresented—but the explicit goal has been to be universal and inclusive. Whether biased or not, the survey's broad multidisciplinary focus implies that it also contributes directly to the third purpose. ME expressions are assigned to the individual topics to make explicit the diversity of domains—and the relationships among them—involved in each topic, while also helping to identify where the topics fit within computing, and science, as a whole. Readers uncomfortable with formal or semiformal languages or who simply do not want to try to puzzle out what particular expressions are intended to denote are more than welcome to skip over them during the survey.

The survey is potentially relevant to anyone concerned with the first, second, or third listed purposes, including professionals working within the field, students just learning about or considering entering the field, and people outside of the field who have an interest in it. However, the level of detail provided on the topics surveyed and the amount of background assumed in their presentation can have a major impact on the survey's accessibility for each potential audience. It seems likely that both the level of detail desired and the amount of background that can be assumed will decrease as we proceed across these three potential audiences. The survey provided in chapters 3 and 4 mostly assumes a level of understanding corresponding to the first audience while leaving open the possibility that sufficiently motivated or guided members of the other two will still find it useful. In contrast, a very detailed survey is beyond what is necessary to support any of the three listed purposes, so the survey proceeds at a relatively high level.

We have already seen how the survey contributes to one aspect of fulfilling the third purpose, but more broadly the third purpose is concerned with countering a too common tendency to construe computing in an overly narrow fashion, potentially limiting its overall scope, significance, and future. It is not surprising that many people tend to view computing through the lens of what they see on a regular basis. Thus, to much of industry and the general population, computing becomes about the hardware and software they use. For academic computer scientists, it becomes what is covered by the traditional disciplinary structure. For scientists outside of computing, it becomes the tools from computational science and informatics that they use in pursuing their own research. These individual views all derive from what should be a positive—the pervasive utility of computing in helping people accomplish their personal and professional goals—but even in combination these topics fall short of its full richness and of what makes computing an exciting and crucially important scientific enterprise. A true understanding of computing should enable the nature and potential of the field to be explained properly, both to the general public and to our colleagues, students, and selves.

Much of the argument here derives from definitions that will be provided in chapter 1 plus the relational architecture in chapter 2 and the dyadic survey in chapters 3 and 4. Together these foster a view of computing that is broad, frequently multidisciplinary, and symmetric in how it relates to the other sciences. So, for example, although the provision by computing of tools for the other sciences fits naturally within the archi-

tecture, this topic is far from all there is to the architecture (or the survey). Some of computing's relationships have nothing to do with providing tools, and the symmetry of those that do implies that the other scientific domains may stand just as much in the role of providing tools for computing as the other way around. In addition to the relevant material in chapters 2–4, aspects of this argument will also be covered in chapter 5 (section 5.2), as part of a broader discussion of the use of computing in science. The intended audience for this discussion is anyone who wants to expand their view of what computing is, subject to the same accessibility qualifications that were mentioned previously.

The fourth listed purpose seeks to leverage the relational approach in exploring forms of higher order coherence, or macrostructures, over complex computing topics and organizations. Three such forms of relational macrostructure are examined in chapter 5: (1) two-dimensional tabular, (2) complex expression synthesis and decomposition, and (3) layered hierarchical. Topics that will be analyzed include mixtures of real and virtual worlds, methods for the pursuit of science, computing research institutes, and academic computing. The intent is to leverage these four analyses to provide new insights into the topics analyzed while simultaneously exploring and illustrating the utility of these macrostructure forms and providing further feedback on the applicability and utility of the relational approach for understanding complex aspects of computing. This is of potential interest to anyone concerned with understanding complex scientific areas and organizations in general, understanding any of these four specific topics in particular, or simply understanding computing as a whole better.

The fifth listed purpose focuses in chapter 6 on the development of the notion of a great scientific domain, plus the argument that computing forms the fourth such domain.[7] This particular notion of a great scientific domain is a new one, but it is intended as an elaboration and refinement of the long-familiar division of the sciences into broad domains that study the physical, living, and social worlds. These three existing great scientific domains are the crown jewels of human intellectual achievement. Each covers a broad swath of the understanding of our world and ourselves while simultaneously enabling the development of technologies that ever increasingly alter the world and us. Although computing has undoubtedly spurred massive technological developments, it is most often viewed as a form of engineering rather than as a part of science that is on a par with standard disciplines such as physics, chemistry, biology, or psychology, not to mention any question of parity with an entire domain such as the

physical, life, or social sciences. So it is a rather large leap to propose that computing forms the fourth such domain. But, if it is true, it provides an even stronger argument than the survey for the importance and future of computing.

A key part of this is by necessity a discussion of what science is. There are many extant definitions of science, covering a range of senses from "a particular branch of knowledge"[8] at the broad, permissive end of the spectrum to "a system of acquiring knowledge based on scientific method, and to the organized body of knowledge gained through such research"[9] at the narrow, restrictive end. Yet, none quite matched the way I as a practicing scientist think about science. Continued pursuance of the reflections on computing that originally produced the relational architecture have led to a very broad and rather nonstandard answer to the question, one that incorporates both *understanding* (as is the focus in most traditional science) and *shaping* (as is the focus in engineering and other professions) and includes essentially anything that increases our understanding over time.

Both this discussion and the definition of a great scientific domain may be controversial. They are due to a philosophical amateur who may be both ignorant and naïve in key ways; they bulldoze through many of the traditional ideas and arguments about science; and they embody a form of circularity in which reflections on computing lead to notions of science and great scientific domains that are then used to make claims about computing. But the intent here is more to initiate a conversation on these topics than to provide final answers. Backing down from raising controversial claims because not all of the *t's* can yet be crossed nor all of the *i's* dotted would not do this, nor would waiting until the evidence is so solid that the claims will be accepted with little dispute. My concern in including this material is therefore not so much with ultimately being proved right about these claims but with providing food for thought in service of accelerating the field's progress toward compelling answers.

There is a need and a role for professional scientists to consider and speculate about the natures of their domains and not just leave it to philosophers. My hope is that computer scientists, and possibly other scientists and members of the general population, will find this discussion thought provoking, even if not authoritative from a philosophical perspective. Getting something out of this material will however require a willingness to deal with its philosophical nature.

The overall structure of this book follows from its purposes. Chapter 1 defines and scopes computing and the computing sciences, including a necessary prefatory discussion about the nature of science; chapter 2 intro-

duces the relational approach, as composed of the relational architecture and the ME language; chapters 3 and 4 discuss and survey the implementation and interaction relations; chapter 5 focuses on macrostructures and analyses; chapter 6 covers the more controversial topics, including a second pass on the nature of science, the definition of a great scientific domain, and the claim that computing is the fourth such domain; and chapter 7 concludes. Yet before we get started on this material, this preface wraps up with a note on writing style, a section on the history and background behind this book, and acknowledgments of those to whom I am most indebted for help with it.

This book contains a wide diversity of material that varies in its nature, the style in which it is written, my level of expertise concerning it, and the types of citations included for it. The background and acknowledgments sections are, for example, personal, and therefore are written in such a style. But I have also felt free elsewhere to resort to personal anecdotes when they seemed apropos to the point being made. In contrast, the relational architecture and ME language are technical in nature and are presented in a more professional style. The survey is more of a mixed bag. The presentation style here is a mixture of description and gee whiz, with a dash of technical, as befits the material and the purposes for its presentation. I have expertise in some of the areas covered in this book and a working familiarity with others but have had to learn about some of the topics from scratch in service of fleshing out the respective sections. The citations here reflect this, sometimes being appropriately scholarly but at other times citing more general sources, such as Wikipedia. Although thus not always authoritative, the survey still hopefully fulfills its intended purposes. The philosophical material is written in a more polemical style, opinionated and undoubtedly controversial. It captures a working scientist's, rather than a professional philosopher's, views on the material, with the citations limited to the relevant sources consulted during these reflections. I have also felt free to resort to a combination of personal and polemical styles for a few non-philosophical topics—such as the discussion of academic computing—where personal experience has led to a particularly strong opinion.

Background

If I were to go back to the late 1990s and try to predict what the future would bring for me professionally, it would have been quite unlikely to include this book. At the time, I was a confirmed AI researcher with a

long-term commitment to understanding the architecture of cognition; that is, the fixed structure underlying thought in both natural (e.g., people) and artificial (e.g., computer) systems. I was just coming off a successful multiyear project funded by the Defense Advanced Research Projects Agency (DARPA)—which is probably best known for having given the Internet its start—to use such an architecture in building simulated pilots and commanders for large-scale virtual military exercises.[10] What occurred then, however, was a classic case study in *desires and diversions*, as outlined in Allen Newell's last public lecture of that name.[11]

In his talk, Newell—one of the four founders of AI—reflected on styles of research careers, with a particular focus on his own style, the all-encompassing desire to answer a single ultimate scientific question; in his case, understanding the nature of the human mind. Such a question can inspire a life-long scientific career, obscuring from view—or at least from serious consideration and investigation—all other topics. Yet, diversions do inevitably occur, generally for social reasons, and may last for years. For Newell, these diversions included foundational work on computer architecture, speech understanding, the psychology of human–computer interaction, and hypertext systems. To him, the important point was ultimately to make such diversions count, not only by significant contributions to the subject matter of the diversion but also by salvaging something for the central question. Given that this book is about all of computing, with a strong focus on multidisciplinarity, it is notable that the ACM/AAAI Allen Newell Award was created after his death to honor career contributions that have breadth within computer science, or that bridge computer science and other disciplines.

Until 1998, I followed a research path that was closely tied to Newell's, working closely with him from the time I entered the graduate program in computer science at Carnegie Mellon University in 1976 until his death in 1992, and continuing to work on what had become a large-scale joint research project until 1998. He was initially my PhD advisor but later a friend, mentor, and collaborator. One of the reasons we had such a successful working relationship over so many years was that I was driven by a scientific question that was closely related to his and on which he had done much of the seminal work. In my case, it was the understanding of cognitive architecture. I was particularly interested in seeking a uniform architecture that could yield the full richness of intelligent thought and behavior from the interactions among a small set of very general mechanisms—essentially what could be considered cognitive Newton's laws. The uniform approach to cognitive architecture was not, and still is not, the

dominant paradigm, but to me it had an irresistible appeal both scientifically and aesthetically.

In the early 1980s, John Laird, Allen Newell, and I instituted the *Soar* project to develop, understand, and apply a new cognitive architecture—eponymously called Soar—that would capture our best joint understanding of the mechanisms necessary for producing intelligent behavior.[12] Laird was a fellow graduate student and Newell advisee whose PhD thesis developed the problem-solving model at the heart of Soar, and my thesis focused on a model of human practice that would become Soar's main learning mechanism over most of its history.[13] Soar continues to be developed and applied to this day under Laird's leadership at the University of Michigan, roughly thirty years after the first version was built. But my own active participation ended in 1998, after spending much of the 1990s working on the DARPA project mentioned above, and then spending a sabbatical year exploring, largely unsuccessfully, the role of emotions in Soar.

I left the project because I had hit a wall; a metaphoric wall to be sure, but it still brought progress on my central research desire to a halt. I no longer felt that I was making a significant difference on what I had hoped and expected to be a career-defining question and had little enthusiasm left for the chase. After painful and extended soul searching, I concluded that a major change was necessary, but whether it was simply to be a diversion—of whatever length—from which I would ultimately return or a permanent shift was unclear at the time.

When Herb Schorr, executive director of the Information Sciences Institute (ISI)—a large research institute in Marina del Rey that is part of the Viterbi School of Engineering at the University of Southern California (USC) and where I had pursued much of my own research during the prior decade—offered me a chance to play a leadership role in ISI's New Directions activities, I jumped at the opportunity it afforded for refreshing my understanding of the broader world of computing and for working all around its cutting edge. At that time, ISI engaged in research and development across much of computing, spanning computer science (software), computer engineering (hardware), information science (information), and information technology (applications). It had been quite successful since its inception in 1972, with notable contributions in networking—such as the development of the Domain Name System (DNS), which associates hierarchical names with resources on the Internet, and the attendant creation of the initial top-level Internet domains, such as .com and .edu—AI, grid computing, software engineering, computer architecture, and high-performance computing. But, as a soft-money research institute that was

totally dependent on external contracts and grants, ISI could only remain large and successful by continually striking out into new areas of value.

My role during what ended up as a decade (1998–2007) of leading New Directions was to help understand where computing was heading and where ISI should be going, as well as to lead, instigate, and facilitate new efforts in these directions. This meant working across a broad range of computing disciplines, both within ISI's existing areas of expertise and in new areas of potential interest. It also meant working extensively at the interfaces between these areas of computing and an even wider diversity of critical topics and problems outside of computing. At various times this included interactions with the Getty Center about technology and the arts; Paramount Digital Entertainment and the U.S. Department of Defense (DoD) about combining computing and entertainment for military training, culminating in the creation of the U.S. Army–funded USC Institute for Creative Technologies (ICT); the Southern California Earthquake Center (SCEC) about the development of a community earthquake modeling environment; USC's Keck School of Medicine about biomedical computing; the National Science Foundation (NSF) and others about responding to unexpected events, both natural and human engendered; and a panoply of academic disciplines and industrial sectors about automated building construction. (ISI has continued down this path in more recent years as well, with for example the creation of a quantum computing center, in conjunction with Lockheed Martin and D-Wave, to house a 128-qubit quantum computer.)

As I reflected on the breadth of topics and interactions with which I was involved, I began searching for any form of commonality, coherence, or structure that might be lurking behind what initially appeared to be nothing more than a hodgepodge begotten from a liaison between need and opportunity. Just as with the earlier work on cognitive architecture, I was inexorably driven to investigate whether this diversity of topics was merely the complex surface manifestation of an underlying coherent whole—what I eventually began to think of as an architecture for the domain of computing—that might itself be explained uniformly in terms of interactions among a small set of general principles. The relational approach ultimately emerged from these reflections.

The first exercise in disseminating the results of these reflections was a short article published in *IEEE Computer* in which a precursor of the relational architecture—referred to as a framework at that point—was proposed as a basis for the rational reconstruction of the sibling disciplines of computer science and engineering.[14] The positive responses to this article

helped encourage me to continue, but other responses pointed out the potential for controversy in such an endeavor. More will be said about this latter bit in chapter 5 (section 5.4).

A desire to introduce this overall perspective to students who were in the process of developing their own views on the field led to a subsequent graduate seminar in which the framework was further developed and used as a vehicle for introducing to them many of the new and exciting developments in the field.[15] This in turn, along with further external encouragement and the opportunity of an upcoming sabbatical, led to the idea of a book that would make the ideas available more broadly. Originally, this was conceived of as following the structure of the course and for an intended audience of everyone from the technically interested layperson to professionals in the field, but with a particular focus on students considering a career in computing. Reality, however, in the form of feedback from publishers and from family and friends who had tried their best to make it through early drafts of the introductory chapters, forced a change: It was made clear that I needed to retarget, and reduce, the intended audience along the lines mentioned in the previous section.

While writing this book, I realized that the relational architecture implicitly assumed a symmetry between computing and the physical, life, and social sciences. The process of understanding and justifying this symmetry led to the notion of a great scientific domain and the idea that computing was the fourth such domain, although it took the enthusiasm of Peter Denning—a former president of the Association for Computing Machinery (ACM) who has been leading the push to identify the great principles in computing—to make me realize that this latter idea was not just a supporting point in the development of the architecture but a critically important point in its own right. This led to a joint column on this topic in *Communications of the ACM*.[16]

Formulation of the notion of a great scientific domain and the need to establish whether or not computing was one also led me to think more about what computing is in the abstract; that is, what distinguishes the subject matter studied by the computing sciences from that studied in the physical, life, and social sciences. Although the resulting definition of computing as *information transformation* is neither terribly novel nor controversial in its own right, its combination with the notion of computing as a great scientific domain has led to the controversial proposal that mathematics should properly be viewed as part of the computing sciences. These thoughts yielded a recent contribution to an ACM *Ubiquity* online symposium "What Is Computation?"[17] The definition of computing is discussed

further in chapter 1, but the controversial nature of the proposal concerning mathematics, and the fact that the rest of the book does not depend on this proposal, have led to deferring its discussion until chapter 6.

I have also been tempted by several opportunities to apply the relational approach further and further beyond my core expertise in computing. In one case, this led to an article analyzing the relationship between science and society[18] and in another case to a discussion of the digital humanities.[19] The latter article includes a potentially controversial claim that is analogous to the one about mathematics, but here proposing that the humanities should properly be viewed as part of the social sciences. As with the proposal concerning mathematics, this one is discussed further in chapter 6.

This brings us back up to the present and the culmination of this more than decade-long diversion into new directions in computing and the nature of the computing sciences. One of my personal hopes in writing this book is that it will help make this diversion count by challenging readers to rethink their own assumptions about computing and assisting them in attaining a better understanding of it, its nature and structure, its role in the larger world, and its potential for the future. When I teach or mentor students, what I value most is the opportunity to expose them to new ways of thinking. New facts are essential for progress, but if they just fit into existing, well-trodden paths, they too often miss their latent ability to transform. The current contribution may be short on new facts, but the hope is that it will provide a fresh perspective on those already known while helping to illuminate a path toward the future.

From a more local and personal perspective, this diversion has already counted through the new directions that we were able to initiate at ISI. But I am also in the process of seeing what I can salvage from it for my own ultimate scientific question. Writing this book has made me realize how absolutely central the metaphor of Newton's laws—of finding a small core of essential regularity across diverse phenomena and integrating this core uniformly into a coherent whole—is to my whole intellectual life. Without such a focus, I struggle to make progress and too often feel like I am not making the kind of difference that I most value. In summer 2008, I returned to my core desire of understanding cognitive architecture, but with a new inspiration for how to create architectures capable of producing the broad diversity of intelligent behaviors from a small set of principles. In a second bit of salvage from my decade of new directions, this new research is being pursued at USC's ICT, an institute I helped to create in the late 1990s. The results from this new effort are promising,[20,21] and I am excited about its future.

Acknowledgments

I would first like to thank Allen Newell, who taught me to be a scientist, inspired my approach to science, guided my early career, and who has always stood as an icon of the true scientist to which I could only aspire to approximate in my own career. I frequently play a video of his "Desires and Diversions" talk to groups of students, both in classes I teach and in more informal settings, to help get them started on thinking about their own research styles and future scientific careers. It inspires me to reflect anew on my own career each time I see it.

These re-perusals also keep reminding me that in the talk, he warned against attempting multidisciplinary science without being a professional in all of the fields involved. I kept this advice in the back of my mind as a warning as I pursued forays into philosophy and areas of polyadic computing in which I have little to no real expertise, but I have not been able to resist including such material here. Only time will tell whether I am right in doing so or whether this is another case where I would have been better off following his advice.

The meat for the initial reflections in this book came out of my decade of involvement in New Directions activities at ISI. Herb Schorr offered me the opportunity to lead these activities at a time when I needed a major change from my existing research path. He further provided critical inspiration, support, mentoring, and encouragement over the years of my involvement. I am also, however, indebted to researchers from across ISI and other organizations with whom I was able to work on these new directions and from whom I learned a lot about many aspects of computing and its interactions with other domains.

Peter Denning has provided a range of helpful interactions about this book's topic and material, encouraged the overall enterprise, forced me to rethink and revise a number of aspects, suggested the name of the relational architecture, and provided the insight that the identification of computing as the fourth great scientific domain was a major point in its own right rather than merely being a supporting point in the development of the relational architecture. He has also repeatedly cautioned me about the more controversial aspects of the book, such as the proposed subsumption of mathematics within computing. This has helped me decide how such material should be presented, but otherwise I'm afraid I have treated it like Newell's advice on multidisciplinary work, keeping it in the back of my mind while still pushing forward with what seemed right to me. Here, too, only time will tell who was right.

A range of family and friends made dogged attempts to read early drafts of the introductory chapters and to provide useful feedback as I unsuccessfully struggled early on to write a book that would be accessible and of interest to a general audience. This included Pam Fenton, Sharon Elaine Lee, David Nagy, Anne Rosenbloom, Florence Rosenbloom, Kate Rosenbloom, and Michael Rosenbloom. Several anonymous reviewers for MIT Press also went through a similar exercise, either with just the initial chapters or a full early draft. I want to thank them all for their efforts and apologize for their difficulties. They did provide an invaluable service in helping me get this book on the right track.

I would also like to thank the students in the class I taught at USC on this subject, who helped debug some of the ideas and deepen some of the insights; my colleagues at ICT who have helped further broaden my understanding of the computing sciences; Julia Kim of ICT, in particular, for reading through a full draft of this book and, among other useful pieces of feedback, suggesting the shift of the more philosophical material to near the end; and my colleagues in USC's Computer Science Department who listened to my early ideas on this topic and lived through a multiyear experiment in restructuring the department along lines suggested by it (more about which can be found in chapter 5, section 5.4).

Neither this book nor the new research project on cognitive architecture could have been begun without the sabbatical support provided by USC's Viterbi School of Engineering during the 2008–2009 academic year. That year gave me the time to concentrate on these two new challenging efforts and really helped get both off to flying starts. Likewise, this book could not have been completed without the ability to write it as part of my normal faculty responsibilities or the opportunity of a second sabbatical unusually soon after the previous one. USC's Dean of Engineering, Yannis Yortsos, also provided helpful last-minute pointers to several newly published, yet highly relevant, books.

The Web in general, and Wikipedia[22] in particular, have been invaluable resources in writing this book, providing background and pointers on topics I needed to know about but with which I previously had little experience, along with specific facts, quotes, and figures that have been incorporated directly into the text as appropriate. Dictionary.com[23] also proved invaluable in clarifying concepts and providing alternative ways of expressing them, and Engadget[24] and Slashdot[25] helped keep me abreast of some of the latest discoveries and inventions. The diffuse impact of these various sources is acknowledgeable though not citable. Specific reuse is cited where appropriate.

1 The Computing Sciences

Anyone conversant with science is likely to be acquainted with the physical, life, and social sciences (figure 1.1). Each is a significant domain of intellectual endeavor that seeks to understand a broad but distinctive swath of reality. The physical sciences study physical, nonliving systems, from the smallest subatomic particles to the immensity of the universe—or multiverse, if our universe turns out to be just one among many. Disciplines within the physical sciences include physics, chemistry, astronomy, and geology. The life sciences study living organisms, which may be unicellular (e.g., bacteria), multicellular (e.g., plants and animals), or noncellular (e.g., viruses, which exist on the margin between living and nonliving). Disciplines within the life sciences include biology and ecology. The social sciences study human behavior in individuals and groups, both past and present, and both organized and not so organized. Disciplines within the social sciences include anthropology, economics, history, political science, psychology, and sociology. Together, these three scientific domains cover much of reality.

The content of each of these domains comprises a distinct combination of *structures* and *processes*. Structures are the *things of interest* in a scientific domain. In the physical sciences, a partial list includes particles and energy (physics), atoms and molecules (chemistry), stars and planets (astronomy), and rocks and rivers (geology). In the life sciences, we see everything from organic molecules to cells, organs, organisms, and social groups (for nonhuman organisms). In the social sciences, the structural focus is on people, their minds, and the artifacts that they produce and use.

Processes actively alter structures over time. In the physical sciences, we see elementary forces and their consequences (physics), chemical reactions (chemistry), the birth and evolution of the cosmos and its major components (astronomy), and the creation and evolution/erosion of land masses and bodies of water (geology). In the life sciences, we see the creation,

Figure 1.1
The three traditional great scientific domains.

evolution, and maintenance and death of organisms and their components (biology). We also see the interaction of life forms with their environments (ecology). In the social sciences, we see varying genres of cognition and behavior, such as individual and group cognition (psychology), past behavior (history), monetary behavior (economics), group behavior (sociology), and group behavior in a physical context (geography).

The distinctions among these domains are of course not without problems. For example, the question of what is life, and thus what fundamentally separates the life sciences from the physical sciences, is a difficult one. It will likely just get more difficult as humans increasingly tinker with creating life, or at least its near facsimile. The discipline of artificial life is focused on just such an enterprise, whether in organic/biological or inorganic/robotic hardware technologies or merely in software.[1] In the reverse direction, there are also emerging fields, such as DNA nanotechnology, in which core biological molecules are being used as the basis for extraordinarily tiny machines.[2] Such machines are made of the essential stuff of life but would generally not be considered alive. This kind of work may ultimately force us to rethink the boundary between these two great domains.

This is not the only such problem of course. As one further example, consider the dividing line between the life sciences and social sciences. This division reifies a form of the mind–body distinction, with the body and brain within the domain of the life sciences and the mind situated within the social sciences. The disciplines of cognitive neuroscience[3] and cognitive neuropsychology[4] are concerned with bridging this divide from its two sides by investigating how cognition arises from the workings of the brain. But they currently represent only the tip of the iceberg, as more and more of human behavior becomes firmly grounded in the workings of the body.

To some extent, such boundaries are thus social constructions,[5] although distinguishing among the domains is still useful to the extent that they differ strongly in both their core subject matter and how this material is studied and as long as we do not let such distinctions cause us to lose sight of the underlying unity of all of nature and thus of the unity we also ultimately should demand from science (what has been termed *consilience*[6]). A bigger problem from our perspective is that none of these domains covers the study of computing. Matter, energy, and force can be used to implement computing, just as they implement life, but neither computing nor life is best studied as a purely physical phenomenon. They each introduce new kinds of phenomena that enable, and in fact demand, new approaches to their study. The basic phenomena of computing are also not directly identifiable with biological organisms and the processes that support their lives or with people and the social processes that make them human. As with the physical domain, there are interesting overlaps between these domains and computing, which we will spend considerable time mapping out later in this book, but these multidisciplinary connections should not blind us to the core fact that the basic phenomena of computing are neither biological nor social.

So if computing is not part of the three traditional domains of science, what is it? One possibility is that it simply isn't a science. Some would argue that it is properly part of engineering rather than science, with more of a concern about building things than understanding them. Computer science did after all grow out of electrical engineering at a number of universities. According to this view, some science may go on in service of computing, but such *engineering science* is a subsidiary activity that does not rise to the level of a true science. A second possibility is that computing is part of mathematics—the original home for computer science at many universities where it did not come out of electrical engineering. But then mathematics has its own issues concerning whether or not it is science. A third possibility is that computing forms an additional scientific domain in its own right. This book explores this third way, hypothesizing that computing is not only a science, and the basis for a scientific domain that is distinct from the traditional three, but that it also forms the basis for what can be called a *great scientific domain*—the *computing sciences*—that is the equal of the physical, life, and social sciences.

Computing isn't simply about computers (i.e., *hardware*) nor is it just about programming (i.e., *software*) or formal principles (i.e., *theory*). Although all three of these topics are important—hardware provides the physical devices that enable computing to occur, software enables the

specification of what should be computed, and theory answers questions about what is computable and how efficiently answers can be computed—the computing sciences comprise much more than just these bare essentials. Computer science, one of the traditional disciplines within the computing sciences, also includes for example such topics as artificial intelligence (AI), with its goal of developing computational systems capable of human levels of intelligent behavior; human–computer interaction, with its focus on facilitating human use of computers; networking, with its emphasis on communication among computers; and graphics, where researchers investigate visual rendering of both real and imagined scenes. But this list still just barely begins to scratch the surface of conventional topics within computing, particularly when we look beyond the discipline of computer science to other computing disciplines, such as computer engineering, information science, informatics, and computational science.

Such a list fails even more miserably when it comes to the rich space of more exotic multidisciplinary topics currently being explored in computing and related fields, such as monkeys controlling robotic arms by pure thought (figure 1.2)[7]; quantum computers of heretofore inconceivable power[8]; technology-enabled superhuman intelligence (what is now consid-

Figure 1.2
Monkey reaches to grasp doorknob-like object via thought control of a robot arm (University of Pittsburgh).
Image courtesy of UPMC.

Figure 1.3
Simulated humans combining virtual bodies, minds, emotions, and language for virtual reality training of soldiers in cross-cultural interaction and negotiation (University of Southern California).
Image courtesy of Arno Hartholt, USC Institute for Creative Technologies.

ered as part of the *singularity*)[9]; the deep intermingling of the real and virtual worlds, as people start to interact with each other and with virtual people in computationally created environments (figure 1.3)[10]; robots capable of constructing houses automatically from blueprints and raw materials (figure 1.4)[11]; and the possibility that the entire universe is itself simply a massive computer.[12] Such notions may read like science fiction, but they are all under active investigation.

Yet, neither does computing merely consist of an ad hoc catalog of topics relating to computers, although it is possible to come away with such an impression when reading such lists or even when examining the online self-descriptions of many of our best computer science departments. Computing forms the basis for a diverse but fundamentally coherent scientific domain. It is a domain that has already made substantial contributions, both intellectual and societal; that has gone through some recent doldrums due to the burst of the dot-com bubble; but that is as exciting

Figure 1.4
Robotic gantry capable of rapid, automated extrusion of concrete walls (University of Southern California).

as ever in terms of the work that is going on and its revolutionary potential for the future.

This book comprises an extended attempt to articulate the nature, structure, current excitement, and future potential of the computing sciences, culminating with an evaluation of the hypothesis that computing is the fourth great scientific domain. The approach will be unconventional. There is a minimal introduction to how computing works, and not much at all on its history. For those who want a better introduction to its basic principles, St. Amant's *Computing for Ordinary Mortals*,[13] Hillis's *The Pattern on the Stone,*[14] and Biermann's *Great Ideas in Computer Science*[15] take complementary approaches. Denning's work "The Great Principles of Computing" is also relevant.[16] A range of books on the history of computing can be found simply by searching online via Google or at an online bookstore such as Amazon.com. Beyond these gaps, there will also be no serious attempt to justify computing's status as a science via arguments about the depth or fundamental nature of its discoveries. I am content to leave that to Peter Denning and others.[17] Nor will we focus to a great extent on the

traditional disciplinary structure within computing, except as necessary to relate it to the organizational concepts to be introduced here. What we will do is first discuss the nature of computing and the scope of the computing sciences in the remainder of this chapter before moving on to an inherently multidisciplinary, *relational approach* to understanding the domain of computing as a great scientific domain.

Understanding the nature of computing in terms of a succinct definition is essential for scoping both the field in general and this book in particular. Without such, we cannot know what to include in the computing sciences or how to argue for their amounting to a great scientific domain. We cannot even adequately explain to our scientific colleagues in other fields, to our students, or to the public at large what it is we do. The definition of computing we will settle on is conventional and mostly uncontroversial. The approach to scoping the computing sciences, in contrast, begins with a largely conventional—although still possibly controversial—proposal, and then proceeds from there through a succession of less conventional and likely more controversial alternatives.

The relational approach to understanding computing is novel in both its structure and implications. It suggests a strong parallel between what is generally viewed as core versus multidisciplinary research. It assumes and exploits a symmetry between computing and the other three great scientific domains. And, most critically, it provides a distinctly different way of understanding the individual topics within computing, how they relate to each other, and how computing as a whole relates to the other domains. These differences are leveraged to provide a new kind of multidisciplinary overview of the domain of computing, with a strong focus on what is most exciting for the present and future. They also provide insight into several forms of macrostructure that can be defined over the domain. The ultimate result is a surprisingly diverse yet coherent perspective on the domain of computing that can be leveraged, along with an excursion into philosophy of science, to evaluate the hypothesis that computing comprises the fourth great scientific domain.

1.1 What Is Computing?

Existing definitions of computing tend to focus on particular disciplines within the overall domain, such as computer science or information technology, rather than on the domain as a whole. However, there is a core motif woven through most such definitions that can be distilled out as the essence of computing as a subject for study. Let's start with definitions of

computer science, which has a long and often distinguished tradition of attempts to define it[18] dating back to at least the late 1960s.[19] Informally, computer science studies computing and information, both in the abstract and in real implemented systems. It focuses on theory (the mathematical and logical foundations of computing), algorithms (sequences of instructions to be followed to perform tasks), and all kinds of software systems and issues. It may also include work on computer hardware as well. Typical efforts to define it include "the study of (the phenomena surrounding) computers,"[20] "the study of algorithms,"[21] "the field of computer hardware and software,"[22] and "the study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems."[23] There are not as many attempts at defining the other disciplines within the computing sciences, but typical ones consider computer engineering to be "the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems and computer-controlled equipment,"[24] information science to be "the scientific study of the gathering, manipulation, classification, storage, and retrieval of recorded knowledge,"[25] and information technology to be "the development, implementation, and maintenance of computer hardware and software systems to organize and communicate information electronically."[26]

Perhaps most directly on point has been a recent symposium in the Association for Computing Machinery (ACM) online publication *Ubiquity*, which attempted to answer the question, *What is computation?* One contribution defined it as process,[27] another as symbol manipulation,[28] a third as the transformation of representations,[29] and several—including my own—focused on information transformation.[30,31] The common motif running through most of these and the earlier definitions, either implicitly or explicitly, is that computing involves the *transformation of information*. This can be expressed alternatively as *information processing* or simply referred to as *information and computation*, where computation is understood to be the process of transforming information. The particular focus in any individual definition may be on the fact that transformation is occurring, or on the abstract specification of transformations (i.e., algorithms) or on the concrete hardware and software that enable the transformations to occur in the physical world, or on the maintenance of information, or on its denotational aspects. But, no matter how this motif is modulated, it sits at the core of all of computing and thus should serve us well as a definition of the subject matter of computing.

The technical definition of information itself derives from information theory, where it is structure—patterns of binary choices—that resolves uncertainty. A single such choice amounts to one *bit* that is capable of being either 0 or 1. One bit is sufficient to resolve whether an unbiased coin comes up heads or tails. More bits can resolve larger sources of uncertainty, such as distinguishing this book's name out of the set of all possible strings of English words. But information need not be literally expressed in bits. More broadly, it can be thought of as any content expressed in some medium—whether electrical signals in wires, magnetic domains on disks, or marks on paper—that resolves uncertainty. This can consist of bits. But it can also consist of numeric values and measurements, such as the balance in a checkbook, GPS coordinates, or seismic data from an earthquake. It can consist of strings of characters—fragments of text—covering everything from short nonsense strings, such as *ax5q*, to web pages, to the contents of this entire book. It can consist of online audio and/or video files encoded in some format, such as MP3 for audio files. It can consist of knowledge about the world, such as the generalization that *all men are mortal* or the mostly true statement that *books are found in libraries or bookstores*. It can consist of models of how things work, such as a computer-aided design model of an airplane or car, or a model of circulation within the earth's atmosphere. Or it can consist of programs; that is, content that specifies sequences of operations to be performed by computers. Content can be about our world—and be more or less accurate in the process—about imaginary or virtual worlds, or about no world at all.

In some of the definitions listed earlier, the terms *symbol* or *representation* were used instead of *information*. Both of the former terms focus on the relationship between two structures, where the first denotes the second—such as when the phrase *this book* represents, or acts as a symbol for, the book you are currently reading—rather than on structures that determine choices. Yet information is usually composed of symbols and thus represents, whereas representations generally comprise information. For our purposes here, we will not worry about subtle differences among these terms, simply going with *information* as our choice; but for those interested in this topic, a more thorough treatment of the distinction between information and representation in the context of defining computing can be found in my *Ubiquity* article.[32] For those wanting a broad overview of the topic of information—including its history, theory, technology, and impact—James Gleick's 2011 book *The Information* is a good source.[33]

Transformation then is the process of executing computational operations that alter such information. Such operations could increase a

checking account balance by crediting a deposit, predict the climate later in this century, convert an audio file on an iPod into signals capable of driving speakers, or conclude by reasoning from the general rule concerning mortality and the fact that I am a man that I am mortal. The concept of transformation is general enough to subsume all of the other activities one normally thinks of as being associated with information, such as acquisition, storage, transmission, analysis, understanding, presentation, and application. Software and hardware concern the specification and execution of information transformations.

1.2 The Scope of the Computing Sciences

Given a definition of computing as the transformation of information, what then is comprised by the computing sciences? We will explore this question in two directions. First, we will look at the domain of computing in isolation, as a pure or unidisciplinary domain, and ask what is properly part of its scope. This discussion will by necessity delve partway into the question of what science is, and isn't, starting with a mildly controversial baseline and then proceeding to two proposals for moving beyond this baseline that, although they have come to be second nature to me, are likely to raise quite a few more eyebrows. We will then examine the status of multidisciplinary work that involves computing, asking to what extent such work should also be considered as part of the computing sciences.

1.2.1 Pure Computing

At a minimum, the computing sciences should include anything that applies the traditional analytical or experimental methods of science to understanding aspects of computing. Analyzing the complexity of algorithms provides a prototypical example of such an activity, but it is far from the only one. Much more pervasive is the use of experiments in understanding essentially all aspects of computing and all varieties of computing systems. However, even the classification of these kinds of activities as science is sometimes denied because of the largely non-natural, man-made origin of what is studied. Herbert Simon—another of the four founders of AI (along with Allen Newell, who was mentioned in the preface) and the sole computer scientist to have won a Nobel prize, albeit in economics—attempted to cope with this by referring to computer science as a *science of the artificial*, being the study of man-made artifacts.[34] Peter Denning has argued more recently that the modifier *artificial* is unnecessary and that computer science is simply a science like any other,

given that we have recently come to realize that computing does show up in the natural world, including in a number of ways within our own bodies.[35]

I believe that we should go even further to accept that the distinction between natural and artificial is increasingly untenable and that it is past time it were dropped as a central distinction in science. The distinction seems to originate in a tradition that God created both nature and people, but with people occupying a special position outside of, and in a dominating position over, nature. Within this tradition, everything God created is natural, along with anything else engendered by processes in nature, whereas anything created by humans is somehow outside of nature and thus artificial. A beehive when created by bees is thus considered natural, whereas a similar structure created by people would be considered artificial. If, however, as evolution implies, people are merely one more fragment of nature, then their products are as natural as anything else.

Furthermore, although it has generally been easy historically to distinguish human products from natural products, this has become—and will likely continue to become—more and more problematic as our understanding of nature continues to improve and we are increasingly able to alter it at its most fundamental levels. Consider food flavorings. Both natural and artificial flavors may consist of identical molecules, with only the sources from which their atoms are derived differing. Similarly, plants first evolved without human intervention, and then under general pressure from human selection, and now via pointed genetic modifications. Are these plants really becoming more artificial? They are still made out of the same chemical and biological ingredients as the original "natural" plants. When doctors influence stem cells to become organ cells, are the new cells natural or artificial? The body cannot tell the difference. And nanotechnology now gives us the ability to alter both the living and physical worlds at the molecular level. The future seems likely to look more and more like this, where we will need to understand a world in which human and nonhuman effects are increasingly difficult to distinguish. Thus, even in these traditional domains, the distinction between natural and artificial appears to be heading toward the intellectual scrap heap.

Whether you accept computing as a natural science, or only as a science of the artificial, or you deny the relevance of the distinction, hopefully you will accept the baseline notion that the traditional kinds of scientific activities discussed so far fit appropriately within computing as a science and thus within the computing sciences. The next two proposals are likely to be more jarring but hopefully will make as much sense.

The first proposal takes off from a notion I have come to accept as a practicing scientist that science should include all activities that *tend to increase our understanding over time*. The standard analytical and experimental methods within science are geared toward generating conclusions that are valid to a high degree of certainty. Such levels of certainty can be critical when results are to be applied in real-world situations, but the process of understanding can be more gradual in general, with progress possible by a variety of methods that vary dramatically in the certainty they provide and the kinds of understanding they yield.

When I read a scientific article, what I care about is learning something new and important that I can be convinced is true. I am agnostic, at least in principle, concerning what methods were used to invent or discover the new thing or what methods were used to convince me of their reality, as long as they achieve the desired ends. Great science requires all three of these attributes: novelty, importance, and veracity. Good science makes some compromises. To many researchers, all that is important for good science is veracity, and work may be publishable with miniscule quantities of the other two attributes as long as there is sufficient evidence of—or methodology for establishing—truth. In contrast, I often learn more from sufficiently novel and important conjectures—even before there is a great deal of evidence or methodology in their favor—leaving me more comfortable in labeling such papers as good science than in applying such a label to traditional small-but-validated results. I do not learn of necessary truth from such articles, but they may still revolutionize my way of thinking about the topic, opening up new possibilities and plausibilities never previously considered.

This distinction is surprisingly reminiscent—at least to a devotee of English literature such as me—of an interchange in Jane Austen's *Persuasion*[36] between Anne Elliot and Mr. Elliot concerning the nature of good company. The former says that she prefers "the company of clever, well-informed people, who have a great deal of conversation," to which the latter replies "Good company requires only birth, education, and manners, and with regard to education is not very nice." Mr. Elliot then goes on to state with respect to Anne's notion that "that is not good company; that is the best." What is "best" in both company and science is substance that inspires and yields new insights irrespective of either the form in which it is clothed or whether it is based on novel facts or new ways of thinking.

The broader notion of science as understanding includes not only the kinds of explorations and conjectures discussed earlier, which identify new scientific possibilities, but also replacement of complex theories with

simpler ones, even when this does not increase their ability to predict observable phenomena (a form of progress that has recently been central to my own research); and weak analytical and experimental methods that increase the likelihood of one theory over another even if the difference is not statistically significant. All of these can tend to increase our understanding over time and thus are fully part of science under this extended notion.

On the flip side, it is important that a scientific enterprise also not tend to increase our misunderstanding over time. Fortune telling, astrology, and religious prophecy can lead to correct predictions, particularly when ambiguity is combined with generous post hoc interpretations and rationalizations, but none of these ultimately has demonstrated any ability to predict better than random guessing. On the whole, they contribute much more to misunderstanding than to understanding despite occasional hits. Such activities are thus clearly not scientific, even in the presence of occasional legal rulings to the contrary.[37] Still, although the difference is clear here, there can be gray areas where it is difficult to determine whether some activity tends to increase our understanding and should therefore be considered within the scope of science. Work on a normative scientific method attempts to deal with this by prevalidating the approach taken to understanding. As long as the scientific method is used, what comes out of it will be science. The problem, of course, is that much of actual science does not proceed by such a method, and it would be greatly impoverished if it were forced to do so. Not to mention that much trivial science follows the method to the letter.

One key research-style dichotomy I have noticed among scientists of my acquaintance is whether they are more focused on *similarities* or *differences*. The former care most for identifying what is common across a range of diverse phenomena, whereas the latter are most deeply concerned with what distinguishes the phenomena. Science as a whole cannot proceed without both kinds of activities, but individual scientists often seem to fall into one camp or the other. The recognition of this distinction among research styles goes back to at least the mid-nineteenth century in biology, where it is known as *lumpers* versus *splitters*.[38]

Both this book and my earlier research on cognitive architecture identify me as a fully committed member of the lumper camp, focused on similarity and commonality. I cannot help seeing similarities as more important than differences, getting the most satisfaction out of uncovering hidden commonalities and looking to the benefits of integration over those of differentiation. This strategy is clearly a factor in the move to identify science

with the full breadth of progressive understanding activities. The commonality among such activities and how they all contribute to our knowledge of the world becomes their predominant trait, with the differences among them, although still important, becoming a subsidiary concern.

The second proposal goes even further in this lumping process, proposing that great scientific domains should really be thought of as comprising a combination of understanding plus a general constructive activity that we will refer to as *shaping*. The task of shaping the world is generally considered the realm of engineering, whether the activity involves creating something new—such as a new computer, bridge, appliance, or chemical compound—or altering some aspect of the world that already exists; for example, by upgrading a computer or retrofitting a bridge. Traditional engineering disciplines focus on shaping the physical aspects of the world, with emphases such as civil, mechanical, chemical, electrical, nuclear, environmental, aerospace, and computer engineering. However, it would be wrong to conclude from this that shaping only occurs within the physical sciences. Shaping also occurs in the life and social sciences, just under different names, such as medicine and education. The use of the term *engineering* is conventionally limited to quantitative, mathematical approaches to shaping. As aspects of shaping diverge from this, there is more resistance to calling them engineering, but all progress in shaping is of consequence—not just the aspects traditionally labeled engineering—just as all aspects of understanding are of consequence.

The proposal here starts with this broadening from the traditional notion of engineering to the full scope of shaping, but then also includes shaping as a full, closely intertwined partner with understanding in the composition of great scientific domains.[39] Science and engineering, when considered together, are two closely interlinked faces of the same activity—somewhat like the Roman god Janus (figure 1.5)—both of which deeply embrace understanding *and* shaping. Science is most particularly about understanding the world, but it is also an inherently creative activity that shapes how we think and perceive the world through the development of new concepts and theories. It also physically shapes the world in creating the experimental equipment needed to investigate the world,[40] such as the massive Large Hadron Collider (figure 1.6).

Allen Newell used to talk about how the Computer Science Department at Carnegie Mellon University evaluated faculty for promotion based not so much on their publication record but on the extent to which they *pushed around the field;* that is, on how their work shaped the future of computer science as a discipline. This could involve discoveries that alter

Figure 1.5
The two-faced Roman god Janus.

how people think and what problems they work on, but it can also involve the creation of tools—whether conceptual, theoretical, computational, or physical—that enable new directions to be pursued, new discoveries to be made, and new tools to be built. It is the tools that make science cumulative, enabling later scientists to *stand on the shoulders* of their predecessors.[41] He even stated that all that really mattered in science was the tools. If other scientists could not, or need not, make use of your contributions in pursuing their own work, it was as if your contributions never existed, much like trees falling in the forest in the absence of witnesses. According to such a view, there can be no productivity in science without shaping of the domain.

Engineering takes the dual perspective from science, being most particularly about shaping the world but also inherently concerned with understanding what is being created and the principles underlying these creative acts and their outputs. There is a constant reciprocation between science and engineering, where they perpetually challenge and stimulate each other. Science provides theories and data upon which engineering can be based. Engineering in turn poses questions for science to answer while simultaneously evaluating the validity of scientific hypotheses by assessing whether technologies based on them succeed in the manner predicted.

Figure 1.6
The Large Hadron Collider, a 17-mile-circumference particle accelerator at the European Organization for Nuclear Research (CERN) near Geneva.
Image courtesy of CERN.

Within computing, it can be extraordinarily difficult to distinguish between understanding and shaping. One might think a priori that it would map onto the difference between computer science and engineering, but it mostly does not. Despite use of the word *science* in its name, computer science is frequently considered to be an engineering discipline in addition to or instead of a scientific discipline. My own computer science department is situated within a school of engineering—the USC Viterbi School of Engineering—as are computer science departments at many other universities. The development of algorithms and software systems, along with programming in general, are clearly engineering activities; yet, computer science is also inherently about understanding computation. As a computer scientist, I have always felt ambivalent about whether to consider myself a scientist or an engineer. I create new things—concepts, theories, and systems—and sit within a school of engineering, but my drive has always been to get to the bottom of things, to understand the *how* and the *why*.

Computer engineering focuses on computer systems, primarily hardware and aspects of systems software—software that is *close to* or integrated with the hardware. It overlaps extensively with both electrical engineering—the study of the flow of electricity, including through the kinds of semiconductor materials used in computers—and computer science. Computer engineering is in some sense more engineering-like than computer science because it is closer to the physical domain and thus also more amenable to formalization via traditional mathematics, but in the same sense it is more science-like as well. The distinction between the two disciplines is thus really more between a focus on software in computer science and hardware in computer engineering. Both include substantial aspects of understanding *and* shaping.

But the problem goes beyond merely confusion in labeling to permeate much of the field. Because computing largely understands what it creates, it is difficult in general to tell when shaping leaves off and understanding begins. This is often viewed as an embarrassment, contributing to the notion that computing is not science. To articulate more clearly the breadth and depth of computing's science base, academics continue to work hard at separating out understanding from shaping. But what if the more fundamental problem instead turned out to be that we have been looking at this issue backward all of this time? In other words, what if the inherent intertwining of understanding and shaping within computing were actually a strength rather than a weakness? Furthermore, what if this meant that computing is not a problem child within the sciences but a model for the future of the other sciences in terms of the centrality of such intertwining within great scientific domains?

In computing, this intertwining of understanding and shaping has actually been one of its greatest strengths rather than a weakness for which we should feel apologetic. It is a key factor in computing's astonishingly rapid development. Think for example of Xerox PARC in its heyday, when a ferment of exploration, experimentation, and creation combined to revolutionize the dominant paradigm of computing. Although not referring to computing, Richard Feynman—a Nobel laureate physicist—captured the overall notion well in a statement on his blackboard at his death in 1988: "What I cannot create, I do not understand."[42] The life and social sciences in particular have long suffered from their limited ability to shape their domains in conjunction with their understanding of them. As our ability to create and manipulate living and thinking systems continues to improve, the life and social sciences will have an increasing opportunity, and in fact an imperative, to embrace the intertwining of understanding and shaping

that has so long been a major factor in computing. Although people have long shaped the physical domain, even there our ability to manipulate it at its most fundamental levels is taking a giant leap with the advent of nanotechnology.

We may have to wait until scientists from the other scientific domains fully appreciate both the inevitability and the power of intertwining understanding and shaping in their own work and domains before we can hope to see a broader acceptance of what computing has been both confronting and leveraging since its inception. But this may not be too far in the future, as intertwining increasingly becomes the norm across the sciences, just as it has been in computing since the beginning.

In the meantime, I have increasingly bonded to the notion that at the top level, the distinction should be between content domains—physical, life, social, and computing—rather than science versus engineering, with each such domain then being a blend of understanding and shaping. Science and engineering, as traditionally construed, are central to understanding and shaping, respectively, but are inherently more limited in presupposing particular domains of interest and methods for exploring these domains while disregarding most understanding or shaping activities that fall outside of the boundaries defined by these domains and methods. The traditional methods focus on experiments whose parameters can be controlled by the investigator, theories expressed in the language of mathematics, and more recently simulations implemented on computers. The domains of science and engineering tend to be limited to those that can be investigated via these methods. Deriving the meaning and significance of the works of Jane Austen is, for example, clearly an activity of understanding, but it is part of the humanities that is not generally considered part of science. Similarly, medicine and education shape our bodies and minds but are not traditionally part of engineering.

The notion of a great scientific domain, as it will be more fully developed later, assumes the intertwining of understanding and shaping. Defining a great scientific domain in this way is far from standard, but the centrality of this combination emerged directly out of my experience as a computer scientist. I have come to consider the computing sciences as comprising *the understanding and shaping of the transformation of information*. This subsumes the science and engineering of computing; basic and applied work with computers (including scientific computing and the many informatics disciplines); the academic disciplines of computer science and engineering; information science and technology; and the divide between academic and industrial computing. Understanding is

more central in those segments of the domain characterized by such terms as *science*, *basic*, and *academic*, whereas shaping takes the lead in those segments better characterized as *engineering*, *applied*, and *industrial*. However, it is all computing science. I occasionally find myself debating with computer science colleagues whether work on the more applied side can form the basis for good academic computing research. In my view, it clearly can, as long as it yields something sufficiently novel and important concerning computing.

In the remainder of this book, we will assume that the computing sciences span the full breadth of both understanding and shaping and argue further (in chapter 6) that they amount to a great scientific domain. If your preference is still to keep science separate from engineering or to keep each defined more narrowly than what is implied by understanding and shaping, you can mostly substitute *science* for *understanding* and *engineering* for *shaping* in what is to come and also substitute *science and engineering domains* for *great scientific domains*. The fit will not be perfect, particularly when we slip outside of the traditional boundaries of science and engineering, but many of the concepts and messages will still apply.

1.2.2 Multidisciplinary Computing

To what extent does multidisciplinary work that involves some aspect of computing belong within the computing sciences? Multidisciplinary work in general often holds a tenuous position within academic departments, sometimes accepted, sometimes accepted only grudgingly as peripherally within the discipline, and sometimes considered outside of the scope of the discipline. I still vividly recall my experience in the 1980s as a young assistant professor at Stanford University, where I had a joint appointment in computer science and psychology working on cognitive architectures via a combination of psychologically motivated AI and computational modeling of human cognition. This had been my dream job, but the reality was that I was all too often viewed by psychologists as a computer scientist because I did not do experiments nor was I concerned primarily with the accuracy of fits to human data, and computer scientists too often viewed me as a psychologist because I explored psychologically inspired mechanisms rather than more formal methods (such as logic) and was not primarily concerned with core computational issues such as optimality. My concern instead was how to develop a single integrated model/theory/system with maximal breadth of coverage across the basic phenomena of intelligence. But, in the process I had managed to situate myself outside of the predominant paradigm boundaries in both of my home departments.

When it comes to great scientific domains, similar issues arise. Should work that spans two domains be considered as fully within both domains, as only within one domain—based on criteria such as which domain most benefits from the combination—or considered to be outside of both domains? Hopefully, most people will agree that the last alternative is self-defeating, particularly as more and more of the most exciting work within computing and the other sciences becomes multidisciplinary. If the boundaries drawn around domains and the organizations that investigate them are too narrow, multidisciplinary research falls between the cracks, a result that is dysfunctional both for scientific progress in general and for the individuals involved in particular. Similar risks also result when departments, disciplines, and domains become too focused on the benefits to themselves as a criterion for whether multidisciplinary work is within their scope. The relational approach, and in fact much of the rest of this book, can be viewed as an argument for the naturalness and importance of including within the computing sciences any work that involves significant understanding or shaping of computing, whether multidisciplinary or not; and we will proceed under this assumption as we make our way through the remainder of it.

1.3 Summary

If you accept the arguments in this chapter, we have a notion of computing as the transformation of information and of the computing sciences as a multidisciplinary activity that involves the understanding and shaping of computing. This yields a broad view of computing that spans what traditionally is partitioned across science and engineering, basic and applied, computer science and engineering, information science and technology, and academic and industrial computing. It also subsumes the distinction between pure and multidisciplinary computing—and core versus peripheral computing—bringing computational science and the plethora of informatics disciplines fully into the fold. Last, but not least, it sets us up for a discussion of the importance and vitality of the science of computing. Computing is not merely the handmaiden of the other sciences, nor is it just a pragmatic art or a form of engineering; it is a great scientific domain in its own right.

2 The Relational Approach

The relational approach to understanding computing derives from a simple core idea of examining the ways in which computing relates to itself and to the other three great scientific domains. It originally grew out of a desire to determine if there was some form of coherence lurking behind the wide array of multidisciplinary topics I was working on as director of New Directions at the University of Southern California's Information Sciences Institute (ISI), ranging from automated construction to technology and the arts, biomedical informatics, simulation-based training, and responding to unexpected events. Over time, this led to the development of the relational architecture and the accompanying metascience expression (ME) language, which together provide an inherently multidisciplinary perspective on computing.

Architecture is an integrative concept. Simply put, an architecture describes how a set of parts combine to produce a functional whole. It provides structure and coherence, enabling the resulting assemblage to yield more than just the sum of its parts. In the everyday world, its most common usage is in the design and construction of buildings. Within computing, architecture is central to the design and construction of hardware. It both characterizes the hardware to be constructed and specifies the instruction set in which the hardware is to be programmed. Within cognitive science—a multidisciplinary area that combines "Artificial Intelligence, Linguistics, Anthropology, Psychology, Neuroscience, Philosophy, and Education"[1]—architecture is applied to key aspects of integrated models of cognition. In analogy to computer architecture, a cognitive architecture specifies the fixed structure underlying cognition while also specifying a language that can be used to encode knowledge—the variable part of cognition—within the system.

Architecture becomes a core tool in science by supporting the processes of understanding and shaping. In the preface to the fourth edition of their

classic text on computer architecture, Hennessy and Patterson state that their "goal has been to describe the basic principles underlying what will be tomorrow's technological developments," that their "primary objective in writing our first book was to change the way people learn and think," and that they "have strived to produce a new edition that will continue to be as relevant for professional engineers and architects as it is for those involved in advanced computer architecture and design courses."[2] The earlier classic text in computer architecture, originally by Bell and Newell, also had similar goals of providing a better understanding of the range of existing computers plus tools to assist in designing new ones.[3]

The discipline of cognitive architecture is less coherent, with the primary home for the understanding of (natural) cognition being within cognitive psychology and that for the shaping of (artificial) cognition being within artificial intelligence; although in artificial intelligence, the term *cognitive architecture* is often replaced by another term, such as *intelligent agent architecture* or *artificial general intelligence*, to avoid the *natural* connotations many place on the use of the term *cognition*. At one point, there was hope that cognitive science would be the home for work such as this that spans both natural and artificial cognition. However, its focus has narrowed to an emphasis on natural cognition, although insights are still welcome from artificial cognition as long as they bear on natural cognition.

In my earlier collaborations with John Laird and Allen Newell on the Soar cognitive architecture, we explicitly held the dual goals of understanding intelligence—whether natural or artificial—and shaping (artificial) intelligence. Our approach was based on a belief that these goals were best pursued in tandem, where they could cross-inform each other: Understanding human thought (the best working example we have of intelligence) can help guide decisions in developing artificially intelligent systems, and computationally implemented architectures can serve as key elements of unified theories of cognition[4] (integrated models of thought that combine both architectures and knowledge) to support experimentation and analysis and feed back into our understanding of the human mind. We were clearly outliers in taking such an approach but found it to yield a fertile ground within which to work.[5]

The rationale for the development of the relational architecture was to understand computing better, but the architecture itself is symmetric across domains and thus is potentially applicable in understanding any topics within science. On occasion in the forthcoming discussion, the concern will be with science in general, but mostly it will be with the architecture's application to computing. Science as a whole actually does

other domain plus a single relationship between computing and the second domain. All three are thus instances of what will be called *dyadic computing*, at the simpler end of the space of multidisciplinary forms of computing. Further comparison reveals that one pair of topics, human–computer interaction and artificial intelligence, involves the same two domains, just with different relationships, whereas a second pair, artificial intelligence and weather forecasting, shares the same relationship, differing only in which other domain is related to computing. These analyses thus reveal nonhierarchical similarities across the classes in the standard ontology.

The third step leverages these similarities to develop a new form of organization over science that is inherently multidisciplinary and nonhierarchical. When we are talking about dyadic computing, involving only one domain beyond computing and one relationship, it is possible to lay out the architecture as a two-dimensional table. Such a table will be provided later, in chapter 5, once all of the necessary groundwork for it has been presented. But understanding and presenting the full architecture requires the combinatoric flexibility of a language. Just as Bell and Newell, in writing their text, were impelled to create a new language—actually two new languages, Instruction Set Processor (ISP) and Processor Memory Switch (PMS)—in service of understanding and shaping computer architecture, a new language ultimately becomes necessary here as well. The ME language is a first stab at just such a tool.

Metascience can be thought of as *science about science*. The name of the language thus embodies the intent that it will assist in understanding and shaping the scientific enterprise. The focus here is on its use in understanding and shaping computing science, but as with the relational architecture from which it derives, the language is symmetric in how it handles the four great scientific domains. At bottom, the ME language is domain independent. Yet, because most of the experience with it to date has been in the context of computing, how useful it will prove outside of this domain—as well as whether it will require extensions adequately to model the full range of topics outside of computing—remains to be seen. Within computing, the ME language has so far proved adequate for all of the explorations in this book, although occasional discussions will come up about ways that it may at some point be worthwhile to extend it.

With the relational architecture and the ME language in hand, we can go beyond simply taxonomizing science to a deeper form of understanding and shaping of it. This can be by providing insights into the nature of individual disciplines, revealing hidden commonality across disciplines,

flocking, and bacterial growth. One of the key concepts here is *stigmergy*, where communication among organisms occurs indirectly through the effects of their actions on the environment, such as the pheromone signals left by ants (L°⇔P), rather than through direct organism-to-organism communication (L*). Collective computing is one of the two pillars of the field of *swarm intelligence*,[48] with the other being a biologically inspired counterpart that will be discussed at the end of this subsection.

These five forms of biological computation—phylogenetic, ontogenetic, immunological, neural and collective—do not exhaust the repertoire of computational processes exhibited by living systems—the processes that occur in cell metabolism and at cellular membranes are other obvious candidates, for example—but they are enough for our purposes here. What may be more interesting and relevant instead is how natural forms of biological computation are being converted into human-controlled computational processes. In some cases, existing natural computational devices, such as molecules and cells, are being harnessed to perform computations of interest to people. Work in this area is usually called *biological computing* or *biocomputing*. Depending on how much human intervention is required to make this happen, biocomputing can be considered either a form of naturally occurring computing (C/L)—as it is in table 3.2—or contrived computing (S→C/L). In both cases, however, it is grounded in real biological technology.

DNA computing, or *biomolecular computing*, is probably the most well known fragment of biological computing. It derives from ontogenetic computing, but with the goal of harnessing natural DNA transformations to perform useful work on demand. The earliest DNA computer solved the directed Hamiltonian path problem of finding a path between two nodes in a directed graph that visits each node in the graph exactly once while traveling along the links only in the directions indicated.[49] Nodes and links were represented as strings of *nucleotides*, the molecules that combine to form DNA and RNA, with the link nucleotides complementary to the node nucleotides (figure 3.9). When mixed together in large quantities, random paths through the graph became encoded in nucleotide sequences. Filtering steps, involving amplification and purification, then winnowed this large set of molecules down to those that were solutions to the problem. The initial step of generating random paths involved 10^{14} operations occurring in parallel, with the entire computation requiring seven days of laboratory work. Follow-on work has since extended this general approach to many other computational problems, including the creation of a single-molecule Turing machine,[50] a DNA computer that plays Tic-Tac-Toe,[51] RNA

Figure 3.10
Neural flight control system based on rat cortical neurons (University of Florida). Image courtesy of Thomas DeMarse, University of Florida. Figure 2 in Thomas B. DeMarse and Karl P. Dockendorf, "Adaptive flight control with living neuronal networks on microelectrode arrays," *Proceedings of the International Journal of Computation and Neural Networks*, 3 (2005): 1548–1551.

There are even attempts at building useful computational devices directly from the immune system. For example, there is a proposal to build a *biomolecular immune-computer* as a device to control portions of the natural immune system, much as chemical computers have been proposed as a means for controlling chemical reactions in situ.[55]

What we have been referring to in this section as life-inspired computing is more typically known as *biologically inspired computing*.[56] But, either way, the key is using simulations of living structures and processes as the basis for new approaches to computing (C/l). The simulation may be so approximate that the relationship to the biological system is at best metaphorical or it can strive for as close a match as current understanding and technology can enable. If we look at biologically inspired computing through the lens of the five varieties of naturally occurring biological computing identified earlier, the first topic of interest is *evolutionary computing*,[57] which takes its

Index