Computer Science Curriculum Developments in the USA in the 1960s

G.K. Gupta
School of Computer Science & Software Engineering
Monash University
Clayton, Victoria 3800 Australia
(gopal@infotech.monash.edu.au)

Abstract

The discipline of Computer Science (CS) was born in the early 1960s when a number of universities started teaching courses in the new discipline. A number of important conferences held during the 1960s to discuss the nature of CS and whether CS programs should be offered in universities culminated in the development of the ACM Curriculum 68. Curriculum 68 was very influential in providing direction to academics who were struggling to support or introduce a CS program in their universities and appears to have been widely used. This paper discusses the curriculum efforts in the USA in the 1960s and reviews papers on CS education published during the period.

1. Introduction

The discipline of Computer Science (CS) was born in the early 1960s when a number of universities in the USA and in other countries started teaching courses in this new discipline. As a result the Association for Computing Machinery (ACM), in 1962, established the Curriculum Committee on Computer Science. The most important CS education milestone of the 1960s was the development of the ACM Computing Curriculum 68, a model curriculum for Computer Science, which was very influential and appears to have been widely used not only in the USA but worldwide. In this paper, several major CS curriculum activities including the following are described and reviewed:

- Conference of University Computing Centre Directors, June 1960
- ACM National Conference Panels, 1963 and 1964
- ACM Curriculum Committee on CS Preliminary Recommendations, 1965
- Stony Brook Conference, 1967
- ACM Curriculum 68

As well, there is a brief discussion of the COSINE Committee of the Commission on Engineering Education recommendations regarding computing education and a review of a number of papers on CS education published during the 1960s.

The paper is organized as follows. In the next section the early days of CS education are described. Section 3 discusses the 1963 and 1964 ACM National Conference Panel papers. Section 4 provides a summary of the 1965 Preliminary Recommendations of the ACM Curriculum Committee on CS. Section 5 reviews the papers presented at the 1967 Stony Brook Conference on CS education while Section 6 describes a number of other papers published before 1968. ACM Curriculum 68 is discussed in Section 7. Section 8 concludes the paper.

2. The Birth of the Discipline – Early 1960s

Digital computers were first used in education, research and business in the early 1950s. Everyone who became a computing professional in the early years became so through apprenticeship-type training or as a user who needed to use computers, or simply as someone fascinated by them. These people had their formal education in another discipline e.g. mathematics, physics, electrical engineering, or even humanities. The computer vendors started providing computing training in the early years but as the use of computers and demand for computer professionals grew there was an emergence of a number of private computer schools. Information about such schools in the 1950s is now difficult to find. With the commencement of publication of the *Communications of the ACM* in 1958, much more information about the state of CS education becomes available.

The earliest significant papers on CS education appear to be by Louis Fein who was a private consultant in California. He wrote an unpublished report on computing education for Stanford University in 1957 and presented a paper at the 1959 Western Joint Computer Conference (Fein, 1959a) and published two papers, one in *Communications of the ACM* (Fein, 1959b), and another in *Scientific American* (Fein, 1961). He was appointed chairman of the ACM Education Committee in 1960. Fein also organised a panel on "University Education in Information Processing" at the 1962 IFIP Congress held in Munich, Germany Fein (1962).

In his two papers of 1959, Fein explains that he had been studying the operation of university programs in computing since 1956 by holding formal and informal discussions with university administrators, computer centre directors, faculty members, students and industry representatives. He claims that about 150 universities and colleges were involved in computing although only a few had made a determined effort to select a field of interest, set up a policy and implement it. Most were feeling their way. According to Fein, the most important impact on university programs was the education program set up by IBM which involved the company "presenting" IBM 650's to 50 universities on the condition that each of these universities would offer some computing courses. IBM needed trained people to sell their computers and to run them once they were sold. Fein further notes that many universities were offering a variety of courses covering topics in design of computers, programming and applications of computers to those studying business, engineering, sciences, applied mathematics, statistics, logic etc. However, in his view, most universities were putting too much emphasis on the computing equipment and the courses were being designed as supplements to the equipment. In some cases computing courses were quickly put together to take advantage of free computing equipment offers from the vendors. Fein instead thought the scramble to get free equipment was in some cases disgraceful. He suggests instead that the equipment ought to be the supplement and argues that it should be possible to teach many computing courses without any computing equipment, just as a high-energy physics program could be run without a cyclotron.

Fein presented a recommended organisation for computing in a university. Fein appears to have been one of the first people to call the field "Computer Science" when he suggested the establishment of a Graduate School of Computer Sciences (analogous to the Harvard Business School) consisting of five departments viz. Computer Department (including the computer centre), Operations Research, Systems, Information and Communication, and Philosophy of Organisation. He recommended a role for the computer centre, presented a list of courses, research topics and equipment required showing that computer science was a discipline. He obviously had a clear vision for computer science in universities. Amusingly, Fein also discussed the difficulties in attracting good faculty and suggested that higher salaries might possibly be needed in a competitive market.

Fein (1961) presents an interesting and insightful description of the university environment in the US around 1960 regarding the politics and difficulties of establishing a new department of computer science (Fein uses the name "Synnoetics" rather than CS). He details the role of computer centre directors who were managing and often having to fund the computer centres by selling computer and programming time but were reluctant to let go of their teaching activities. Fein very strongly supports establishment of computer science (Synnoetics, in his words) departments,noting that such departments could have been established even earlier. He shows his frustration by indicating that there was already a demand (from students), ability to supply the demand (by faculty) and ability to obtain finance to establish new departments. Fein calls the teaching of computing courses by computer centres "bootlegging" operations and computer centre directors "successful bootleggers" and notes that "in the end, all bootlegging operations must go out of business". Fein presents a carefully-worked-out plan, including financials, for a CS department consisting of 55 faculty members graduating 100 undergraduates, 50 Master's and 15 doctoral students annually. He presents a list of courses including courses on Turing Machines, automatic programming, compilers, algorithms, foundations of models, non-numeric models, heuristics, advice giving, simulation, pattern recognition, formal languages, man-automation systems, problem solving, and measurements of synnoetic systems. A list of possible computing-related courses that could be offered by other university departments is also included (e.g. Computer-aided medical diagnosis by the Medical School, equipment reliability and design by Engineering, behavioural simulation

by Psychology, machine-guided taxonomy by Botany, social impact by Sociology) and a list of research topics is presented.

Perhaps the first conference dealing with university computing education (amongst other topics) was the Conference of University Computing Centre Directors held in June 1960 attended by 98 participants (Report, 1960a). The major issues at the conference were:

- What were the areas of real intellectual challenge for research in the use of computing machines?
- What sort of courses were being offered in CS and what others should be offered?
- How should a computer centre be financed and what should be its place in the university administration?
- How could a computing centre be of maximum service to the research and education activities of a university?
- How should government agencies best support university computing centres?

The computer centre directors made a number of recommendations including that a university computer centre should be administered and financed like a university library. The directors were very interested in CS curriculum, since about 90% of them had been academics before becoming computer centre directors. They recommended that the universities offer undergraduate and graduate courses in CS. Furthermore, they noted that this was the first conference to discuss detailed CS curricula. Most computer centres were offering non-credit short courses for users and some also offered courses for credits including courses on introduction to digital computation, advanced programming, even theory of automata, machine design, numerical analysis and operations research. The directors discussed, but did not agree on, whether the computer centres should themselves develop into academic departments.

Given that early computers were used primarily for numerical computations, it is not surprising to find that from the very beginning engineering educators were very interested in using computers in engineering education. Lindvall (1955) recommends that engineering education must be fundamental, but the implications of high-speed computing to engineering education were frightening. A conference with the title "The Use of Computers in Engineering Classroom Instruction" was held at the University of Michigan on April 29-30, 1960. A report of the conference is available in Report (1960b). The conference topics were: (1) How, when and where are programming and student use of computers to be placed in the undergraduate and graduate curriculum? (2) Actual use of computers in the classroom (3) Computer facilities and mechanisms for use by students and faculty members (4) Languages. The Report highlights a variety of opinions on how, when and where computing should be introduced in engineering education. The computer scientists attending the conference (including Bruce Arden, Richard Hamming, Elliot Organick and Alan Perlis) argued strongly that the computer should not be introduced to engineering students as a super slide rule. Alan Perlis from the (then) Carnegie Institute of Technology suggested that the freshman course on computers should include the following:

- Concept of notation and representation (for communicating algorithms)
- Optimisation or mountain climbing (for nonlinear problems)
- Model building and simulation
- Iteration and recursion (obtaining solutions via successive approximations)
- Inventory control over tools (generalised vs particular solutions)

- Simulating the computer (to gain insight into the working of a computer)

There was agreement that the faculty members who were teaching computing courses must know how to program a computer in the language that was being used by the students. It was felt that all faculty should have first-hand experience in programming computers. All students, at least once a semester, should acquire the complete experience of defining problems, programming, coding and running the problem on a computer. There was considerable discussion on what computing facilities were needed for the students and whether departments needed facilities of their own. At that time, a computer was a major item of capital equipment whose productivity had to be optimised. The idea that students or faculty members should be able to run their own jobs would have been considered totally unacceptable. The issue of what language should be used by students came up and there was some discussion whether ACT I or the Bell System was better (Fortran was not mentioned, perhaps it was not yet common) with consensus that a more general-purpose language should be used.

Tompkins (1963) notes that about 14,000 customer personnel participated in IBM Customer Education programs in the USA during 1962. Most of these attended one-day to five-day short courses given in one of 22 District Education Centres although some participated through correspondence. Much of the instruction in these short courses was aimed at particular machines but some was more general. In addition, IBM organised special symposia to highlight possible uses of computers. In 1960 (some references suggest an earlier date, perhaps 1957), IBM launched a graduate-level school of its own, called the Systems Research Institute (SRI) in New York. About 80 students were attending SRI in 1962. A number of branches of SRI were established, including one in Canberra, Australia. Although it is difficult to find details of SRI programs in the 1960s, it appears that courses were being offered in such diverse topics as systems architecture, computational linguistics, human factors, simulation, information theory, queuing theory, algorithm development and statistics. Some programs were short (4 weeks or less) while others took perhaps six months or longer. At least in the early days, students were encourage to develop their own program by choosing the lectures they liked. Other companies, Remington Rand, Burroughs, and NCR, also provided training.

Although some computing courses were being offered by major universities as computers were installed through out the 1950s, it was only by the late 1950s that universities started to recognise that they had a significant role to play in computing education. The first CS programs, mostly at graduate level, were introduced by a number of US universities in the late 1950s.

As noted earlier, although the first computers in the universities were installed in the early 1950s, many more were installed in the late 1950s with the availability of the UNIVAC I in 1957, IBM 650 in 1958 and the IBM scheme of donating equipment to universities. The computing environment in the early 1960s involved large computer installations at major universities. Tompkins (1963) states that 187 US universities and colleges had computers by 1962, with the most popular computer being the IBM 1620, although larger schools had the IBM 7070/7090. These computers were university resources primarily for research which often involved numerical computations. One of the major concerns of the computer centres was to use the machines effectively and to keep them running day and night given the large capital investment which they represented.

Recognizing the importance of computing education, ACM formed the Curriculum Committee on CS in 1962 as a subcommittee of the Education Committee. Graduate computing programs were starting to be introduced in US universities and thus there was considerable interest in CS curriculum. The Committee organized a panel discussion on CS

education during the 1963 ACM National Conference. The papers presented at this panel were published in the Communications of the ACM. Another panel discussion was organized for the 1964 National Conference. The Committee became an independent ACM committee in 1964 and began serious efforts to formulate a detailed curriculum. It presented preliminary recommendations in 1965 which were substantially revised and extended and published in 1968 as Curriculum 68 which included recommendations for undergraduate as well as graduate programs.

University computer centres were mostly headed by academics who also contributed to early CS education, since they were often the most knowledgeable computing academics on campus. Computing therefore was one of the few disciplines in universities that had a dual role to play in teaching and research as well as in providing a service. This worked well in the early years, since it provided the computing students opportunities to experience the practical side of computing within the university environment itself while the arrangement helped the computer centres by providing much-needed computing skills that were available in the academic departments. Academics with a serious interest in CS were often in Mathematics or EE departments. Forsythe (1967) notes that the faculty in the Department of Computer Science at Stanford held PhDs in the following fields: mathematics (4), applied science (1), electrical engineering (2), industrial administration (1) and physics (2). Conte (1964) notes that eight out of ten faculty members in the Department of Computer Science at Purdue held joint appointments with the Department of Mathematics. Computer scientists came from many backgrounds and most early computer scientists were not recognised senior scientists. Most computing pioneers, for example Alan Turing, John Von Neumann, J. Presper Eckert and John Mauchly, were unfortunately never seriously involved in CS education. This created enormous difficulties for the new discipline since it neither had the backing of a mother discipline nor the leadership of well-known scientists.

Tompkins (1963) notes that the main trends in undergraduate computing education in the early 1960s included the following:

1. Some universities only offered informal non-credit computing courses followed by use of computers for solving problems since academics in these universities thought that the intellectual content of computer programming courses was not high.
2. Some universities offered isolated credit courses in computer programming in the junior or senior year particularly in engineering, mathematics and business programs.
3. Some universities offered general introductory programming courses for wider audiences; these were often given by computer centre staff.
4. Some offered mixed undergraduate-graduate courses on programming, logic and systems design.
5. Others presented a full computing curriculum with a major in engineering, mathematics or applied mathematics.

At the graduate level, most major universities were already offering courses in computing in the early 1960s and many were offering full graduate majors leading to master's and/or doctorate degree. The graduate programs were either being offered within some established department or by an inter-departmental committee. Tompkins (1963) lists computing courses of a number of universities in the USA.

Gorn (1963) suggests that a new basic discipline called "The Computer and Information Sciences" was emerging. He then considers three questions. Firstly, what are the characteristics of this new discipline? This question Gorn answers by discussing mechanical languages and some of their properties. Secondly, how to distinguish the new discipline from others? The essential differences in Gorn's view were that of attitude and purpose (in the new discipline, the pragmatic question of the relation of symbols to users is a central issue) and that of background (a student in the new discipline must acquire basic knowledge about digital computers and basic principles of computer programming in addition to basic mathematics background). Finally, how is the new discipline expected to develop with the established disciplines? Gorn notes that although computer oriented people in different universities had appeared in different departments, this could not continue since the other disciplines that included CS would have to limit the nourishment they could afford to such a growing child. Gorn also describes three roles for the new discipline in a university: providing general education, providing advanced specialised education in a professional way and offering service "tool" courses for many disciplines just like Mathematics did.

The first CS departments were established during 1962-5 with the Purdue department having been established within the Division of Mathematical Sciences in 1962 (Conte, 1964) and the Stanford department within the School of Humanities and Sciences in January 1965 (Forsythe, 1967). CS departments were also established in many other countries in the 1960s. For example, a number of departments were established in Australia: the University of Sydney established the Basser Computing Department within its School of Physics around 1956 (the first program, a postgraduate diploma in numerical analysis and automatic computing, was started in 1959). It has been reported that a numerical analysis and computing course was being offered at the University of Sydney as early as 1947 and courses were offered at the University of Melbourne from 1955.

3. 1963 and 1964 ACM National Conference Panel

As a background to the CS curriculum discussions, it is useful to recall the state of computing in the early 1960s. Fortran was developed in 1957 which was followed by the development of Lisp, Algol 60, Cobol and APL. The PDP-1 had been introduced, the MULTICS project had started at MIT and a variety of machines were on the market. The concept of artificial intelligence had been introduced in the mid-1950s. Against this background panel discussions on CS education were held during the 1963 and 1964 ACM National Conferences. At the 1963 Panel six papers were presented describing several courses which could form the basis for a CS curriculum. These papers were published in the April 1964 issue of the *Communications of the ACM* with a critique of each paper by another educator. The aim of the 1964 panel discussion was to address complete undergraduate programs as opposed to the 1963 panel which discussed courses at different universities.

3.1 1963 Panel

In an introductory article for the 1963 Panel, Keenan (1964) discusses the nature of CS, presents an overview of the papers and attempts to define CS. He notes that it is the intellectual orientation of the investigator that makes CS different from other disciplines like mathematics, linguistics, electrical engineering or physics. He notes that a computer scientist is concerned with the following four topics:

1. Organisation and interaction of equipment constituting an information processing system

2. Development of software systems with which to control and communicate with the equipment
3. Derivation and study of procedures and basic theories for the specification of processes
4. Application of systems, software, procedures and theories of computer science to other disciplines

Keenan gives examples of the kind of work that was involved in each of these. He also notes that over 15,000 computers were in use at that time with a production rate of 500 computers a month and questions whether the ability to build computers was outstripping the ability to educate people who could make intelligent use of the machines. Keenan goes on to discuss the type of CS education that was needed and classifies education into the following five types:

1. **General education** – to educate people trained in other disciplines who did not really understand the capabilities, the limitations or the potential of the computer.

2. **Training of programmers** - the technical training was becoming less satisfactory and more programmers with university education were needed.

3. **Orientation of scientists** –engineering, physical sciences, behavioural sciences, medical sciences and other fields were becoming dependent on computers and needed CS education to be included in their curricula.

4. **Education of computer specialists** –systems programmers and people who could deal with more complex tasks in analysis and programming needed to be trained in computer science, perhaps at master's level.

5. **Development of computer scientists** – researchers and teachers in computer science were needed to carry out graduate studies with research to be suitable for these positions.

Keenan's introduction is followed by six papers presented at the 1963 Panel:

- An introductory course *Programming digital computers* by Alan Perlis of Carnegie Institute of Technology
- An introductory course *Introduction to digital computing* by Bruce Arden of University of Michigan
- Two courses in *Numerical analysis* by George Forsythe of Stanford University
- Four courses in *Logic* by Robert Korfhage of Purdue University
- One course in *Mechanical languages* by Saul Gorn of University of Pennsylvania
- Fours courses in *Logic design and switching theory* by David Muller of University of Illinois.

Perlis (1964) discusses a first course in programming which had been offered at the Carnegie Institute of Technology since 1958. The basic aims of the course were to teach programming, and to teach the nature of computers through giving the students a large number of examples in problem solving. The major themes covered in the course were: (1) structure of algorithms, (2) structure of languages, (3) structure of machines, (4) structure of programs and (5) structure of data.

Arden (1964) presents another view of computing which implied that the only aim of CS was to train programmers and analysts. He notes that a programmer-analyst was concerned with algorithms (two types of algorithms were noted, one requiring numerical approximation and another that was discrete or non-numerical in nature), languages for their expression and machines for their representation. Arden also presents a first course that surveyed relevant topics and provided, largely by examples, an introduction to programming with an emphasis on numerical algorithms. The course included examples of algorithm development and, concurrently, programming practice using actual algorithmic languages. Arden poses the question whether the first course should be a survey course or should concentrate on the development of working skills. So the issues of breadth vs. depth and the role of programming in the first course have been around since the birth of the discipline. A book based on these ideas, *An Introduction to Digital Computing*, had been published by Arden in 1962.

Forsythe (1964) presents two courses in numerical analysis designed for Stanford students, a first course (30 lectures) for freshmen and sophomores and a senior course (90 lectures) for juniors, seniors and graduate students. Forsythe notes the importance of good exercises and comments that there should be a marriage of good analysis and imaginative programming. He presents a list of possible textbooks for these courses.

Korfhage (1964) proposes a series of four logic courses, introduction to logic and algorithms, logical design, mathematical logic, and a graduate course on computability and algorithms. These were essentially theory courses.

Gorn (1964) discusses a course in programming languages which was being taught at the Moore School of Electrical Engineering at the University of Pennsylvania for first-year graduate students. Gorn notes that the course was bursting the bounds of a one semester course even though it merely covered the study of one-dimensional, purely sequential languages and their processors. The course exercises mainly concerned design symbol manipulation processors, generators, recognisers, translators, and various interpreters. Gorn presents a long list of references that were used by him in the course.

Muller (1964) discusses the place of logical design and switching theory in the CS curriculum and presents an outline of four courses. The courses included an introductory course followed by three courses — Switching Theory I, Switching Theory II, and Logical Design. It appears that at least two of these courses were for graduate students. Muller lists a number of books that could be used in the courses.

3.2 1964 Panel

The 1964 Panel included three presentations of complete undergraduate CS programs. Schweppe (1964) presented a program proposed for the University of Maryland, Conte (1964) presented the program at Purdue and Varga (1965) presented the program at Case Institute of Technology.

The program proposed at Maryland for introduction in 1965 recommended that a minimum of 30 credit hours of work be required to major in CS consisting of all the courses listed below. It was envisaged that electives would be allowed when other courses were available.

- Elementary Algorithmic Analysis (Lower division, 3 credits)
- Two Numerical Calculus Laboratories (Lower division, 4 credits)
- Language and Structure of Computers (Upper division, 3 credits)
- Logic of Digital Computers (Upper division, 3 credits)

- Non-numerical Processing (Upper division, 3 credits)
- Dynamic Modelling and Simulation (Upper division, 3 credits)
- Information Structures and Referencing (Upper division, 3 credits)
- Algorithmic Language Translation (Upper division, 3 credits)
- Numerical Analysis I (Upper division, 3 credits)
- Seminar in Computer Science (Upper division, 2 credits)

It was proposed that a student majoring in CS could choose one of several options, the three which were listed mathematics, physics and electronics. This program placed much less emphasis on theory and hardware than presentations at the 1963 conference had envisaged.

Conte (1964) provides a one-page description of the Purdue program which had been initiated in 1962. The program required undergraduates majoring in CS to take at least 24 hours of work in mathematics and a minimum of 16 in CS. The CS courses included 7 hours in programming and systems, 6 hours in numerical analysis and 3 hours in Boolean algebra and switching theory.

Conte also discusses the graduate program at Purdue and notes that there were three major areas of concentration — numerical analysis, logic and automata theory, and programming and systems. Every graduate student was required to take a balanced selection of courses from the three areas (six courses were offered in each of them) as well as from mathematics before starting work in the students chosen major area. A Masters level program which did not require as much mathematics was also available.

Varga (1964) describes an undergraduate program in computer technology that was started at Case in 1963 after it was decided that the program should be offered in the Division of Engineering rather than in the Department of Mathematics. All computing undergraduates were required to take the common engineering core curriculum for the four years, which included calculus, differential equations, chemistry, physics, mechanics, thermodynamics, etc. as well as a course in numerical methods that included an introduction to computing. In addition to the core, a student could choose to take computer engineering or numerical methods and programming. The following courses were offered:

- Digital Systems
- Sequential Machines, Circuits and Computer Design (4 courses)
- Numerical Analysis
- Advanced Programming
- Electives

Being in Engineering, the program emphasised digital circuits and hardware design. It was noted that almost all computer technology students chose the numerical methods and programming option.

These three papers provided three different approaches. The Case program was offered in Engineering and was hardware-oriented. The Purdue program was offered by the CS Department within the Division of Mathematical Sciences, with considerable emphasis on mathematics. The proposed Maryland program was developed in the Computer Science Centre of the University and is quite different from the other two.

4. 1965 Preliminary Recommendations

A number of major computing developments had taken place by 1965. These included the development of Sketchpad, Basic, large software projects like SAGE and SABRE. New hardware on the market included the low-end PDP-8, mid-range machines such as the earlier IBM System/360s, the Control Data 3000 series and larger machines such as CDC6600.

The 1965 Preliminary Recommendations were a result of about three years effort by the ACM Curriculum Committee on CS, including the 1963 and 1964 panel discussions at the ACM National Conferences as well as a one-week meeting during the summer of 1964 at The Homestead in Poughkeepsie, New York. The Preliminary Report discussed the nature of the discipline of CS and noted the following:

> Computer Science is concerned with information in much the same sense that Physics is concerned with energy; it is devoted to the representation, storage, manipulation and presentation of information. Some forms of information have been more thoroughly studied and are better understood than others; nevertheless, all forms of information - numeric, alphabetic, pictorial, verbal, tactile, olfactory are of interest to Computer Science.
>
> Mathematics too is concerned with information and its structure and this tends to confuse those not well versed in both mathematics and computer science. The mathematician is interested in discovering the syntactic relations between elements based on a set of axioms which may have no physical reality. The computer scientist is interested in discovering the pragmatic means by which information can be transformed to model and analyse the information transformations in the real world. The pragmatic aspects of this interest leads to inquiry into effective ways to represent information, effective algorithms to transform information, effective languages with which to express algorithms, effective means to monitor the process and to display the transformed information and to accomplish these at reasonable cost.

The Preliminary Recommendations were intended to be flexible enough that students receiving the bachelors degree could follow one of the paths listed below that were similar to what Keenan (1964) had proposed:

1. Undertake graduate work in CS
2. Contribute to the rapidly growing profession of systems programming
3. Work on application programming
4. Undertake graduate work in a field other than CS

The Recommendations included a list of courses and noted that all were based on courses already in existence, for example some were based on courses described at the 1963 and 1964 ACM National Conferences. The recommendations contained a catalogue-type description of each course with a list of references for each and noted that adequate texts were not available for some of them.

The Recommendations included six basic courses (three of these were required and two were highly recommended), four theory courses (one required), three numerical algorithms courses (one required and two highly recommended) and three elective courses on computer models and applications. Some 13 supporting courses, mostly in mathematics, were

included. The structure of the recommended program is shown in the table below that is reproduced from the 1965 report[1].

Courses \ Recommendations	*Computer Science*				*Supporting*
	Basic Courses	*Theory Courses*	*Numerical Algorithms*	*Computer Models and Applications*	
Required	1 Introduction to Algorithmic Processes 2 Computer Organization and Programming 4 Information Structures	5 Algorithmic Languages and Compilers	3 Numerical Calculus (or Course 7)		Beginning Analysis (12 cr) Linear Algebra (3 cr)
Highly Recommended Electives	6 Logic Design and Switching Theory 9 Computer and Programming Systems		7 Numerical Analysis I 8 Numerical Analysis II		Algebraic Structures Statistical Methods Differential Equations Advanced Calculus Physics (6 cr)
Other Electives	10 Combinatorics and Graph Theory	13 Constructive Logic 14 Introduction to Automata Theory 15 Formal Languages		11 Systems Simulations 12 Mathematical Optimization Techniques 16 Heuristic Programming	Analog Computers Electronics Probability and Statistics Theory Linguistics Logic Philosophy and Philosophy of Science

Table 1: 1965 Preliminary Recommendations

The recommendations required 15 credit points of mathematics and strongly recommended another four mathematics subjects and 6 points of physics, but no courses on computer architecture, operating systems, artificial intelligence, computer graphics or software development were included. In fact, the elective courses included only one course (Computer and Programming Systems) that dealt with software or programming. Most electives were theoretical, dealing with numerical analysis, automata and formal languages, or dealt with applications.

The recommendations also included a discussion of how this curriculum might be implemented given the diversity of structures in universities. Given that some members of the Committee held appointments within computer centres and others in Mathematics departments, it was recommended that computer scientists involved in providing computational facilities should be closely involved and cooperation with mathematics departments was encouraged. It was suggested that in some universities an independent department of computer science would be appropriate.

The aim of the Preliminary Recommendations was to give some guidance to universities already offering or about to offer an undergraduate degree in CS. The Report solicited ideas, experiences, and criticisms.

In response to the publication of the Preliminary Recommendations, two letters were published in the *Communications of the ACM*. The letter by Parnas (1966) deals with differences between university education in CS and technical training in programming. It notes that the two can be distinguished by the degree of perspective they attempt to give

[1] The course numbers show the sequence of courses in the preliminary recommendations.

their students. While a technical training student is taught only a particular language to solve a problem, a CS student needs a broad view of problem-solving techniques because a narrow view will result in a reduced ability to think in new ways and to adapt to new ideas. Parnas criticises the Preliminary Recommendations as lacking provision for the creative skills that a CS student ought to have. Parnas claims that a student will be relatively free from the limitations of a particular programming language if introduced to several, quite distinct, programming languages. He comments in detail on a number of courses in the Preliminary Recommendations. One of Parnas' recommendations is that major projects be included in the courses so as to provide students with an opportunity to work on an interesting unpublished problem.

A letter by Carlson (1966) essentially opposes the creation of the CS discipline. He claims that if a CS department was needed then one would also need departments for the railroad, automobile, airplane or television. Carlson claims that CS has no body of cohesive and consistent theory and so can hardly be classed as a discipline.

5. 1967 Stony Brook Conference

Given the concerns of the CS community regarding what should be taught in a CS program and at what level, a conference was held at Stony Brook during June 1967. The aims according to Finerman (1967c) included to discuss CS graduate programs in detail against a background where some academics were questioning the very premise of such programs. Other objectives were to discuss the size of the faculty necessary to teach the courses and perform research and the number of undergraduate and graduate courses necessary to sustain a CS program. Furthermore, the organisational structure of CS in the university was to be discussed and so was its relationship with the computer centre. The conference was not able to reach agreement that there should be separate departments of CS. Finerman (1967b) provides a good summary.

The papers presented at this conference show that many participants were concerned about the role of CS in higher education. For example, Beckman (1967) stated that CS lacks the coherence of some other disciplines. There was a fear that CS was not considered to have sufficient intellectual respectability within the academic community given the commonly expressed view that a computer was just a tool and a body of study based upon a tool was not a proper intellectual discipline. It was noted by some that objections to CS sometimes resulted from political rather than intellectual factors. The fears were not helped by Oettinger (1967), President of the ACM at that time, who noted that departments of CS had no place in the eternal scheme of things and he hoped that CS curriculum would not be full of theory about automata but would also include physics, mathematics, economics and social sciences. He also criticised the use of the word "science" in computer science. There was concern that some of the two-year and four-year college CS programs offered at that time were ill-advised.

Beckman (1967) comments that there was an enormous range of intellectual activity in computing, but most observers were not familiar with the full scope of CS. He estimated that about 100 colleges already offered programs that could be considered in the purview of CS and that the number of programs could grow to 500 within three years. Beckman presents a description of CS academic programs in 19 universities in the USA. The issue of a suitable organisational structure for CS was discussed and Zadeh (1967) from University of California, Berkeley argued that pioneering work in CS was being done in EE departments and many such departments took it as their responsibility to provide their students with extensive training in digital information processing and CS. In his view, it did not make sense to establish separate CS departments when there were established EE departments already offering computing programs.

Although many people were reluctant to support establishment of a new department, Alan Perlis and Stanley Gill were strong proponents of a separate CS department structure. In their view, CS had other interests outside Mathematics and EE which would be damaged if it was in either Mathematics or in EE. Perlis discussed the CS program at Carnegie-Mellon University and stated that CMU had appointed their first CS PhD as a faculty member. All CS PhD students at CMU were mathematics majors. The CMU PhD program consisted of logic and algebraic theory courses in the first year as well as programming languages, systems programming, complex information processing systems, and a seminar. In the second year, formal structures in CS, automata theory, design of digital computer systems, numerical analysis, optimisation techniques and linguistics courses were offered. Perlis suggested that no one should obtain a graduate degree in CS who was not able to expertly program in at least *three different* programming languages and on a large scale. He further commented that programming is at the root of CS not machines, not algorithms, not recursive function theory, not mathematical linguistics. In his view, CS existed only because of computer programming. Thus from the very early days of CS education, such views encouraged the belief that "CS = Programming".

In an interesting paper, Slamecka (1967) took a very different approach and suggested that a discipline may be viewed as a subset of knowledge which has three important characteristics: scope, depth, and structure. For it to be science, a discipline had to have a rather general principle, a phenomenon, or an entity occurring widely as a primitive in the universe or in man's conscious world. What was the denominator of computer and information science? In Slamecka's view, it was unlikely to be the computer. A complex machine can hardly be considered a principle; it may be of interest to a branch of engineering. The algorithm was also not a primitive since it was a process acting on entities or elements. Slamecka recommended that a more basic denominator was needed and it was information.

If information and symbols were the denominator, the entire process of converting a problem to its solution was information manipulation. In Slamecka's view, CS was trying to understand and control via symbols a variety of problems; few concepts are as powerful. Slamecka recommended that the discipline be called Information Science and Engineering. He described the structure of the discipline by using a classification three levels, the level of entities, the level of processes and the level of systems. He claimed that similar structures existed in other disciplines, for example chemical sciences. Applying this structure, the discipline of information science and engineering could be viewed as consisting of three theories or concerns:

1. *Theory of information* – this remained to be developed but would include the nature and properties of information including information representation, information relations, information measure, and information structure.

2. *Theory of information process* - concerned with the generation, transmission, transformation, storage and control of information. Slamecka discussed three levels of information process theory — syntactic, semantic and pragmatic.

3. *Theory of information systems* – the theory of all systems that generate, store, and/or transmit information. Slamecka noted that difficult areas in the theory of information systems lie at the semantic and pragmatic levels and these were poorly developed. The theory of systems dealing with semantic information transport and handling was also poorly developed, and so was the understanding of the pragmatics of information by its generator and user as components of systems.

Based on the structure described above, Slamecka noted that theories of information and of information process are of the nature of science while theory of information systems was engineering in its character. CS therefore straddled and united science and engineering. Slamecka discussed the graduate degree program at Georgia Institute of Technology where three formal options were available. Option 1 was theory-oriented. Option 2 comprised a program in the study of information engineering and, specifically, in the design of information processing systems. Option 3 was design-oriented and was concerned with the study and design of computer systems and utilities. Unfortunately, there appears to have been little support for Slamecka's views given that there were others who were presenting a much simpler view that CS was mostly about programming.

Tom Hull, chairman of the ACM Committee on Graduate Education, discussed PhD programs and the issue whether a BS degree in CS provided a sufficient basis to go on to graduate studies (Hull, 1967). Many participants thought a BS degree in CS was *not* a good preparation, with some participants questioning the necessity or desirability of undergraduate programs in CS. Elliot Organick remarked that throughout the conference there was a negative or apologetic or inconsiderate attitude towards undergraduate CS programs from many participants but that universities must develop the capacity to supply adequately educated computing graduates to meet the critical need of the society. Others argued that computing specialisation should come at the graduate level.

There was considerable discussion at the conference about the master's and doctoral programs. Issues considered were whether a master's degree should be developing a university scholar or an industry practitioner. Ashenhurst (1967) proposed three types of specialisations in master's programs; a pre-doctoral program and two professional programs. The professional programs were assumed to be terminal. In discussion of doctoral degree programs, consideration was also given as to whether both academic and professional degree programs should be offered. It was noted that students in the doctoral program would require a stronger mathematics background than standard courses were providing. An industry person noted that CS graduates had too narrow a background to meet the needs of the industry. Hull (1967) identified five core subject areas for PhD programs: logic design, switching theory, computer circuits and devices; computer organisation; computability, formal languages and automata; numerical mathematics and operations research. Many participants argued that all doctoral students should be exposed to some aspects of artificial intelligence and urges that a course on large systems be included.

6. Other Pre-1968 Papers

Atchison (1966) deals with the debate about the mathematics content of undergraduate CS programs although debate was still continuing about whether undergraduate CS program itself was needed. The 1965 Preliminary Recommendations had included 15 semester hours of mathematics as a requirement with several other mathematics courses highly recommended. A subcommittee was formed to deal with the issues, as many people in computing (including some with mathematics PhDs) believed that not all CS students needed the recommended mathematics while there were others who believed that far more mathematics was required. The subcommittee recommended that the 15 semester hours of mathematics should continue to be required since CS majors needed sufficient mathematics background to enter CS graduate programs. In addition, a "Discrete Mathematical Structures" course should be introduced as a required course.

In an influential paper, Forsythe (1967) addressed many of the issues that were facing the CS community in the mid 1960s. Forsythe defined CS as "the art and science of representing and processing information and, in particular, processing information with the

logical engines called automatic digital computers". He noted that the difference between engineering and CS was that "computer scientists work with a very abstract medium (information) and design systems typically far more complex in detail than most elaborate engineering systems".

Forsythe identified different groups of students to whom CS education should be directed — non-technical students, specialists in other fields, and CS specialists. He recommended that to achieve these objectives it was necessary to create a department of CS, perhaps in the school of letters and sciences. In Forsythe's view, without such a step, computer education could not even keep up with the progress in the discipline. Utilising a number of computer scientists that were scattered through other university departments would be quite ineffective in dealing with CS.

Forsythe noted that Stanford had no undergraduate program in CS and had no intention of starting one. There were 80 full-time graduate students in CS at Stanford in 1965-6 and the first two PhD degrees were awarded in 1966. Curiously enough, Forsythe's view was that a bachelor's degree in CS was a terminal degree although it could be appropriate for graduate work in another discipline. Forsythe also remarked that the 1965 Preliminary Recommendations were similar to a master's curriculum.

Elliot (1968) presents results of a survey of the coursework done by CS master's degree students at 25 US universities and concludes that all students were taking a course either in Fortran, Algol or MAD, one or two courses in numerical analysis were a standard part of all programs but business related courses were mostly missing. There was considerable emphasis on software design and construction but little on hardware design. Topics like Boolean algebra and automata theory were also part of most programs and artificial intelligence was present in many, but there were few courses on computer applications or numerical control.

The COSINE Committee of the National Academy of Engineering was formed in 1965 to recommend directions for including computing in EE curricula. The article COSINE (1968) is a condensation of one of the earlier reports (COSINE, 1967) that discussed the role EE departments needed to play in providing undergraduates with competence in computer science. The report noted that although all branches of engineering are concerned with the use of digital computer, EE had vital concern not only with the use but also with the conception, design and construction of digital computers. The report also noted that the content and underlying philosophy of basic EE courses needed modifying, particularly in circuits and systems, to interweave the use of computers for analysis and design with the development of basic theory. To achieve these aims, EE curricula needed to be made much more flexible allowing a student almost a full year of electives from courses that were comparable to those taken by CS students. The aims of such a program should be:

1. To provide students with a thorough understanding of computer systems based on fundamental principles
2. To give students relevant background in discrete mathematics
3. To give students access to a variety of subjects covering specialised and advanced aspects of CS

The Report noted that it was difficult to specify a detailed curriculum and presented subject areas rather than courses that could be covered. The recommendations included only a small number of computing courses; a number of significant areas were not included, for example, data structures, databases, computer graphics and artificial intelligence. The report recommended that an option within EE might be the best route for offering the

Computer Engineering (CE) program. The COSINE reports were influential in the development of many CE programs.

7 Curriculum 68

During the mid-1960s, the number of CS programs being offered in the USA and in other countries was growing rapidly and experience from these programs and new developments in computing led to the 1965 Preliminary Recommendations being substantially revised over the two years 1966-7 with the support of the National Science Foundation. The ACM Committee received many comments, criticisms and suggestions following the publication of the Preliminary Recommendations. The Curriculum 68 report recognised that the debate on the justification and description of CS as a discipline had continued. It also noted that the name of the discipline was still under discussion, including whether it should be called CS, or perhaps given a name that had a broader scope, for example Information Science or even Computing and Information Science. The Committee selected CS.

The Committee stated that one of its immediate objectives was to organise known material into a rational academic curriculum and provide a sense of direction to the colleges and universities for undergraduate and master's degree programs. The report also briefly discussed doctoral programs.

As pointed out earlier, Curriculum 68 was developed in an environment when CS was trying to justify itself as a discipline. Some computer scientists considered the establishment of doctoral programs to be very important for the discipline to achieve respectability. The report assumed that computer scientists engaged in providing computational facilities would be closely involved in the curriculum. The report was visionary given the environment in which it was prepared and was very influential. It was adopted by many universities in the USA and overseas and to some extent helped standardise the CS curriculum. The recommendations also encouraged publication of many textbooks since the authors could now write for the recommended courses.

The Curriculum 68 report records the Committee's concern about how much it should advocate undergraduate programs as opposed to graduate programs, but it found that both undergraduate and graduate programs were growing rapidly (the table below includes planned programs as well as actual programs) and therefore it was necessary to provide direction to both.

Degree Programs	1964-65	1968-69
BS	11	92
MS	17	76
PhD	12	38

Table 2: Planned and Actual CS Programs

The report made it clear that establishing a CS program in a college or university faced formidable difficulties since the problems of finding adequate faculty, providing adequate laboratory facilities, and developing new courses in an area that did not have many textbooks were significant.

The Committee selected a classification of CS, somewhat similar to what Slamecka (1967) had proposed:

1. *Information structures and processes* – including the representation and transformation of information structures and theoretical models for such representations and transformations. This area also included data structures, programming languages and models of computation.

2. *Information processing systems* - systems having the ability to transform information; interaction of software and hardware. This area embraced computer design and organisation, translators and interpreters, computer and operating systems and special purpose systems.

3. *Methodologies* - application areas of computing including numerical mathematics, data processing and file management, text processing, symbol manipulation, computer graphics, simulation, information retrieval, artificial intelligence, process control and instructional systems.

plus two related areas, namely mathematical sciences and physical and engineering sciences.

The 1965 Preliminary Recommendations had envisaged 16 courses; 11 were retained in Curriculum 68 while two (Computer Organisation and Programming, Algorithmic Languages and Compilers) of the others were split into two courses each and three were omitted (Combinatorics and Graph Theory, Mathematical Optimization Techniques and Constructive Logic) since they were considered outside the discipline of CS. Seven new courses were added, including a course on discrete mathematics and another on computer organisation; a total of 22.

Curriculum 68 recommended that a major in CS should consist of at least 30 semester hours of CS courses, including eight core subjects consisting of four basic courses and the first four intermediate courses, plus two electives from the remaining intermediate courses. This required that only about a quarter of a four-year degree program be devoted to CS courses and therefore the recommendations were quite flexible.

Curriculum 68 discussed the role of programming experience and asserted that *developing programming skill is by no means the main purpose of an undergraduate program in CS; nevertheless such skill is an important by-product*. The Report also recommended "true-to-life" programming projects perhaps during summer employment or by some other means.

The Committee accepted the recommendations reported in Atchison (1966) regarding the mathematics context of a CS undergraduate program. A minimum of 18 semester hours of mathematics courses (Introductory Calculus, Mathematical Analysis, Probability, Linear Algebra and two electives) was recommended. The Committee noted that undue specialization was not appropriate in the undergraduate program but technical electives could be used to develop, for example, specialisations in applied systems programming or computer organisation and design or scientific applied programming or data processing programming.

7.1 Curriculum 68 - Basic and Intermediate Courses

Curriculum 68 recommended four basic courses, nine intermediate courses, and nine advanced courses. The core curriculum, as noted earlier, consisted of four basic courses and the first four intermediate courses. The basic courses were intended to be taught primarily at the freshman-sophomore level. The proposed number of hours of lectures and laboratories each week and the number of credit hours are shown for each course. Laboratory work would be confined to three of the four basic courses and two advanced

courses. Detailed curriculum and an annotated bibliography was provided for each course. The four basic courses were:

- B1: Introduction to computing (2-2-3)
- B2: Computers and programming (2-2-3)
- B3: Introduction to discrete structures (3-0-3)
- B4: Numerical calculus (2-2-3)

These were supplemented by nine intermediate courses which were designed for junior-senior level. The first four were part of the core curriculum and were therefore required.

- I1: Data structures (3-0-3)
- I2: Programming languages (3-0-3)
- I3: Computer organisation (3-0-3)
- I4: Systems programming (3-0-3)
- I5: Compiler construction (3-0-3)
- I6: Switching theory (3-0-3)
- I7: Sequential machines (3-0-3)
- I8: Numerical analysis I (3-0-3)
- I9: Numerical analysis II (3-0-3)

These 13 courses provided a fair balance with the eight required courses including one numerical analysis course and a discrete structures course.

7.2 Curriculum 68: Advanced Courses

The advanced courses were to be offered at junior-senior level or at graduate level. The advanced courses were classified as such either because of their higher level prerequisites and required maturity or because of their concern with special applications. Detailed curriculum and an annotated bibliography were provided for each advanced course.

- A1: Formal languages and syntactical analysis (3-0-3)
- A2: Advanced computer organisation (3-0-3)
- A3: Analogue and hybrid computing (2-2-3)
- A4: System simulation (3-0-3)
- A5: Information organisation and retrieval (3-0-3)
- A6: Computer graphics (2-2-3)
- A7: Theory of computability (3-0-3)
- A8: Large-scale information processing systems (3-0-3)
- A9: Artificial intelligence and heuristic programming (3-0-3)

A8 was to deal with business data processing systems, information storage and retrieval systems, command and control systems and computer centres. A9 was to deal with computer applications that attempt to achieve goals considered to require human mental

capabilities. A3 dealt with the concerns of engineers, some of whom were still using analogue computers.

7.3 Curriculum 68 – Master's Program

The report recommended that the undergraduate preparation needed for graduate study in CS should include the following three parts:

1. Knowledge of CS including algorithmic processes, programming, computer organisation, discrete structures and numerical mathematics
2. Knowledge of mathematics including calculus and linear algebra and knowledge of probability and statistics
3. Additional knowledge of some field such as CS, mathematics, EE, library science, etc

The Committee recommended that a master's program include both breadth and depth. To obtain breadth, courses were to be taken from each of the three areas of CS (viz, Information structures and processes, Information processing systems, and Methodologies). To obtain depth, a student was to develop an area of concentration in which a thesis or project should be done. It was recommended that a master's program consist of nine courses including two courses from each the three areas of CS and further courses in mathematics or CS containing high mathematical content.

7.4 Curriculum 68 - PhD Recommendations

It was noted that doctoral programs would naturally vary from university to university depending on the interests of the faculty members in the department and therefore detailed recommendations for the doctoral program were not possible. The report referred to the Stony Brook Conference for further discussion of such programs. Following Curriculum 68, a number of solicited articles dealing with research and teaching areas relevant to doctoral programs were published. These included papers by McNaughton (1968) on theory, Kuno and Oettinger (1968) on computational linguistics, Arden (1969) on programming, and Salton (1969) on information organisation and information system design.

7.5 Curriculum 68 - Summary

Curriculum 68 was developed during the days when numerical and scientific applications accounted for the majority of computing applications but the curriculum included courses on most CS areas that appeared relevant at that time. The curriculum provided strong support to those academics wanting to offer undergraduate CS programs in an environment where some academics were still arguing that undergraduate programs in CS were not required. The curriculum was considered by some to be designed for mathematically - oriented CS areas. Programming was emphasised but did not dominate the curriculum.

8. Conclusions

This paper has described the debates that were going on in the 1960s about the nature of CS, whether it was a respectable discipline and whether universities ought to offer CS programs. If they were to offer CS programs, what should be included in them? Other issues were whether departments of CS needed to be established at universities and what role computer centres ought to play in teaching. Major developments in CS curriculum in the USA during the 1960s culminated in the ACM proposals presented as Curriculum 68. Curriculum 68 was very influential in providing direction to academics who were struggling

to support or introduce CS programs within their universities. The number of universities offering CS degree programs grew rapidly following the publication of Curriculum 68.

Acknowledgements

I wish to thank Karl Reed for carefully reading a draft and making a number of valuable suggestions.

References

ACM (1965), ACM Curriculum Committee on Computer Science, An Undergraduate Program in Computer Science - Preliminary Recommendations, *Comm. ACM*, Vol 8, No 9, pp 543-52.

ACM (1968), ACM Curriculum Committee on Computer Science, Curriculum 68: Recommendations for the undergraduate program in computer science, *Comm. ACM*, Vol 11, No 3, pp 151-97.

Arden, B. W. (1964), On Introducing Digital Computing, *Comm. ACM*, Vol 7, No 4, pp 212-14.

Arden, B. W. (1969), The Role of Programming in a PhD Computer Science Program, *Comm. ACM*, Vol 12, No 1, pp 31-7.

Ashenhurst, R. L. (1967), The Master's Program in Computing Science – A Report of the Workshop, in Finerman (1967c), pp 123-54.

Atchison, W. F. (1966), Mathematics for Undergraduate Computer Scientists, *Comm. ACM*, Vol 9, No 9, pp 662-3.

Atchison, W. F. and J. W. Hamblen, (1964), Status of Computer Science Curricula in Colleges and Universities, *Comm. ACM*, Vol 7, No 4, pp 225-7.

Beckman, F. S. (1967), Graduate Computer Science Program at American Universities, in Finerman (1967c), pp 39-60.

Carlson, J. W. (1966), On Determining C. S. Education Programs, *Comm. ACM*, Vol 9, No 3, pp 135.

Conte, S. D. (1964), The Computer Sciences Program at Purdue University, *Proc 1964 ACM National Conference*, pp L1.2.1.

COSINE (1967), Cosine Committee, Computer Science in Electrical Engineering, Commission on Engineering Education, Washington DC.

COSINE (1968), Commission on Engineering Education, Computer Science in Electrical Engineering, IEEE Spectrum, March, pp 96-103.

CUPM (1964), Committee on the Undergraduate Program in Mathematics, *Recommendations on the Undergraduate Mathematics Program for Work in Computing*, Berkeley, California.

Elliot, R. W. (1968), Master's Level Computer Science Curricula, Comm. ACM, Vol 11, No 7, pp 507-8.

Fein, L. (1959a), The Role of the University in Computers, Data Processing and Related Fields, *Proc. Western Joint Computer Conference*, San Francisco, Vol 15, pp 119-26.

Fein, L. (1959b), The Role of the University in Computers, Data Processing and Related Fields, *Comm. ACM*, Vol 2, pp 7-14.

Fein, L. (1961), The Computer-Related Sciences (synnoetics) at a University in the Year 1975, *Am Scientist*, Vol 49, No 2, pp 149-68.

Fein, L. (1962), Organiser. Panel on University Education in Information Processing, IFIP62, pp 763-765.

Finerman, A. (1967a), University Education in Computing Science: Introduction, in Finerman (1967c), pp 1-4.

Finerman, A. (1967b), University Education in Computing Science: Summary, in Finerman (1967c), pp 193-214.

Finerman, A. (1967c), Ed., *University Education in Computing Science*, ACM Monograph, Academic Press, New York.

Forsythe, G. E. (1961), Engineering Students Must Learn both Computing and Mathematics, *Journal of Engg Educ.*, Vol 52, pp 177-88.

Forsythe, G. E. (1963), Educational Implications of the Computing Revolution, in *Applications of Digital Computers*, V. F. Freibergen and W. Pragen (Eds), Ginn, Boston, pp 166-78.

Forsythe, G. E. (1964), An Undergraduate Curriculum in Numerical Analysis, *Comm. ACM*, Vol 7, No 4, pp 214-215.

Forsythe, G. E. (1967), A University's Education Program in Computer Science, *Comm. ACM*, Vol 10, No 1, pp 3-11.

Gorn, S. (1963), The Computer and Information Sciences: A New Basic Discipline, *SIAM Review*, 5, pp 150-5.

Gorn, S. (1964), Mechanical Languages: A Course Specification, *Comm. ACM*, Vol 7, No 4, pp 219-2.

Hull, T. E. (1967), The Doctoral Program in Computing Science - A Report of the Workshop, in Finerman (1967c), pp 155-68

Keenan, T. (1964), Computers and Education, *Comm. ACM*, Vol 7, No 4, pp 205-9.

Korfhage, R. R. (1964), Logic for the Computer Sciences, *Comm. ACM*, Vol 7, No 4, pp 216-8.

Kuno, S. and A. G. Oettinger (1968), Computational Linguistics in a PhD Computer Science Program, *Comm. ACM*, Vol 11, No 12, pp 831-6.

Lindvall, F. C. (1955), Computers Challenge Engineering Education, *Proc. Western Fall Joint Conference*, Los Angeles, Vol 7, pp 41-3.

McNaughton, R. (1968), Automata, Formal Languages, Abstract Switching and Computability in a PhD Computer Science Program, *Comm. ACM*, Vol 11, No 11, pp 738-40.

Muller, D. E. (1964), The Place of Logical Design and Switching Theory in the Computer Curriculum, *Comm. ACM*, Vol 7, No 4, pp 222-5.

Oettinger, A. G. (1967), Computers and Education, in Finerman (1967c), pp. 27-38.

Parnas, D. L. (1966), On the Preliminary Report of C^3S, *Comm. ACM*, Vol 9, No 4, pp 242-3.

Perlis, A. J. (1964), Programming of Digital Computers, *Comm. ACM*, Vol 7, No 4, pp 210-1.

Report (1960a), Report on a Conference of University Computing Centre Directors, Comm. ACM, Vol 3, No 10, pp 519-521.

Report (1960b), Conference Report on The Use of Computers in Engineering Classroom Instruction, *Comm. ACM*, Vol 3, No 10, pp 522-7.

Salton, G. (1969), Information Science in a PhD Computer Science Program, *Comm. ACM*, Vol 12, No 2, pp 111-7.

Schweppe, E. J. (1964), A Proposed Academic Program in the Computer Sciences, *Proc 1964 ACM National Conference*, pp L1.1.1 - L1.1.2.

Slamecka, V. (1967), The Science and Engineering of Information, in Finerman (1967c), pp 81-92.

Tompkins, H. E. (1963), Computer Education, in *Advances in Computers*, Vol 4, Academic Press, New York, pp. 135-168.

Varga, R. S. (1964), Computer Technology at Case, *Proc 1964 ACM National Conference*, pp L1.3.1-L1.3.2.

Zadeh, L. A. (1967), The Dilemma of Computer Sciences, in Finerman (1967c), pp 61-8.