01 Sep 1988

Computer Science—a Mathematical Science And Accreditation

Frank Garnett Walters
Missouri University of Science and Technology

Recommended Citation

F. G. Walters, "Computer Science—a Mathematical Science And Accreditation," *ACM SIGCSE Bulletin*, vol. 20, no. 3, pp. 53 - 56, Association for Computing Machinery (ACM), Sep 1988.
The definitive version is available at https://doi.org/10.1145/51594.51605

COMPUTER SCIENCE--A MATHEMATICAL SCIENCE and ACCREDITATION

F. GARNETT WALTERS
Associate Professor Computer Science
Department of Computer Science
University of Missouri-Rolla
Rolla, MO 65401

INTRODUCTION

During the panel discussion entitled "Recruiting more Computer Science students--What to do after the 'glamor' has gone away?" at the 1988 SIGCSE Technical Symposium held in Atlanta, I made some comments which, apparently, were not understood as intended. In particular, the comments concerning accreditation and computer science being a mathematical science. It is my intent to attempt to clarify these comments.

ACCREDITATION

My comments concerning accreditation were:

> The field of computer science will become healthy when the quality of programs in Computer Science, Information Systems, Data Processing, etc., is regulated by the Computing Sciences Accreditation Board and the sponsoring institutions.

The question which arose following my comments was:

> How will that generate more majors?

During the late seventies many high school students were "taken" with computers and computer games. They believed that their "calling" was to become a computer scientist. As this large number of students arrived at college to major in computer science, we became painfully aware of the shortage of computer science faculty. Class size increased, more classes were taught by teaching assistants, less personal attention was given to students and the research effort took a nosedive. This was an unhealthy environment in which to place the students and the faculty. As students were counseled on career opportunties, the enrollment began to moderate and level off at a place where the current faculty could handle the teaching responsibilities.

During this "explosion" of student enrollment, many programs were introduced at colleges and universities with little or no hope of supporting a new program at an acceptable level. In addition, the magic of "computer science" caused programs of questionable quality to be introduced. Thus, an oversupply of programmers was thrust upon industry at a time when the nature of the needs of industry was changing. The nature of the qualifications of a professional in the field has changed drastically over the last five to ten years. Programmers, as such, are no longer in high demand. Programming is still a skill, but not the ultimate goal of a computer science major. The future belongs to those who have a strong background in the area of algorithms and data structures and a breadth of knowledge in all aspects of computer science and related topics.

Given that my premises are correct:

1) programs were introduced at colleges and universities with little or no hope of supporting a new program at an acceptable level,
2) programs of questionable quality were introduced, and
3) the nature of the qualifications of a professional in the field has changed drastically over the last five to ten years,

then let us see what effect accreditation could have on student enrollment.

PREMISE 1

When many programs were introduced in the seventies, enrollment was high and these programs attracted many students, which helped pay the bills of the college or university. Large amounts of funds were made available because the return on the investment was great. With declining enrollment the programs are now a liability. It is difficult to justify the large expenditures with few students. Because we

are maturing, as a discipline, accreditation is becoming essential. To attract top quality students will require an accredited program. To become accredited will require a minimum-size faculty, facilities and breadth of course offering. Institutional support will have to be guaranteed before the accreditation is certified. I see many colleges and universities simply saying that the investment is too great and the computer science programs will be folded back into the department from which they emerged. Courses in computer science will continue to be taught, but no clearly identifiable degree program will exist. Students will migrate to accredited programs.

PREMISE 2

As was the case with programs with little or no support, programs of questionable quality will not receive accreditation. Here I see the problem as colleges and universities not being able to recruit and retain qualified faculty. It is true that the PhD production is on the rise, but well-qualified faculty will not go to schools without accreditation. Consequently, these programs will fail, perhaps not because of lack of support, but because of lack of quality. There will remain courses in computer science, but housed in departments other than computer science. Students will migrate to accredited programs.

PREMISE 3

As our discipline matures, we begin to look for a definition for the discipline. An attempt was made to define the discipline by a Task Force chaired by Peter Denning[1]. A draft report was presented at the SIGCSE Technical Symposium in Atlanta from which the definition proposed was:

> Computer Science and Engineering is the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application.

As we struggle with a definition, we are aware that no matter what the definition becomes, it will be, by necessity, different than the original idea first generated in the early fifties. In the fifties, we equated computer science with programming. If you could converse in many languages, you were a better computer scientist than someone with less language background. No real thought was given to how efficient we were processing the problems or data, to what could be automated or to any concept of what computing was all about. Industry is asking far more of our graduates now than those who completed their formal education in the fifties. How do we implement "quality control"? My belief is through accreditation. Once we have decided on a formal definition of our discipline, then it becomes incumbent on the field to regulate the quality of its graduates. As we stamp "computer scientist" on the "forehead" of the graduate, industry will dictate that this stamp of approval has meaning. The only guarantee of this certification will be accreditation.

MATHEMATICAL SCIENCE

My comments concerning computer science being a mathematical science were:

> The field of computer science will become healthy when the "word" is properly filtered to the news media and down to the high schools that computer science is a mathematical science and that positions are available for students who enroll in and complete an accredited program.

The question which arose following my comments was:

> How will that generate more majors?

Curriculum '78 leaves the distinct impression that computer science is non-mathematical. Students are becoming painfully aware that to be a computer science major requires a mathematical background.

My premise is that:

> the future belongs to those who have a strong background in the area of algorithms and data structures and a breadth of knowledge in all aspects of computer science and related topics.

Let us see what the effect of being a mathematical science could have on student enrollment. It is time to address the all important topic of the place of mathematics in computer science. Many articles have been written on this subject dating from 1974 to the present by such people as Knuth(1974)[2], Dijkstra(1974)[3], Ralston & Shaw(1980)[4], Ralston(1981)[5], Denning(1985)[6], Gibbs & Tucker(1986)[7], Hopcroft(1987)[8] and Berztiss(1987)[9] expounding upon the need to admit that mathematics is the basis for computer science. In fact, in 1974 Knuth[2] stated that an argument could be made that mathematics is a part of computer science. I do not propose that I am more knowledgeable than the others who have presented their views. My hope is that with repeated exposure, the message will be accepted.

Given that we accept the definition proposed by Denning's[1] Task Force, then the pertinent question becomes:

> How much of what kind of mathematics is necessary?

A very simplistic answer would be that this is dependent on the environment in which the program resides. The engineering environment would probably require the standard calculus/analytic geometry that has been taught for years. The reason being that these programs require some physics and differential calculus is desirable for that study. The liberal arts environment would probably not require the same kind of mathematics. The reason being that "arts" implies a study of different topics than "science" or "engineering". However, this simple answer does not address the question of quality control of our discipline or accreditation.

There must be a minimal level of mathematics taught to our majors. That mathematics does not necessarily mean calculus/analytic geometry. In fact, I believe, the mathematics which should be taught is the mathematical thinking process which ensures that we can become "critical thinkers" and "problem solvers". The analytical thinking process is what is important and can be taught without teaching the standard calculus/analytic geometry courses. One of my recent BS degree graduates reported that the company was critical of her work because she tried to analyze each problem before attempting the solution. I have reassured her that this is the proper way to approach the subject, not withstanding the company's views.

The definition of our discipline now being considered includes some mathematics at the level of Linear Algebra and Statistics. Also included, is some study of digital logic, circuits, switching theory, etc., which will require differing levels of mathematical background. To approach the question of

> How much of what kind of mathematics is necessary?

we need to deviate from our normal approach of "top down design" and "go to" a "bottom up" approach. That is, consider what the graduate, at each level, needs to have as mathematical reasoning and then decide where and what needs to be taught in your own environment to accomplish these desired results. Since our goal was to implement quality control, we do not need to guarantee that each finished product passes through the same process but only that the end product has certain minimum levels of understanding, no matter what intermediate steps were used in the process. This allows for the different environments of different schools to co-exist and produce meaningful graduates. As an example, suppose we want a graduate with a BS degree to know how to solve nonlinear equations in one unknown. If this is to include, say, Newton's Method, then some differential calculus is required. However, if we want only to teach Bisection, False Position and the Secant Method, no differential calculus is required, only the ability to write the equation of a straight line through two points. While I may prefer the former approach including Newton's Method, I can achieve the desired results without that one topic. The question of discrete mathematics--should it be taught, at what level should it be taught and by whom should it be taught--is a more difficult problem. If we study any formal mathematics, it should include a course or courses in discrete mathematics. When asked why study this topic, my usual response is "We deal with discrete sets of data; therefore, what could be more appropriate". It is true that some of these topics can be learned in a standard calculus/analytic geometry class. I believe that we should teach this course as early as possible, even prior to a standard mathematics course. I prefer that the student have an adequate background in college algebra before starting discrete mathematics. Therefore, discrete mathematics should be taught no later than the second semester of a student's career. The all important question of

> Who should teach this course?

can be answered with a two-prong statement. The mathematics department should teach this course, if they will assign competent faculty to the course. If this is thought of as a nondesirable assignment, then the computer science department must teach the course. In any case, the course content needs to be controlled by the computer science discipline. The economics of the campus may make it nearly impossible to introduce a course in discrete mathematics. However, a viable degree program should have the resources to cause this course to be taught. Otherwise, the program will fall into the category of no institutional support. As students recognize and realize that mathematics is necessary for success as a professional computer scientist, they will migrate to those programs which are mathematical based.

CONCLUSION:

One day we will look back at the eighties and laugh at these discussions. Then we will have matured as a discipline. Long before that day, we will be forced to answer these questions of accreditation and mathematics. We need not be unanimous in our acceptance of a definition, but we must accept the fact that computer science is a mathematical science and accredi-

tation is the only vehicle which can provide the quality control to allow us to mature. We owe it to our profession to address these questions, provide guidance for our accrediting boards (ABET/CSAB) and create an environment in which our majors can grow and become successful in their chosen profession.

REFERENCES

1. Denning, Peter J., Draft Report of the ACM Task Force on the Core of Computer Science, SIGCSE Technical Symposium, Atlanta, GA Feb. 1988

2. Knuth, Donald E., Computer Science and Its Relation to Mathematics, The American Mathematical Monthly 81, (April 1974) 323-343.

3. Dijkstra, E. W., Programming as a Discipline of Mathematical Nature, The American Mathematical Monthly 81, (June-July 1974) 608-612.

4. Ralston, Anthony and Shaw, Mary, Curriculum '78-Is Computer Science Really that Unmathematical?, Communications of the ACM 23,2, (Feb. 1980) 67-70.

5. Ralston, Anthony, Computer Science, Mathematics, and the Undergraduate Curriculum in Both, The American Mathematical Monthly 88, (Aug-Sept 1981) 472-485.

6. Denning, Peter J., The Science of Computing, What is Computer Science?, American Scientist 73, (Jan-Feb 1985) 16-19.

7. Gibbs, Norman E. and Tucker, Allen B., A Model Curriculum for a Liberal Arts Degree in Computer Science, Communications of the ACM 29,3, (March 1986) 202-210.

8. Hopcroft, John E., Computer Science: The Emergence of a Discipline, Communications of the ACM 30,3, (March 1987) 198-202.

9. Berztiss, Alfs, A Mathematical Focused Curriculum for Computer Science, Communications of the ACM 30,5, (May 1987) 356-365.

****

UNIFYING-- continued from page 52

Appendix IV
Blackjack Getbet Shell

```
Program Bjack(Input,Output);
(* comment *)
Var BET: Integer;
   Procedure Welcomemessage;
   ---
   ---
   Procedure Getbet(Var BET: Integer);
   (* Prompts user for bet amount and ensures
      that a proper amount is given *)
   ---
   ---
Begin (*Main*)
Welcomemessage;
Getbet(BET);
While BET<>0 Do
   Begin
   (* Play the hand *)
   Getbet(BET);
   End; (* While *)
End.
```

Appendix III
Blackjack Pseudo-Code

```
Initialize;
Welcomemessage;

Getbet(bet);
While bet<>0 Do
   Begin
   If needed Then Shufflecards;
   Dealthehand;
   While player isn't finished Do
      Begin
      get player's command;
      implement player's command;
      End While;
   If necessary Then handle dealer;
   Pay off bets;
   End While;

Outputstats;
```