8. Mathematics Curriculum and the Needs of Computer Science

William L. Scherlis and Mary Shaw

In the summer of 1982, the Sloan Foundation conducted a workshop on the curriculum for the first two years of college mathematics. Scherlis and Shaw were invited to contribute a paper on the relation between computer science and mathematics, especially the support that computer science needs from the mathematics curriculum[101]. The paper is included here to elaborate the nature of our dependence on mathematics.

Although computer science is not a proper part of mathematics, it nonetheless relies heavily on mathematics for its foundations and its methods. Computer science education must depend on the mathematics curriculum for specific ideas and techniques from discrete mathematics, for an understanding of mathematical modes of thought, and for a genuine appreciation for power of abstraction. This paper is a examination of these needs, intended to initiate discussion of the implementation of appropriate mathematics curriculum.

8.1. Some Words about Computer Science

Computer science is concerned with the phenomena surrounding computers and computation; it embraces the study of algorithms, the representation and organization of information, the management of complexity, and the relationship between computers and their users. Computer science is like engineering in that it is largely a problem-solving discipline, concerned with the design and construction of systems. But the computer scientist, like the mathematician, must be able to make deliberate use of the intellectual tools of abstraction and of analysis and synthesis. The relationship between computer science and mathematics is very close and has been discussed at length in the literature. Two very interesting examinations of this relationship are presented by Arden [24] and Knuth [62].

Computer science is a mathematical discipline—so much so that the boundary between computer science and mathematics is often quite hard to pin down. While both disciplines are concerned primarily with abstract structures, computer science is not simply a branch of mathematics. It relies on skills, attitudes, and techniques derived from mathematics, but it is concerned not so much with proofs and the existence of structures as it is with algorithms and the design and organization of structures. In this sense computer science is an engineering discipline. Like engineering, it is pragmatic and empirical and is concerned with the selection, evaluation, and comparison of designs for implementation. But in computer science this
study is focused on the behavior of systems such as algorithms, computer organizations, and data representations—that is, on abstract rather than on concrete systems.

This paper addresses the mathematical component of a good undergraduate computer science curriculum. It begins by describing the general nature of the mathematical needs of computer science undergraduates and then discusses some specific mathematical topics that are particularly helpful in computer science education. These mathematical topics include not only traditional mathematical subjects that can be taught in self-contained courses, such as discrete mathematics, but also certain mathematical modes of thought that pervade computer science thinking and that cannot be taught easily on their own. In the last sections we consider the impact of these needs on the curriculum.

8.2. Mathematical Aspects of Undergraduate Computer Science

There is a persistent misconception that computer science consists merely of writing computer programs and that, as a result, the education of a computer scientist consists merely of training in skills related to coding and debugging computer programs. On the contrary, the discipline embraces principles and techniques for the design, construction, and analysis of a wide variety of complex systems. Even programming, to be successful, requires the careful application of scientific principles.

Since the principles of computer science are largely mathematical, computer science curricula must necessarily rely on support from mathematics. The traditional mathematics and applied mathematics "service" curricula, steeped as they are in continuous mathematics, do not, however, provide adequate support for computer science. The demands of computer science on mathematics are in many respects quite different from the demands of traditional scientific or engineering disciplines. The most important difference is that, to a much greater extent than in other disciplines, abstraction is an essential tool of every computer scientist, not just of the theoretician. The computer scientist is not simply a user of mathematical results; he must use his mathematical tools in much the same way as a mathematician does.

A computer science undergraduate curriculum must attempt to develop in the student an appreciation of the power of abstraction and an ability to discover abstractions suitable to new situations. This ability is what mathematicians call mathematical maturity (see [108] for further discussion). Mathematical maturity will not be fostered if mathematics is taught to computer science students as a mere skill or as an unpleasant necessity.

Like other scientific and engineering disciplines, computer science must