THE ROLE OF MATHEMATICS IN COMPUTER SCIENCE COLLEGE EDUCATION

K. Renuga Devi,
Assistant Professor,
Department of Mathematics,
Jayaraj Annapackiam College for Women (Autonomous),
Periyakulam. Theni(Dt),Tamilnadu.

D. Abinaya,
Assistant Professor,
Department of Mathematics,
Jayaraj Annapackiam College for Women (Autonomous),
Periyakulam, Theni(Dt),Tamilnadu.

B. Amalajasmine,
Assistant Professor,
Department of Mathematics,
Jayaraj Annapackiam College for Women (Autonomous),
Periyakulam, Theni(Dt),Tamilnadu.

Abstract: Nowadays mathematics is an important foundation for many science disciplines. Similarly, discrete mathematics and logic are foundations for computer based disciplines such as computer science. However, these essential foundations are often taught independently and relevant connections to computing, required to motivate the mathematics, are usually not made. Mathematics is a natural complementary discipline for learning, understanding and appreciating many fundamental computer science concepts. Accordingly, for the students benefit, foundational mathematics should be introduced early and integrated throughout the curriculum. This paper provides motivation, specific and general guidelines, curriculum structures and a representative first course for significantly enhancing the mathematical reasoning skills of computer science graduates. Over twenty years teaching foundational computing, talking to and surveying students, alumni, educators and corporate people have convinced the author that graduates of mathematically oriented programs will be better general problem solvers and software developer.

Keywords: Mathematics and Computer Science Education, software developers, Computer scientists

I.INTRODUCTION

Scientific and engineering disciplines generally are closely coupled to mathematics. The natural sciences make mathematical models of the phenomena they study; both the natural and social sciences rely on statistics to tease meaning out of raw data; engineers depend on mathematical models at all stages of system design, construction, and maintenance. The one pair of exceptions to this rule appears to be computer science and software engineering.

Practicing software developers make little use of mathematics [20, 31], and conventional wisdom says the same of computer science students. Yet it would be very strange if the relationship between computer science, software engineering, and mathematics were really as loose as it seems. At the very least it would be suspicious for computer science and software engineering to be the only non-mathematical members of the science and engineering family; at the worst it would be downright dangerous for the disciplines to reject methods that characterize the fields whose names they use. This paper argues that, although the day-to-day practice of computing often requires little if any mathematics, there are nonetheless important connections between computer science, software engineering, and mathematics. The next section discusses the roles mathematics plays in computer science, including how specific mathematical topics interact with specific computer science topics, and how mathematical reasoning complements computer science reasoning. The third section explores the role mathematics plays in computer science education and analyzes the disparity between its role in the general discipline and its role in education. A brief conclusion then summarizes the main points and their implications for computer science curricula. Although the rest of the paper focuses on "computer science," we use the term generically rather than to identify a single precise discipline: our ultimate concern is with the education of computing professionals, most of whom still receive that education through a program that identifies itself as "computer science." Our argument and conclusions apply to software engineering as well as to computer science.

Computer scientists use math in their professional lives in several ways. First, mathematics provides the theoretical basis for many subfields of computer science, and important analytic tools for others; computer scientists thus apply specific mathematical topics to specific computing problems. More generally, mathematics provides a framework for reasoning about computing and computing problems, and even more broadly, provides a mental discipline for solving those problems. Specific Mathematical Topics It is reasonably easy to identify individual pieces of mathematics that find use in specific areas of computing (e.g., "Boolean algebra can be used to manipulate conditional expressions"). What is hard is identifying an appropriate level of detail at which to analyze computing's uses of mathematics, and imposing some standard of completeness on that analysis. Note that we excluded the "general and reference" category from our analysis as orthogonal to that analysis, and "mathematics of computing" because our goal was essentially to match its elements to the other categories. The value of this approach is that it brings some objectivity to the process of aligning mathematics and computer science; the price of that objectivity is that everyone will no doubt see ways in which the alignment differs from their personal perceptions. We offer the approach as a first step in developing a comprehensive understanding of what math is important for computer science, but certainly don't expect our analysis to be the last word on the subject.

Certain computing topics also have much higher connectivity than others. Not surprisingly, "theory of computation," which includes mathematical models of computation plus analysis of algorithms, is connected to many mathematical topics. The high connectivity of "computing methodologies" is perhaps more surprising - it is due to the category being a broad one that contains many subtopics. Further comments seem appropriate for the "domain mathematics" topic linked to "applied computing." All applied computing is in some domain that has its own mathematical tools or foundations, and at some level the people involved in any applied computing project have to understand that mat hematics.

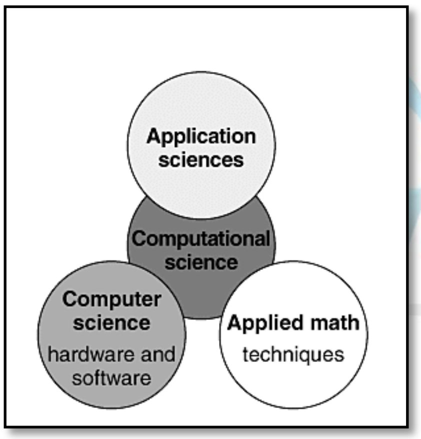


Figure.1.1. Applied computer science

Figure 1.1 showed that the importance of mathematics in computing field. Mathematics and Reasoning Many activities within computing require practitioners to analyze problems and potential solutions logically and carefully - often applying tools and techniques from mathematics. For example,

- Whenever problems are put forward or solutions proposed, users should ask what assumptions are being made and how those assumptions might impact any results obtained or program behaviors.
- When an algorithm is proposed as a solution to a problem, developers and researchers must determine whether the algorithm is correct and uses resources efficiently.
- When programs are put forward as implementations of algorithms, testing organizations and users may formally as well as empirically verify that the software behaves according to identified specifications. (Instances of required formal verification do exist - for example electronic gambling devices are subject to mathematically defined fairness requirements in some jurisdictions [22]; one of the authors recently saw a position announcement from a gaming company seeking someone to "create, test, and analyze new games" but also to "compose ... mathematical proofs for game submissions to ... regulators" [27].)
- When several potential solutions are suggested for a problem, practitioners should be able to analyze the relative advantages and disadvantages of those solutions under varying assumptions. Bruce [5] provides several specific examples of mathematical reasoning in these and other computing activities. The bottom line is that computing professionals need to reason logically - not just in hypothetical or classroom settings, but in real research and development projects. More abstractly, there are close connections between problem solving in computer science and in mathematics. Devlin [9] observes that computer science is a mass of abstractions built on other abstractions, and that mathematics is the age-old language and practice of abstraction. Ralston [24] argues that even if computing professionals seldom use math explicitly, the logical thinking central to mathematics is also central to computing. In her widely cited "computational thinking" paper [35], Jeannette Wing develops this idea in depth. She credits computer science with a distinctively powerful approach to problem solving, which, among other defining characteristics, "complements and combines mathematical and engineering thinking." The term "computational thinking" is broadly defined in her paper, and has since been applied by other authors to almost any thought process remotely associated with [P]eople who create new algorithms or designs need some ability to independently apply mathematical techniques, and at the high-math end of the spectrum, those who conduct research in an area need a deep ability to work with its mathematics.

II.MATHEMATICS AND REASONING

Many activities within computing require practitioners to analyze problems and potential solutions logically and carefully - often applying tools and techniques from mathematics. For example, verification do exist - for example electronic gambling devices are subject to mathematically defined fairness requirements in some jurisdiction one of the authors recently saw a position announcement from a gaming company seeking someone to "create, test, and analyze new games" but also to "compose ... mathematical proofs for game submissions to ... regulators".)

- When several potential solutions are suggested for a problem, practitioners should be able to analyze the relative advantages and disadvantages of those solutions under varying assumptions.

III.MATHEMATICS' ROLE IN COMPUTER SCIENCE EDUCATION

As seen in the previous section, the relationship between math- ematics and computer science has two faces: many software engi- neers perform well without relying on mathematics, while at the same time there are rich connections between the fields that can be exploited by those prepared to do so. How then does, and should, mathematics fit into undergraduate computer science curricula?

THE CURRENT STATE OF MATHEMATICS IN COMPUTER SCIENCE CURRICULAM

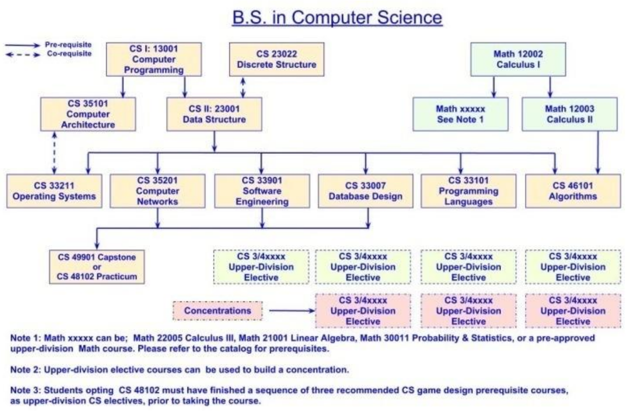
As an indication of what strong undergraduate computer science programs around the world consider appropriate mathematics content, we examined the mathematics requirements of 25 of the first 26 in all the college in tamil nadu. We emphasize that this is not a statistically rigorous study of what "typical" computer science undergraduates experience, but rather an effort to get an international selection of high-quality programs that can provide a general sense of how math is integrated into computer science education. However, the amount of math in the high-quality programs is consistent with the amount of math required in NAAC-accredited India computer science programs surveyed in the late 1990s [21], and our personal experiences suggest that observations about the high quality programs also apply to other programs.

Table 1 provides a summary of how many programs require what sorts of math. The table shows a slight inconsistency between mathematics requirements and the actual connections between math and computer science from Figure 1. Almost all the programs require students to study discrete mathematics, which is appropriate as it includes much of the foundational mathematics for computer science (e.g., logic, some proof methods, set theory, etc.). Three programs, however, do not require this foundation. Furthermore, probability and statistics is the least commonly required of the topics we looked for, despite its heavy use in computer science. Operation research, which has relatively used in computer science, is required almost as often as discrete mathematics, CONM (computer oriented numeric methods and Optimization techniques are widely used in computer science.

TABLE 1. MATHEMATICS REQUIREMENTS OF 25 HIGH- QUALITY COMPUTER SCIENCE PROGRAMS.

Topic	Number of Programs Requiring
Discrete Math	25
CONM	20
Optimization techniques	21

The number of mathematics courses required by the programs varies greatly, from a minimum of 1 to a maximum of 8, with a mode of 5. Figure 2 shows the complete distribution. Programs at the higher end of the distribution often require multiple courses in calculus, linear algebra, and/or differential equations. Very few programs require more than one course in discrete mathematics or in probability and statistics. High numbers of required math courses therefore do not indicate extensive study of the mathematics central to computer science.


B.S. in Computer Science
Pre-requisite
Co-requisite
CS I: 13001 Computer Programming
CS 23022 Discrete Structure
Math 12002 Calculus I
CS 35101 Computer Architecture
CS II: 23001 Data Structure
Math xxxxx See Note 1
Math 12003 Calculus II
CS 33211 Operating Systems
CS 35201 Computer Networks
CS 33901 Software Engineering
CS 33007 Database Design
CS 33101 Programming Languages
CS 46101 Algorithms
CS 49901 Capstone or CS 48102 Practicum
CS 3/4xxxx Upper-Division Elective
Concentrations

Note 1: Math xxxxx can be; Math 22005 Calculus III, Math 21001 Linear Algebra, Math 30011 Probability & Statistics, or a pre-approved upper-division Math course. Please refer to the catalog for prerequisites.

Note 2: Upper-division elective courses can be used to build a concentration.

Note 3: Students opting CS 48102 must have finished a sequence of three recommended CS game design prerequisite courses, as upper-division CS electives, prior to taking the course.

Figure 2: Core curriculam of computer science.

Figure. 2. Shows that 75% of the curriculum of computer science occupy by the mathematics subjects. In every computing subjects must have the mathematics to analyze the data. The prominence of calculus in computer science programs is puzzling. Some amount of calculus can be explained by the fact that one or two calculus courses are a prerequisite for other mathematics in most schools. However, many programs also require multivariable calculus, differential equations, etc., far exceeding what is plausibly necessary to study the mathematics more central to computer science. This amount of calculus may be due to programs be- ing housed in schools of engineering, or being historically derived from engineering programs, which traditionally require substantial amounts of calculus.

IV.DIFFICULTY FACED BY COMPUTER SCIENCE STUDENTS

Computer science programs require students to take a reasonable number of mathematics courses, but much of that mathematics is of limited relevance to computer science as a whole. The remaining mathematics is, on balance, under-utilized in the computer science. Within a learning environment, understanding typically starts at the beginning levels of Bloom's taxonomy—knowledge of specifics (e.g., jargon, truth tables, formal rules of logic), comprehension (e.g., paraphrasing formal rules), and simple applications. Computer science, whether in its mathematical aspects or not, is no exception. Such foundational work is needed as a base for reasoning about algorithms, programs, systems, etc. However, this elementary level of reasoning and understanding is insufficient for actually using computer science in the real world. Students must learn much more than the mechanical application of routine steps. Such learning happens when later courses build upon the foundation laid by introductory ones and provide practice at deeper levels of analysis in both structured and open-ended settings. Although such analysis may not be part of every discussion of every topic in upper-level courses, students need to experience it repeatedly and in multiple contexts. When undergraduate computer science pro- grams fail to do this with the mathematics they require, they limit graduates' ability to use mathematics in either subsequent study or employment.

Formal methods are slowly gaining traction in the software industry; of particular note, programming for concurrency is sweeping through the industry, and automatic model checking is an increasingly vital tool for coping with the subtle timing and synchronization bugs that concurrency brings [18, 8]. While there is ample need for programmers who can write code to given specifications, the more senior developers who pro- duce those specifications often need facility with the mathematics of the application domain [30]. Undergraduates who continue to graduate school, particularly at the doctoral level, will find them- selves in a world of mathematical sophistication unimaginable from the undergraduate perspective - the most pronounced ex- ample is probably the study of programming language theory as a non-mathematical descriptive activity in undergraduate texts such as [26], but as an entirely mathematical exercise in modeling language semantics in such graduate texts as [33].

V.CONCLUSION

Computer science, like the physical sciences and traditional science fields, widely uses mathematics to model the phenomena it studies. Furthermore, computational and mathematical reasoning are closely connected. Yet, paradoxically, many computer science graduates function quite well as professionals without consciously applying mathematics to their work. This paradox leads mathematics to sit uncomfortably in undergraduate computer science curricula while most such curricula include appropriate mathematics, they often also include much mathematics that is not strongly connected to computing, and while they teach some applications of math to computing, they often overlook others. This uncomfortable treatment of mathematics in computer science education has been surprisingly resistant to correction, for a surprisingly long time. While the precise reasons differ from institution to institution, we believe that the overarching one is that computer science faculty simply do not see the problem as urgent. And indeed, as long as computer science graduates find jobs or places in graduate schools in the field, and the field itself is growing, the problem does seem minor.

VI.REFERENCE

[1]. Aho, A. V. and Ullman, J. D. Foundations of Computer Science. (New York: Computer Science Press, 1992).

[2]. Baldwin, D. and Henderson, P. B. "A working group on integrating mathematical reasoning into computer science curricula." http://www.math-in-cs.org/. Accessed 2013 April 30.

[3]. Baldwin, D. and Scragg, G. Algorithms and Data Structures: The Science of Computing. (Hingham, Massachusetts: Charles River Media, 2004).

[4]. Bloom, B. Taxonomy of Educational Objectives: Handbook 1: Cognitive Domain. (Longmans, Green and Company, 1956): 201-207.

[5]. Bruce, K. et al. "Why math?" Communications of the ACM, 46, 9 (2003): 41- 44.

[6]. Cohoon, J. P. and Knight, J. C. "Connecting discrete mathematics and software engineering" in Proceedings of the Thirty-Sixth ASEE/IEEE Frontiers in Education Conference. (New York: IEEE, 2006): M2F-13 - M2F-18.

[7]. Cormen, T. et al. Introduction to Algorithms. 3rd ed. (Cambridge, Massachusetts: MIT Press, 2009).

[8]. IEEE Computer Society, "TechLeader OnCourse."

[9]. http://www.computer.org/portal/web/certifi- cation. Accessed 2013 April 30.

[10]. IEEE Computer Society and Association for Computing Machinery Interim Review Task Force. "Computer Science Curriculum 2008: An Interim Revision of CS 2001." http://www.acm.org/ education/curricula/ComputerScience2008.pdf. Accessed 2013 June 27.

[11]. IEEE Computer Society and Association for Computing Machinery Joint Task Force on Computing Curricula. "Computing Curricula 2001: Computer Science," http://www.acm.org/ education/education/education/curric_vols/cc2001.pdf. Accessed 2013 April 30.

[12]. Jhala, R. and Majumdar, R. "Software model checking." ACM Computing Surveys 41, 4 (2009). doi: 10.1145/1592434.1592438.

[13]. Kedem, Z. et al. eds. "The 2012 ACM Computing Classification System." http://www.acm. org/about/class/2012. Accessed 2013 April 20.

[14]. Lethbridge, T. "Priorities for the education and training of software engineers." Journal of Systems and Software, (2000): 53-71.

[15]. McCauley, R. and Manaris, B. "Computer science education at the start of the 21st century—a survey of accredited programs." in Proceedings of the Thirty-Second ASEE/IEEE Frontiers in Education Conference. (New York: IEEE, 2002): F2G-10 - F2G-15.

[16]. Nevada State Gaming Control Board Gaming Commission. "Regulation 14." http://gaming. nv.gov/modules/showdocument.aspx?documentid=2921 . Accessed 2013 July 17.

[17]. Ralston, A. "The first course in computer science needs a mathematics corequisite."Communications of the ACM, 27, 10 (1984): 1002-1005.

[18]. Ralston, A. "Do we need ANY mathematics in computer science curricula?" inroads—the SIGCSE Bulletin, 37, 2 (2005): 6-9.

[19]. Sahami, M. "A course on probability theory for computer scientists." in Proceedings of SIGCSE 2011, the Forty-Second Technical Symposium on Computer Science Education (New York: ACM, 2011): 263-268