Computer Science Curriculum Developments 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 curriculum efforts in the USA in the 1960s and reviews papers on CS education published during the period.

Biography

Professor Gopal Gupta completed BE(Hons) at the University of Roorkee (now, IIT Roorkee), India, a Master's from the University of Waterloo and PhD from Monash University. He is currently Professor of Computer Science, Monash University. Previously he was Professor and Head of Computer Science, James Cook University and Dean of Information Technology, Bond University. Major areas of interest: biometrics, computer security and computer science education.

Keywords: Computer Science Curriculum, Computer Science Education, Computer Science History

1. Introduction

The discipline of Computer Science (CS) emerged in the early 1960s when a number of universities started teaching courses in this new discipline. This article focuses on efforts to define a coherent curriculum, especially centered around the Association for Computing Machinery (ACM). Building on prior work by its Committee on Education, ACM, in 1962, established a separate Curriculum Committee on Computer Science. The most important CS education milestone of the 1960s was the development of the ACM Curriculum 68, a model curriculum for Computer Science, which was very influential and appears to have been widely used. In this paper, we describe CS curriculum discussions in the 1960s including major CS curriculum activities and 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 and 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 briefly discusses university computer facilities in the 1960s and Section 9 concludes the paper.

2. The Birth of the Discipline – Early 1960s

The early years of the modern computing age, 1940-1955, were dominated by concerns regarding computer hardware building and maintenance. By the mid-1950s a large number of computers were being manufactured, many were being installed in the universities, and the focus of the field started shifting to the use and training for the use of computers.

Although some not-for-credit and for-credit computing courses were being offered in many universities in the 1950s to enable their students and faculty make effective use of computers, the earliest significant papers on CS education appear to be by Louis Fein who was a private consultant in California and a passionate supporter of CS education in universities. He wrote an unpublished report on computing education for Stanford University in 1957 and published three papers (Fein 1959a, 1959b, 1961) and was appointed chairman of the ACM Education Committee in 1960. Fein also organized a panel on "University Education in Information Processing" at the 1962 IFIP (International Federation for Information Processing) Congress held in Munich, Germany (Fein, 1962).

In the two similar 1959 papers that were based on his work for Stanford University, Fein explains that he had been studying the operation of university programs in computing, data processing and related fields since 1956 by holding formal and informal discussions with university administrators, computer center directors, faculty members, students and

industry representatives. About 150 universities and colleges were already 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. The most important impact on university programs at that time was IBM selling heavily discounted IBM 650s to about 50 universities[1] on the condition that they would offer some computing courses. The universities were offering a variety of computer courses but, in Fein's view, most universities were putting too much emphasis on the computing equipment and courses were being designed as supplements to the equipment when equipment ought to be a supplement to the courses. In some cases computing courses were quickly put together to take advantage of the discounted computing equipment and the scramble to get such equipment was in some cases disgraceful.

Fein had a clear vision for computer science education in universities and he appears to have been one of the first to call the field "Computer Science" when he suggested that universities establish a Graduate School of Computer Sciences (analogous to the Harvard Business School) consisting of five departments, namely a Computer Department (including the computer center), Operations Research, Systems, Information and Communication, and Philosophy of Organization. He recommended a role for the computer center, presented a list of courses, research topics and equipment required. In Fein's view, computer science met most of the characteristics that were required for it to be a discipline.

Fein (1961) presents an interesting and insightful description of university politics and of the difficulties of establishing a new department in a US university around 1960. He very strongly supports establishment of university CS departments and 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 details the role of computer center directors who were managing and often having to fund the computer centers by selling computer and programming time but were reluctant to let go of their teaching activities. Fein calls the teaching of computing courses by computer centers "bootlegging" operations and computer center directors "successful bootleggers" and notes that "in the end, all bootlegging operations must go out of business". He presents a carefully worked out plan, including financials, for a CS department (to be called "Synnoetics") which would in 1975 consist of 55 faculty members graduating 100 undergraduates, 50 master's and 15 doctoral students annually. A list of courses is presented 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, and problem solving. A list of possible computing-

[1] Although Fein(1959a) notes that IBM "presented" over 50 computers to universities and Knuth (1972) states that IBM donated about 100 "free" computers, it appears that the computers were not free.

related courses that could be offered by other university departments is also included and so is a list of research topics.

In spite of Fein's strong views about the discipline of computer science, the emerging discipline was suffering an identity crisis in the early 1960s (Ceruzzi, 1988). Some academics thought it was a branch of mathematics, others thought it was electrical engineering while yet others thought it was a new discipline. This identity crisis led to considerable confusion about what Electrical Engineering and Mathematics departments should be doing not only to provide computing knowledge to their own students but also to meet the growing manpower needs of the computing industry.

As noted earlier, many universities were teaching computing courses. These courses, some for credit and others non-credit, included courses in numerical analysis, introductory programming, logic design and switching circuits, offered by Electrical Engineering, Mathematics and by the computer centers. Harvard, M.I.T., Pennsylvania and Illinois had a head start and were the first to offer computing courses but there were many others. Tomkins (1963) provides some details of computing curricula at these universities and at a number of others (including Michigan and Purdue) and lists another ten (including UCLA, Berkeley, Columbia, and Wisconsin) that were offering computing courses. Special summer school sessions were held at the University of Michigan and a 3-year Ford Foundation Project on the Use of Computers in Engineering Education beginning in the fall of 1959 at Michigan was designed to cope with the faculty training problem. This project sponsored workshops, summer terms, visiting professorships, lectures, surveys and preparation of teaching materials. A total of 217 faculty members from 65 universities participated in workshops and summer terms (Katz and Organick, 1960-61, Tomkins, 1963). Some other similar programs were supported by NSF.

Because of the involvement of the electrical engineers and the computer centers, two conferences dealing with university computing education amongst other topics were held in 1960. The first of these was about the use of computers in engineering education held in April 1960 and attended by 35 people (Report, 1960a). The second conference was the conference of computing center directors attended by 98 people, held in June 1960 (Report, 1960b).

The engineering education conference was also attended by a number of computing academics including Bruce Arden, Richard Hamming, Elliot Organick and Alan Perlis. The conference highlighted a diversity of opinions amongst the participants on how, when and where computing should be introduced in engineering education. For example, Perlis believed the introductory course should be more about problem solving and algorithms while Arden thought computing could be used to motivate students' ideas in mathematics. Hamming's view was that the course should focus more on the whole spectrum of applications including those that were not mathematical.

The directors' conference discussed CS curricula and research and how the computer centers should support university teaching and research. The computer center directors made a number of recommendations including that a university computer center being an essential

part of a university be administered and financed like a university library. The directors were very interested in CS curriculum, since about 90% of them were academics, many got involved in computing via their research. 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 centers 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 agreed that CS was a discipline in its own right. They discussed, but did not agree, whether the computer centers should themselves develop into academic departments.

In the early 1960s, the name "Computer Science" started to gain currency in the USA although not all academics agreed with the name. In a talk to engineers in March 1961, George Forsythe said that "the name Computer Sciences is being attached to the discipline, as it emerges" (Forsythe, 1961). He noted that the study of the discipline involved the theory of computer programming, algorithms, numerical analysis, data processing, and the design of computer systems. Forsythe further noted that it was imperative that all students of engineering become intimately acquainted with computers in their first year and that the computing courses be laboratory courses taught by "computer experts".

Recognizing the importance of computing education, the ACM formed a Curriculum Committee on CS in 1962 as a subcommittee of the Education Committee. Graduate computing programs were starting to be introduced in universities and thus there was considerable interest in CS curriculum. A panel discussion on CS education chaired by George Forsythe had been held at the 1961 ACM National Conference. The Committee now organized a panel discussion during the 1963 ACM National Conference and another at the 1964 National Conference. The Committee became an independent committee in 1964 and presented preliminary recommendations in 1965 which were substantially revised and extended and published in 1968 as Curriculum 68.

The computing environment in the early 1960s involved large computer installations at major universities. By 1962, 187 US universities and colleges had computers (Tompkins, 1963). These computers were university resources for research, which often involved numerical computations. One of the major concerns of the computer centers was to use the machines effectively and keep them running day and night given the large investment in them. Tomkins also notes the establishment of the IBM Systems Research Institute (SRI) in 1960 in New York (some references suggest 1957). Although the exact number of Institutes established is not known, SRIs were established in Europe, Latin America and Australia in addition to those in the USA. The first SRI had six departments: Systems, Applications, Programming Languages and Techniques, Machine Organization and Logic, Mathematics and Statistics, and Operations Research (Tomkins, 1963). SRI was offering courses 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 six months or longer. IBM was also

offering customer education programs. 14,000 customer personnel participated in these programs during 1962 (Tomkins, 1963).

Further discussions about the name and nature of the discipline continued throughout the 1960s (Ceruzzi, 1988). For example, a panel discussion on university education in information processing at IFIP62 could not agree on a definition of the discipline. Saul Gorn from Pennsylvania suggested that the discipline was "the study of the synthesis and analysis of mechanical languages and their processors" while others thought the discipline consisted of information storage, retrieval, sorting and transformation. There was a clear difference of opinion between the Europeans and the Americans with the Europeans believing that information processing was not and should not be a new discipline while the Americans thought that it was emerging as a new basic discipline. The European universities therefore were reluctant to offer courses in the emerging discipline (Fein, 1962).

Gorn suggested a new name, "Computer and Information Sciences", for the discipline (Gorn, 1963). In his view, the characteristics of this new discipline were the mechanical languages and some of their properties. The discipline could be distinguished from others by its attitude and purpose since in the new discipline, the pragmatic question of the relation of symbols to users was a central issue. Another important difference was that of background as a student in the new discipline had to acquire basic knowledge about digital computers and basic principles of computer programming in addition to basic mathematical background. Finally, 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.

Although the Division of Computer Science within the Mathematics Department at Stanford was established in 1961 (Knuth, 1972), the first CS department was established at Purdue in 1962 (Rice and Rosen, 2004). A number of universities now started establishing similar departments. CS departments were established at Miami, Wisconsin, Illinois and North Carolina universities soon after[2]. Forsythe (1963) discusses the importance of universities setting up CS departments as soon as they could hire suitable faculty which was "astonishingly hard" to find. He notes that CS was not part of Mathematics and was about halfway between Humanities and Sciences and Engineering. Forsythe presents a list of topics that should be included in CS and describes how about a dozen US universities

[2]A number of departments were also established in Australia — the University of Sydney established the Basser Computing Department within its School of Physics around 1956. It has been reported that a numerical analysis and computing course was being offered at Sydney as early as 1947 and courses were being offered at the University of Melbourne from 1955.

were offering integrated CS graduate programs under several organizations. These organizations included interdepartmental programs as well as a number of programs in Mathematics.

3. 1963 and 1964 ACM National Conference Panels

At the 1963 ACM National Conference, a panel discussion on CS curriculum was held. Six papers were presented describing courses which could form the basis of a CS curriculum. Other educators were asked to write critiques of these papers. The papers and their critiques were published in the April 1964 issue of the *Communications of the ACM*. 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.

In addition to the courses, the name and nature of the discipline was also discussed at the 1963 Panel (Keenan, 1964). In Keenan's view, it was the intellectual orientation of the investigator that made CS different from other disciplines like mathematics, linguistics, electrical engineering, or physics and he notes that a computer scientist was concerned with the following four topics:

1. Organization 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: general education, training of programmers, orientation of scientists, education of computer specialists and development of computer scientists.

Keenan's introduction is followed by six papers:

- An introductory course *Programming digital computers* by Alan Perlis of Carnegie Institute of Technology (Perlis, 1964)
- An introductory course *Introduction to digital computing* by Bruce Arden of the University of Michigan (Arden, 1964)

- Two courses in *Numerical analysis* by George Forsythe of Stanford University (Forsythe, 1964)
- Four courses in *Logic* by Robert Korfhage of Purdue University (Korfhage, 1964)
- One course in *Mechanical languages* by Saul Gorn of the University of Pennsylvania (Gorn, 1964)
- Fours courses in *Logic design and switching theory* by David Muller of the University of Illinois. (Muller, 1964)

Perlis discusses a first one-semester course in programming which had been offered at Carnegie 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: structure of algorithms including iteration and recursion; structure of languages including flowcharts and Algol grammar; structure of machines including introduction to translator and number systems; structure of programs including the distinction between programs and algorithms; and structure of data including vectors and matrices. Perlis also presents a proposal for an undergraduate option in computing within a department of mathematics program which includes a total of only six computing courses in the four-year program (programming, switching theory, computer systems, numerical analysis, logic and theory of computation).

In Arden's view, the only aim of CS was to train programmers and analysts. He notes that a programmer-analyst was concerned with numerical and non-numerical algorithms, languages for their expression and machines for their representation. He then presents a first course that surveyed relevant topics and provided, largely by examples, an introduction to programming with an emphasis on numerical algorithms. He poses the question: should the first course be a survey course, or should the principal concern be 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. In a critique of the course, C. C. Gotlieb questions the compression of computer systems, numerical analysis and programming into one course.

Forsythe presents two courses in numerical analysis: a first course (30 lectures) for freshmen and sophomores requiring a programming prerequisite and a senior course (90 lectures) for juniors, seniors and graduate students requiring a programming course and considerable mathematics background as prerequisites. Forsythe notes the importance of good exercises and comments that there should be a marriage of good analysis and imaginative programming. In a critique of the courses a number of suggestions are made for improving them. Korfhage proposes a series of four theory courses — introduction to logic and algorithms, logical design, mathematical logic, and a graduate course on computability and algorithms. In a critique of these courses, Hao Wang of Harvard suggested that the first course be split in two, one devoted to logic, the other to algorithms.

Gorn describes a course in programming languages for first-year graduate students. A prerequisite course on introduction to digital computers was required. The course was concerned mainly with having the students design symbol manipulation processors, generators, recognizers, translators, and various interpreters. Muller on the other hand 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 which covers computer organization followed by three courses — Switching Theory I, Switching Theory II, and Logical Design. The logical design course includes arithmetic operations and a case study in system design. The last two courses were for graduate students.

Almost all the courses described above were packed with too much material since most university programs that included computing (e.g. mathematics) included only a small number of computing courses. Also, the course descriptions included textbook references which indicated that CS textbooks on many topics were becoming available.

The 1964 Panel included three presentations of complete undergraduate CS programs. Schweppe (1964) presented a program proposed to start at the University of Maryland in Fall 1965. Conte (1964) presented the CS program at Purdue and Varga (1964) presented the undergraduate program at Case Institute of Technology. These provided three different approaches. The Case computer technology program was based on the belief that a computer technology undergraduate program had to be closely related to engineering and the program was therefore offered within the Division of Engineering. It allowed students a choice of either of the two branches: computer engineering or numerical methods and programming. Purdue CS Department was offering undergraduate and graduate degree programs. The undergraduate program included a minimum of 24 hours of mathematics and 16 hours of 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. Three major areas of concentration (numerical analysis, logic and automata theory, and programming and systems) were available at the graduate level with most students entering the graduate program after an undergraduate mathematics degree. The proposed Maryland program was developed in the Computer Science Center of the university and included courses for both undergraduate and masters programs. The programs were quite different from the other two. They emphasized an algorithmic approach to solving problems and included courses like algorithmic methods, computational laboratory, numerical calculus, language and structures of computers, and non-numerical processing. Majors in CS were required to take at least 30 hours of CS courses.

4. 1965 Preliminary Recommendations

By the mid-1960s, it was becoming clear that the name “Computer Science” had overwhelming support within the academic computing community. In fact a large number of universities were already offering or planning to offer undergraduate CS or related programs (Atchison and Hamblen, 1964). Given that the work of the ACM Curriculum

Committee on CS was being overtaken by developments on the campuses, the Committee decided to provide some preliminary guidelines.

These Recommendations were the result of about three years effort by the ACM Curriculum Committee on CS. The report presents the following about the nature of the discipline:

> 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. (ACM, 1965)

The Preliminary Recommendations included five required CS courses, four highly recommended CS courses and a selection of seven CS electives so that the programs based on the recommendations were flexible enough that graduating students could undertake graduate work in CS or in another field or join the industry to work on systems programming or application programming.

The Recommendations presented a list of courses that were based on courses in existence, for example some were based on courses described at the 1963 and 1964 ACM National Conferences. The report also 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 structure of the recommended program is shown in the table below that is reproduced from the 1965 report[3].

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)
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	13 Constructive Logic		11. Systems	Analog Computers

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

	and Graph Theory	14 Introduction to Automata Theory 15 Formal Languages		Simulations 12 Mathematical Optimization Techniques 16 Heuristic Programming	Electronics Probability and Statistics Theory Linguistics Logic Philosophy and Philosophy of Science

These recommendations did not include courses on computer architecture, operating systems, artificial intelligence, computer graphics or software development.

5. 1967 Stony Brook Conference

Originally the aim of this conference was to invite about 30 individuals to assist Stony Brook in founding a graduate CS program. Given the interest of the CS community at that time in issues like what should be taught in a CS program and at what level, it turned into an important four-day conference of about 70 invited participants from educational, industrial and governmental organizations. The conference, that included six Europeans, one Australian and eleven people from industry, was held at Stony Brook during June 1967[4]. The aims of the conference included discussing CS graduate programs in detail, specifying the content and structure of master's and doctoral programs, against a background where some academics were still questioning the very premise of such programs (in spite of more than 20,000 undergraduate CS majors and almost 5,000 graduate majors enrolled in US universities in 1966-67 according to Hamblen (1968)).

Misconceptions about CS existed even in the National Academy of Sciences and other US government bodies since they believed that there was no need for a special program in CS as there was none for electron microscopy and both were just tools. CS was not considered a coherent intellectual discipline. Also, the best computing professionals were trained in mathematics or physics or electrical engineering and therefore training of faculty and students could be carried out by these disciplines. Some believed that computing was only a fad which was going to subside (Oettinger, 1966).

In addition, the conference was to consider the controversial issue of CS organizational structure in a university given the almost total rejection of CS by some academics and total acceptance by others. Issues of the relationship between computer centers and CS academic programs including joint appointments, CS research in computer centers and the role of CS academics in computing services, were also to be discussed. It was noted that people holding conflicting and competing viewpoints were purposely invited to bring important issues into the open for debate.

---

[4] http://www.cs.sunysb.edu/welcome/OurHistory.html provides a list of participants and their group photo.

Many conference participants were concerned about the role of CS in higher education. 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. The fears were not helped by Oettinger (1967), President of the ACM at that time and Professor of Linguistics and of Applied Mathematics at Harvard, who noted that departments of CS had no place in the eternal scheme of things and that it was an intellectual mistake to have CS departments. He then added that he could see no tactical alternative to having them although he was worried about the isolation of CS in a separate department from mathematics and engineering. He also opposed the name "Computer Science" and noted that using that name was dangerously misleading to students and the world at large since most of what CS academics did was engineering. Oettinger thought that CS graduates should be well versed in physics, mathematics, engineering, economics and social sciences. In the discussion that followed Oettinger's talk it was noted that it was perhaps politically correct for a Harvard academic not to believe in CS.

The uncertainties about the intellectual substance of computing came mostly from mathematicians or pure scientists but not from engineers. Friction already existed between pure and applied mathematics. Although there was an enormous range of intellectual activity in computing, most observers were not familiar with the full scope of CS (Beckman, 1967). Beckman briefly described CS programs in 19 US universities and emphasized the importance of proper staffing and organization and the computing subjects taught. The issue of a suitable organization structure for CS was discussed and several different options presented. Beckman concluded that there was no correct organizational structure that would suit all universities. Although Zadeh (1967) from the Department of Electrical Engineering and Computer Science at Berkeley, agreed that there was no perfect organizational structure for CS, he argued that many EE 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 some people were reluctant to support establishment of a new department, others were strong proponents of a separate CS department structure. Alan Perlis from the Department of Computer Science at Carnegie-Mellon and Stanley Gill from the Computing Center at Imperial College, London believed CS had other interests outside Mathematics and EE (Perlis' paper was titled "Computer Science is Neither Mathematics nor Electrical Engineering") that would be damaged if it was placed in either Mathematics or in EE. Gill noted that CS may have been hotch-potch at that time but so was EE and he did not want EE "messing up CS". CS was not part of EE and the misconception had caused too much damage already. Gill disagreed with Zadeh's comment that CS lies close to mathematics in its attitude and remarked that it was the attitude of mathematicians that led to their dismal failure to adopt CS when they had the chance. CS had a very different attitude (Gill, 1967 and Perlis, 1967).

Perlis discussed the CS PhD program at Carnegie-Mellon University which 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, optimization 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".

There were others who had a very different view of the discipline. One of these views suggested that a discipline may be viewed as a subset of knowledge which has three important characteristics: scope, depth, and structure (Slamecka, 1967). For it to be a 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. 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 were as powerful. Slamecka recommended that the discipline be called Information Science and Engineering and described the structure of the discipline by viewing it as consisting of three theories or concerns:

1. *Theory of information* – this remained to be developed but includes 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.

Slamecka noted that theories of information and of information process are of the nature of science while the theory of information systems was engineering in its character. CS

therefore straddled and united science and engineering. Unfortunately, there was little support for Slamecka's views given that there were others who were presenting a much simpler view that CS was mostly about programming.

Four workshops discussing the master's program, the doctoral program, the position of Computing Science in the university structure, and the computing center and the academic program were held at the conference.

The master's program workshop report (Ashenhurst, 1967) noted a number of possible objectives for the master's program but put the heaviest emphasis on the technical professional degree, to some extent integrating it with a predoctoral degree. The workshop focused on a list of questions about the master's program, for example what capabilities should the graduate have, what mathematics and programming level should be prerequisite and whether a thesis or a project should be part of the program? References were made to Curriculum 68 which was under preparation at that time. A list of industry positions that could be suitable for master's graduates was presented. The workshop agreed that a professional master's needed to be more than one year but perhaps not two years.

The PhD program workshop report (Hull, 1967) discussed PhD programs at five universities (Purdue, Pennsylvania, Michigan, Wisconsin and Toronto) and discussed reports from seven subject areas with the aim of determining to what extent each area should be required for all doctoral students. Since four of the five PhD programs discussed were being run by graduate schools that had no undergraduate CS program and were admitting students from many disciplines, the issue of whether a BS degree in CS was a sufficient basis to go on to graduate studies came up. Many participants thought a BS degree in CS was *not* a good preparation, with some participants from successful graduate CS schools questioning the necessity or desirability of undergraduate programs in CS.

There was lively discussion about undergraduate CS programs after each of the master's and doctoral program workshop report presentations. Not only was there concern about admitting CS graduates to CS PhD programs, there was also some concern about admitting a CS graduate to a CS master's. Some people thought that a master's program was likely to cover material similar to that which a CS graduate would have studied at undergraduate level and therefore a university might not need to offer both. Elliot Organick from the University of Houston remarked that he was disturbed that there was no workshop on undergraduate education in the conference. In his view, throughout the conference there was a negative or apologetic or inconsiderate attitude towards undergraduate CS programs from many participants. One of the reasons for this might have been that CS academics were having difficulty figuring out how such a program would fit with CS graduate studies and in the view of some graduate studies were more important for CS to become respectable. Organick noted that universities must develop the capacity to supply adequately educated computing graduates to meet the critical need of the society. He thought that undergraduate programs would not impact graduate programs significantly since no more than 10% or 15% of undergraduates would want to go on to graduate studies. Others thought the ratio could be more like 30%, as it was in engineering.

The CS in university structure workshop report (Atchison, 1967) lists eight different solutions. The workshop unanimously agreed that CS was a separate academic discipline and a suitable administrative structure was needed to nourish the new discipline.

The workshop report on the computing center and the academic program (King, 1967) lists the functions of the computing center and discusses a number of issues about the centers and CS academic programs. It was noted that organization of computing services was likely to change significantly in the future.

6. Other Pre-1968 Papers

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 without which computer education could not even keep up with progress in the discipline. Utilizing a number of computer scientists that were scattered through other university departments would be quite ineffective in satisfying the needs of CS.

Forsythe noted that Stanford had no undergraduate program in CS and had no intention of starting one. Curiously enough, Forsythe's view was that a bachelor's degree in CS would be a terminal degree although CS could be appropriate for graduate work in another discipline. Forsythe also remarked that the 1965 Preliminary Recommendations were similar to a master's curriculum.

In a survey of the coursework done by CS master's degree students at 25 US universities, it was found that all students were taking a programming course (Fortran, Algol or MAD[5]), one or two courses in numerical analysis, and there was considerable emphasis on software design and construction, Boolean algebra, and automata theory. Artificial intelligence was present in many. There were few courses on computer applications, numerical control, hardware design and there were almost no business related courses (Elliot,1968).

[5] MAD, the Michigan Algorithmic Decoder, was designed at the Michigan Computer Center by Bernard Galler, Bruce Arden and Robert Graham. It was designed for rapid compilation for student problems (Tomkins, 1963).

Considerable discussion took place regarding directions for including computing in EE curricula and the role EE departments needed to play in providing undergraduates with CS competence. The COSINE Committee of the National Academy of Engineering was formed in 1965 (COSINE, 1967, 1968). The committee believed that EE had vital concern not only with the use but also with the conception, design and construction of digital computers. A major reorganization of the EE curricula with greater emphasis on discrete systems that could accommodate the needs of students wishing to major in computer sciences within EE was required. It was recommended that EE curricula needed to be much more flexible allowing a student almost a full year of electives from courses that were comparable to those taken by CS students. The Committee presented subject areas rather than courses that could be covered. This 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. An option within EE was recommended as the best route for offering the Computer Engineering program.

7. Curriculum 68

As noted earlier, the number of CS programs being offered in the USA was growing rapidly and experience from these programs and new developments in computing led to the 1965 Preliminary Recommendations being substantially revised. The Curriculum 68 report recognized that the debate on the justification and description of CS as a discipline had progressed. 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. Given the strong support for the name Computer Science, the Committee endorsed it.

The Curriculum 68 committee was concerned 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 and therefore it was necessary to provide direction to both. The Committee's immediate objectives were to organize 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. Doctoral programs were also discussed since computer scientists considered the establishment of doctoral programs to be very important for the discipline to achieve respectability. The report also includes recommendations about service courses, CS minors and continuing education.

The report was well-organized, lucid and complete as well as visionary given the environment in which it was prepared and to some extent it helped standardize the CS curriculum. The recommendations also encouraged publication of many textbooks since the authors could now write for the recommended courses. The Curriculum 68 committee consisted of 12 people, almost all of whom attended the Stony Brook conference. William Atchison, a member of the Committee and the first chairman of the ACM Curriculum subcommittee, has noted that the Committee worked very hard, and sought advice from a

large number of consultants (Curriculum 68 Report lists 36 consultants and 28 others who sent written comments), with the aim of maximizing the shelf-life of Curriculum 68. Since most of the curriculum designers had their training in mathematics, much time was spent on discussing just how much mathematics should be required in the curriculum (Atchison, 1981).

The report made it clear that establishing a CS program in a college or university faced formidable difficulties given the problems of finding adequate faculty, providing adequate laboratory facilities and developing new courses in the discipline.

The Committee proposed a classification of CS, somewhat similar to what Slamecka 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 embraced data structures, programming languages and models of computation.
2. *Information processing systems* – systems having the ability to transform information; interaction of software and hardware. It included computer design and organization, 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.

In addition to the above three areas, two other related areas were listed. These were "Mathematical Sciences" (12 course titles listed) and "Physical and Engineering Sciences" (9 course titles).

The 1965 Preliminary Recommendations had envisaged 16 CS courses; 11 were retained in Curriculum 68 while two of the others (Computer Organization and Programming, Algorithmic Languages and Compilers) 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 CS. Seven new courses were added, including a course on discrete mathematics and another on computer organization; 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 flexible. The Committee noted that undue

specialization was not appropriate in the undergraduate program but technical electives could be used to develop, for example, specializations in applied systems programming, computer organization and design, scientific applied programming, or data processing programming.

The Curriculum 68 committee 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. True-to-life programming projects, perhaps during summer employment, were recommended.

The Committee accepted the recommendations reported in Atchison (1966) regarding the mathematics content 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.

A major Curriculum 68 contribution was detailed course descriptions including course philosophy and associated textbooks and references. In addition, the report provides detailed course interrelationships including prerequisites.

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 were 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 courses included material in some of the courses presented in the 1963 ACM National Conference Panel. B1 was mostly about algorithms and programming using a single language and perhaps introducing a second language that was quite different (e.g. SNOBOL). B2 was about machine language and assembler programming. B3 was a basic discrete mathematics course while B4 was a basic numerical algorithms course. These were supplemented by nine intermediate courses which were designed for junior-senior level:

- I1: Data structures (3-0-3)
- I2: Programming languages (3-0-3)

- I3: Computer organization (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)

The first four intermediate courses were part of the core and consisted of mostly basic CS material. I1 dealt with data structures like lists, arrays, trees and graphs, and searching, sorting and storage structures for them. I2 covered specification of programming language syntax and semantics, storage allocation, list processing, string manipulation and simulation languages. I3 was computer organizations which brought together material from previous courses. I4 was devoted to a study of batch processing, loading, subroutine linkage, databases and their design, design of files, tables and lists, and restricted accessing methods. The next five intermediate courses were more specialized.

The eight required core courses (B1-4, I1-4) provided a fair balance 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 and 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 organization (3-0-3)
- A3: Analogue and hybrid computing (2-2-3)
- A4: System simulation (3-0-3)
- A5: Information organization 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)

The nine advanced courses dealt with a variety of topics. A1 covered the theory of context-free grammars and formal languages, syntactic analysis and the relationship between formal languages and automata. A2 dealt with computer system design including computer arithmetic, storage management, input-output, and comparison of different computer systems. A3 dealt with the concerns of engineers, some of whom were still using analogue computers. A4 dealt with simulation and modelling of discrete systems. A5 covered techniques for organizing, storing, matching and retrieving information. A6 dealt with

handling of graphical information including line drawings, block diagrams and 3-D surfaces. A7 covered abstract models of machines including Turing machines. A8 dealt with data processing and covered information storage and retrieval in the business data processing environment. A9 was to deal with computer applications that attempt to achieve goals considered to require human mental capabilities.

7.3 Curriculum 68 – Master's and PhD Programs

The Committee recommended an outstanding master's program that was to 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 of the three areas of CS and further courses in mathematics or CS containing high mathematical content.

It was noted that doctoral programs would vary from university to university depending on the interests of the faculty members 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.

8. Computing Facilities in the 1960s

The computer facilities available at each university impacted the type of computing courses that were being offered and whether the university introduced a CS program.

As noted earlier, the IBM program of heavily discounting machines to more than 50 universities in the 1950s helped many universities establish computing facilities. More importantly, during the 1950s and 1960s, National Science Foundation provided considerable grant money to universities to establish or upgrade computing facilities (Aspray and Williams, 1994). NSF started getting grant proposals with computing requirements in 1953 and as a result an Ad Hoc Advisory Panel on University Computing Facilities under the chairmanship of John von Neumann was established in 1955. In the beginning, the Foundation redirected some funds to computing but demand for computing facilities was growing and in 1958 NSF received 19 facilities requests, most of them for computing. From 1959, the Foundation decided to allocate a separate budget for large computer equipment purchases. NSF support for computer facilities rose from $2.5 million in 1961 to $11.3 million in 1967 and then declined over the next five years.

As a result of growing computing needs and rapidly changing technology (e.g. time-sharing systems, IBM System 360) a number of investigations into computing needs were conducted. Two important reports, the 1966 Rosser report "Digital Computer Needs in Universities and Colleges" (Rosser, 1966) and the 1967 Pierce report "Computers in Higher

Education" (Pierce, 1967), provided much information, made many recommendations and had considerable impact on funding for computers in higher education. The Rosser committee found that computing facilities at universities were growing quickly. There were 40 university computing centers in 1957, 400 in 1964. It recommended that the academic computing community be doubled over the years 1964 to 1968. The Pierce committee recommended that the Government cooperate with universities to improve computing facilities and called for extensive training of faculty to meet the demand for computing courses. Another survey of expenditures, sources of funds and utilization of computers in higher education was carried out in 1966 by Hamblen (1968) and a list of computers installed and on order is included in this paper. Commenting on some of the information collected during 1966-67, Hamblen (1971) notes that the majority of 32 bachelor's degree programs surveyed by him were lacking adequate hardware. More computers and links to larger computers were needed.

9. Conclusions

This paper has described the debates that were going on in the 1960s about the nature of CS, whether it constituted 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 centers ought to play in teaching.

The debates on whether computing should be in engineering or mathematics were inconclusive since neither engineering nor mathematics were able to reach consensus on what should be taught in computing. Thus there was slow progress in effective curriculum changes in both discipline areas to accommodate computing. As an example, Rice and Rosen (2004) describe friction between CS advocates and mathematics in the early 1960s at Purdue. There were debates in the CS community as well. Korfhage (1969) reports an amusing discussion in which a CS academic was suggesting that the CS curriculum should be less theoretical, more practical, while a voice from industry was calling for more theory. Finally, voices of business-oriented computing advocates were lost in these debates. Some early business computing programs were started in junior colleges while Data Processing Management Association (DPMA) was promoting certificate programs that required three years experience in the field (Tomkins, 1963).

Major developments in CS curriculum in the USA during the 1960s culminated in the ACM proposals presented as Curriculum 68. Although Curriculum 68 did not stop the debates about where a CS department should be located in a university or when a university should start an undergraduate CS program, the landmark contribution was very influential in showing that CS was indeed a discipline. It provided direction to academics who were struggling to support or introduce CS programs within their universities and therefore spawned many undergraduate and graduate degree programs. Curriculum 68 stood the test of time in a very dynamic field. The basic and intermediate courses it recommended are still

the basis of many undergraduate programs. The Curriculum 68 committee showed tremendous foresight and perhaps no curriculum effort since then has had the same impact on CS.

Acknowledgements

I wish to thank Karl Reed for carefully reading a draft and making a number of valuable suggestions. I also thank the referees for their comments which have assisted in improving the paper and Dr David Grier, Editor in Chief of the Annals, for encouragement.

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.

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

Aspray, W. and B. O. Williams (1994), Arming American Scientists: NSF and the Provision of Scientific Computing Facilities for Universities, 1950-1973, *IEEE Annals of the History of Computing*, Vol 16, No 4, pp 60-74.

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.

Atchison, W. F. (1967), The Position of Computing Science in the University Structure – A Report of the Workshop, in Finerman (1967c), pp 169-76.

Atchison, W. F. (1981), Computer education, past, present, and future, ACM SIGCSE Newsletter, Vol 13, No 4, pp 2-6.

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

Ceruzzi, P. (1988), Electronics Technology and Computer Science, 1940-1975: A Coevolution, *IEEE Annals of the History of Computing*, Vol 10, No 4, pp 257-75.

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

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 763-5.

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 Edu.*, Vol 52, No. 3, pp 177-88.

Forsythe, G. E. (1963), Educational Implications of the Computer Revolution, in *Applications of Digital Computers*, V. F. Freiberger and W. Prager (eds.), pp 166-178.

Forsythe, G. E. (1964), An Undergraduate Curriculum in Numerical Analysis, *Comm. ACM*, Vol 7, No 4, pp 214-15.

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

Gill, S. (1967), Planning a Profession, in Finerman (1967c), pp 117-22.

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

Gorn, S. (1964), Mechanical Languages: A Course Specification, *Comm. ACM*, Vol 7, No 4, pp 219-22.

Hamblen, J. W. (1968), Education: Expenditures, sources of funds, and utilization of digital computers for research and instruction in higher education: 1964-65 with projections for 1968-69, *Comm. ACM*, Vol 11, No 4, pp 257-61.

Hamblen, J. W. (1971), Using computers in higher education: past recommendations, status, and needs, *Comm. ACM*, Vol 14, No 11, pp 709-12.

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

Katz, D. L. and Organick, E. I. (1960-61), Use of Computers in Engineering Undergraduate Teaching, J. Engg. Educ., Vol 51, pp 183-205.

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

King, K. (1967), The Computing Center and the Academic Program – A Report of the Workshop, in Finerman (1967c), pp 177-92.

Korfhage, R. R. (1964), Logic for the Computer Sciences, *Comm. ACM*, Vol 7, No 4, pp 216-18.

Korfhage, R. R. (1969), CSE: Theory or Practice, SIGCSE Bulletin, Vol 1, No 4, pp 6-7.

Knuth, D. E. (1972), George Forsythe and the Development of Computer Science, *Comm. ACM*, Vol 15, No 8, pp 721-7.

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. (1966), President's Letter to the ACM Membership. *Comm. ACM*, Vol 9, No 12, pp 838-9.

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

Perlis, A. J. (1964), Programming of Digital Computers, *Comm. ACM*, Vol 7, No 4, pp 210-11.

Perlis, A. J. (1967), Computer Science in Neither Mathematics nor Electrical Engineering, in Finerman (1967c), pp 69-80.

Pierce, J. R. (1967), Chairman, Panel on Computers in Higher Education, President's Science Advisory Committee, "Computers in Higher Education", Government Printing Office, Washington, DC.

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

Report (1960b), Report on a Conference of University Computing Center Directors, *Comm. ACM*, Vol 3, No 10, pp 519-21.

Rice, J. and Rosen, S. (2004). Computer Sciences at Purdue University – 1962 to 2000. IEEE Annals of the History of Computing, Vol 24, No 2, pp 48-61.

Rosser, J. B. (1966), Chairman, Committee on Uses of Computers, National Academy of Sciences-National Research Council, "Digital Computer Needs in Universities and Colleges", NAS-NRC, Publication 1233, Washington, DC.

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 135-68.

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.