CS26

STANFORD UNIVERSITY'S PROGRAM IN COMPUTER SCIENCE

BY

GEORGE E. FORSYTHE

TECHNICAL REPORT CS26

JUNE 25, 1965

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

STANFORD UNIVERSITY'S PROGRAM IN COMPUTER SCIENCE

by

George E. Forsythe

1. What is computer science?

One can obtain delivery in 1966 of automatic digital computers capable of making decisions in less than 150 nanoseconds and multiplying in less than one microsecond. Such machines can retrieve each of a quarter million words in less than 100 nanoseconds, and each of some 3×10^9 characters in something like 0.01 seconds. Since a 500-page book holds approximately 1.5×10^6 characters, the above large store holds the equivalent of 2000 monographs - a substantial little library,

Until the first automatic digital computers were available for delivery approximately 15 years ago, we were limited to making decisions, multiplying, and accessing a fast store in something like 10 seconds, while access to each character of a 2000-book library could hardly take less than 100 seconds, Thus the past 15 years has seer, accelerations of 10^4 (access to slow store), 10^7 (multiplications), and even 10^8 (access to fast store),

In comparison, the speed-up in travel between walking and going by Jet is about 10^2, that in the dissemination of information between manuscript and a large newspaper machine is about 10^6, and that in communication speed between usfng sound and radio waves is also about 10^6. These changes in rates of transportation and communication completely remade the world of earlier times, It is evident that the even greater speed-ups in information processing are remaking our world, It is furthermore clear that it will be many years before our capacity to exploit the new power till catch up with its present capabilities,

I consider computer science to be the art and science of exploiting automatic digital computers, and of creating the technology necessary to understand their use. It deals with such related problems as the design of better machines usfng known components:, the design and implementation of adequate software systems for communication between man and machine, and the design and analysis of methods of representing information by abstract symbols and of processes for manipulating these symbols,, Computer science must also

concern itself with such theoretical subjects supporting this technology as information theory, the logic of the finitely constructable, numerical mathematical analysis, and the psychology of problem solving. Naturally these theoretical subjects are shared by computer science with such disciplines as philosophy, mathematics, and psychology.

Because the representation and processing of information are the core of computer science, many persons refer to our subject as the science of information processing, and indeed the International Federation for Information Processing bears such a name. Until a rather substantial discipline is built up, I prefer to keep explicit our pragmatic goal by retaining the word "computer" in our name, Perhaps the name "computer and information sciences" is a proper compromise at present,

An interesting comparison of computer science with both pure and applied mathematics has been made by Gorn [1]. While Gorn is thinking in [1] only of the scientific applications of computers, the distinctions presented certainly could be transferred to all areas of computing. To Gorn, the pure mathematician is interested in the syntactical relations among symbols, quite apart from their meaning in the physical world or their computability, Thus the all-important questions of pure mathematics deal with the structures of theories, not with their meaning. For example, the majority of practicing pure mathematicians are unconcerned with what numbers actually are, and indeed the question is unsettled; they are instead concerned with what relations exist among numbers and among objects built up from numbers.

The applied mathematician is primarily concerned with the semantics of symbols – what do mathematical theorems mean as applied to the physical world? For example, how can mathematical analysis help us understand physics or economics or electronics? Finally, the computer scientist is concerned with the pragmatics of the applications of mathematics, What algorithms can actually be used to calculate things of a mathematical nature% What does it cost in storage or time or human effort to perform these algorithms'? What guaranteed or probabilistic error bounds can be constructed for the answers? What languages can they be expressed in? What hardware do they require?

An article by Keenan[2] is part of a journal issue devoted mainly to questions of curriculum in computer science. Keenan develops the definition of computer science in several broader areas with which we generally associate the name,

2. The objectives of computer science education.

There are at least three different groups of students to whom' computer science education should be directed, and the objectives are different for each.

a. Nontechnical students

The advent of computers means that almost all citizens of the developed nations will be greatly affected by computers, Hence a general education should include enough background for the citizen to comprehend something about computer science. Since a university educates the future leaders of the community, the students need a background for making decisions in a computerized world, They need to learn what computers are, what they can do, what they cannot do, They need to realize the large role of human beings in creating computer systems — that every bit of automation is achieved by human planning in the large and in detail,

b. Specialists in other technical fields

The automatic computer is one of the most important tools to have been devised in the history of man, and its domain of application is very wide. It is already recognized by most engineers and natural scientists that they must know computing fairly well, Current studies of engineering education emphasize education in general purpose tools with long expected lifetimes like mathematics, English, statistics — at the expense of special information better relegated to handbooks, Automatic computing is indeed becoming recognized as one of the lasting tools.

Zadeh [3], chairman of the Department of Electrical Engineering, University of California, Berkeley, makes a particularly strong claim about the importance of computer science in technical education: he states that electrical engineering will suffer a serious decline in importance as a discipline unless it can absorb a substantial amount of activity in computer science,

It is now clear that students of social science must also acquire a familiarity with computing methods,, And the serious student of humanities will soon find computers indispensable, if he is to carry out research on any substantial volume of data,,

c. In education of computer science specialists

The most important group to be educated in computer science are the future specialists in the field, for they are the seed who will become the creators and the teachers of the future. They must receive enough background to be able to follow and preferably lead the future development of the subject,,

Keenan [2] distinguishes three groups of computer science majors. In my opinion it is probably better to lump the undergraduate major with the group in b. above, for without more than a bachelor's degree the graduate is likely either to become a mere assistant to a computer user or an applications specialist himself.

For the graduate student of computer science the faculty must create a discipline and inculcate standards of performance in it, The student must learn to read and to write the appropriate literature, A background must be built for years to come. Computer science education, like all education, must aim to light a lamp for the student, rather than try to fill the student's bucket with knowledge, This education must create the field's leaders, full of ambition and fire to attack the all-pervasive unsolved problems of computer science,

3. How can a U. S. university realize the above objectives?

The first major step by which a U. S. university can realize the objectives stated above is to create a department of computer science, or the same thing under a different name, The reason for forming a department is to enable computer scientists to acquire faculty of their own choosing, and exercise control over the curriculum of students wishing to specialize in computing. There is abundant experience to show that without such an administrative step computer education will not even keep up with the field.

Without a department, a university may well acquire a number of computer scientists, but they will be scattered and relatively ineffective educationally, Naturally a department cannot be started until there are two or three computer scientists on the faculty. One may expect that the department-to-be will be in a probationary status for two or three years,

A crude survey by the author shows that computer science departments are in existence or imminent in approximately two-thirds of the 15 largest universities of the U. S.

Probably the department of computer science belongs in the school of arts and sciences, because of its close ties with departments of mathematics, philosophy, and psychology. But its relations with engineering departments concerned with systems analysis and computer hardware should be close,

In years to come we may expect departments of computer science to come together with departments of systems analysis, applied physics, operations research, applied mathematics, and so on, inside schools of applied science or even schools of mathematical sciences, We can even hope for a weakening of the power of individual departments, and a concomitant strengthening of the ability of a university to carry out interdisciplinary programs.

A department of computer science has the responsibility of creating curricula for each of the groups mentioned in paragraph 4 above. Thus there will be degree programs for its own majors, There will be service courses for majors in other departments in which computing is recognized to be central, There will be "computer appreciation" courses as part of a general education program., In a university with an adult education program some or all of these will be offered at night for the "re-treading" of graduates of past years,

In all these courses the computer science department must strive hard to extract and teach the <u>essence</u> of computer science, and avoid its transient aspects, For example, introductions to computing should avoid spending large amounts of effort on comparing base 2 and base 10 arithmetic, And the detailed study of a machine language seems better relegated to specialist computer science courses than taught in a first course. Whereas at one time

performing arithmetic on machine-language commands seemed to be, the essence of programming, it is no longer so important, In fact, where multiprocessors share the same routines one cannot permit the basic routines to be modified at all, Finally, the department should insist that its majors devote substantial time to studies in other departments (mathematics, logic, psychology, electronics, etc.). For having control of the computer science curriculum does not imply actually teaching the entire curriculum1

In addition to a computer science department, a university needs a strong computation center in order to attain its educational goals in computer science. Such a center must be organized with machines, languages, and software capable of receiving a large number of programs from different classes of students. In a batch-processing computation center we expect that each student of introductory programming should have one pass per calendar day on a computer, More advanced students will come less often, but with longer runs. Thus the center must be well endowed to receive a very large volume of student programs in addition to its better rewarded goal of serving faculty research work. Unless you have had experience with this job-shop type of computing, you cannot recognize how poorly adapted typical machine systems are to dealing with it, Even with good machine systems it is a large problem in logistics and human relations to run more than 1000 jobs a day for a variety of users,

4. Stanford University's program in computer science,

Since January 1965 Stanford University has had a Computer Science Department within its School of Humanities and Sciences. The new department had been gestating for approximately three years within the Mathematics Department, In September 1965 the Computer Science Department will have positions for ten faculty members at levels of assistant professor and above. The principal fields of interest of the faculty are approximately as follows: numerical analysis (4), artificial intelligence (2), programming languages and systems (1), machine organization (1), computer control of physical data (1), logic and linguistics (1). We need more faculty in programming languages:, as this is the field of largest enrollment of our department,

We have no undergraduate degree program in computer science, and no present intention of starting one. We have approximately 70 graduate majors, of whom approximately 45 are full time students, We offer a Master of Science in Computer Science, and by June 1965 we will have awarded 25 such degrees since 1961, We offer a Ph.D. in Computer Science, None has yet been awarded, but four students have.just passed our written qualifying examinations for the degree, Over the past seven years the author has been the principal thesis advisor of some seven students who have written dissertations in numerical analysis for their Ph.D. in mathematics,

In the appendix are given our catalog descriptions of the courses offered in our department, together with an indication of the requirements for the Ph.D. degree, (A one-quarter 3-unit course normally consists of 30 lectures.) We have also prepared an informal syllabus for our qualifying examination, which includes three papers::

(1) numerical analysis and computational mathematics;

(2) computer and programming systems;

(3) advanced nonnumeric applications, artificial intelligence, and mathematical theory of computation,

Our main service courses are directed to students who have already had calculus, Our one-quarter introduction to computing (course 136) ranks as approximately the fifth most popular non-required course at Stanford, We have two quarters of introductory numerical analysis (courses 137, 138) presupposing course 136, linear algebra,, and ordinary differential equations. We have a special form of course 136 (courses 5,6) directed to younger students of engineering,

We have instituted one course for general students – course 126, presupposing only high-school mathematics and directed to students of humanities and the social sciences,,

Courses 5, 6, 126, 136 all involve very substantial amounts of computer use by the students as individuals, They are effectively laboratory courses.

As time goes on, we anticipate the creation of more sections of course 136, directed to majors in special areas and presupposing different backgrounds and interests,

Several other departments at Stanford teach computing in one form or another, Most of these are applications courses presupposing our course 136,

Our graduate offerings include a Z-quarter sequence on programming systems; a l-quarter course on the logical structure of computers; a 3-quarter sequence on numerical analysis; a z-quarter sequence on elementary artificial intelligence; a 3-quarter sequence covering list-processing, logic of computing, and advanced artificial intelligence; a special numerical analysis course; a l-quarter course on control programming; and a programming laboratory course. We expect to add a l-quarter course next spring on either mathematical linguistics or the logic of theorem-proving.

As we teach computing to specialists of other fields,, it is essential that their own faculty be able to follow up with important applications. In the recent transition period, we found it desirable to teach the faculty directly, We have had three special one-week sessions, two for engineering faculty and one for biomedical faculty, These courses take very careful planning, but pay off very well in knowledge and good will,

The Stanford Computation Center is well organized to receive large numbers of student jobs, mainly in Extended Algol for the Burroughs B5500. However, as the extra costs required to handle student jobs mount into the several tens of thousands of dollars, it will be essential to receive special funding for this work.

The Computer Science Department is not responsible for the Computation Center's service work, but it does provide leadership for system selection and organization, However, the Computer Science Department will itself soon have some large computer systems acquired in connection with research contracts.

5. Miscellaneous remarks.

It would be desirable to have criteria of Stanford's progress towards our goals in computer science education. So far we are too much involved in just getting organized, recruiting, teaching, and so on, to have done much evaluation. We do know that Stanford students appreciate our program very much. Courses 5,126, and 136 have a campus-wide reputation for being time-comsuming, because it takes beginners many hours to get problems correctly keypunched and run. Nevertheless, the enrollments are high. With both formal and informal programming courses, we estimate that we reach about 1000 persons a year. If these people average but two years at Stanford, then about 2000 persons know computing in an active way. With some 10,000 students and 1000 faculty and teaching staff, that means that almost 20 per cent of the Stanford academic family are well acquainted with computing.

We therefore feel that computing is established at Stanford, and that it is time to turn our attention towards raising the quality of our work, We need not only to teach students to program — we need to teach them to program well. We must give many more of the science and engineering students who use the computer an introduction to good algorithms and a fear of bad algorithms.

We need to improve the quality of our graduate instruction. Many of our graduate students are students who were refused admission to Mathematics and take us as a second choice. We don't resent being second choice, but we want first-class students.

We have wondered about the place of numerical analysis in a university with a department of mathematics and another in computer science. No doubt there will be two kinds of numerical analysis Ph.D. degrees. One will be a mathematics degree, following the regular required mathematics courses, with some elective courses in computer science and a thesis in numerical analysis directed by some one in the computer science faculty. This would be appropriate, for example, for a student whose thesis developed asymptotic error bounds for finite difference methods for solving partial differential equations. There might or might not be experimental computations, The other will be a computer science degree, with a number of extra mathematics courses and a thesis in numerical analysis. This thesis would emphasize the algorithmic

aspects of numerical analysis, and would certainly involve experimental computations on a digital computer, It might, for example, explore the kinds of languages appropriate to solving difficult mathematical problems in computer system involving intricate man-machine interactions,

However, in our first year of independent existence, we are seeing some sign that graduate students interested in numerical analysis may find themselves lost between chairs, The principal obstacle to a Ph.D. at Stanford in either mathematics or computer science is the qualifying examinations. A graduate student with rather special interests in numerical analysis is unhappy to find that the qualifying examination in mathematics requires the mastery of mathematical analysis which goes far beyond what is used in numerical analysis, He finds that the examinations in computer science involve too mueh mastery of such subjects as heuristical programming and computer organization for his taste and ability, If a student is so strong that he can conquer one or the other of these examinations, it is surely worth while for him to do so. But I am developing some concern for the numerical analysis student who is not quite so strong or well motivated, Has our creation of a new Ph.D. degree in computer science actually worsened the situation for the would-be numerical analyst? Will it therefore be necessary to create a special inter-departmental program in numerical analysis? How can we hold the line against such a proliferation of programs?

References

[1] Saul Gorn, THE COMPUTER AND INFORMATION SCIENCES: A NEW BASIC DISCIPLINE, SIAM Review, vol. 5(April 1963), pp. 150-155.

[2] Thomas A. Keenan, COMPUTERS AND EDUCATION, Comm. Assoc. Comput. Machinery, vol. 7(April 1964), pp. 205-209.

[3] L. A, Zadeh, ELECTRICAL ENGINEERING AT THE CROSSROADS, paper presented to the Institute for Electrical and Electronic Engineering, March 1965, Proceedings, pp. 47-50.

Computer Science Department
Stanford University
Stanford, California
May 12, 1965

APPENDIX TO CS26
Issued June 25, 1965

STANFORD UNIVERSITY'S PROGRAM IN COMPUTER SCIENCE

BY

GEORGE E. FORSYTHE

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

REVISED
August 1965

COURSES AND DEGREES 1965-66

COMPUTER SCIENCE

OFFERINGS AND FACILITIES

The Department aims to acquaint a variety of students with the technological and intellectual roles of automatic digital computers, and to educate research workers in computer science. In spite of the diversity of the applications, the methods of attacking problems with computers show a considerable unity, and computer science is concerned with the underlying principles. The field is still young, and the student will find many more questions than answers.

Of the numerous areas of computer science, the Department has competence in numerical analysis, artificial intelligence, programming systems and languages, logical design of computer systems, and computer control of external devices.

Courses in data processing are offered by the Industrial Engineering Department and in the Graduate School of Business.

Since computer science is inherently interdisciplinary, graduate students of computer science are expected to include in their study program a good deal of work in other departments; see the list of suggested courses below.

There is no Bachelor's degree in computer science. Undergraduates who wish to enter the field are advised to major in mathematics and include Computer Science 136, 137, 138, and 139 in their course of study,

In connection with its courses and research, the Department makes considerable use of the Computation Center. See the section "Computation Center" in this Bulletin.

PROGRAMS OF STUDY

Master of Science

The University's basic requirements for the Master's degree are discussed in the section "Degrees" in this Bulletin. The following are departmental requirements:

A candidate is expected to complete an approved course program of 45 units; at least 36 units will be in this department, or in the Mathematics Department, or selected from the list of suggested courses in other departments which appears at the end of the course offerings in Computer Science. These 36 units must include 6 units of Computer Science 239 and 15 additional units of, courses numbered 200 or above.

A student whose primary interest is in the numerical aspects of computing should include in his program Mathematics 106, 113, 114, 115, 130, 131, and

Computer Science 136, 137, 138, 237a, b, unless as an undergraduate he has taken these courses or equivalent ones elsewhere.

A student whose primary interest is in the nonnumeric aspects of computing should include in his program Mathematics 113, 130, Philosophy 160a, b, and, Computer Science 136, 137, 139, 231, 236a, b, 238, unless as an undergraduate he has taken these courses or equivalent ones elsewhere.

The candidate must have a 2.50 average in his course work and a 3.00 average in his courses taken in the Computer Science Department,

Doctor of Philosophy

The University's basic requirements for the doctorate (residence, dissertation, examination, etc.) are discussed in the section "Degrees" in this Bulletin. The following are Departmental requirements:

Candidates for the degree of Doctor of Philosophy Will follow such courses as are approved by the Department faculty, subject to general University regulations. Each student's program should be arranged to include work in computer science, mathematics, mathematical logic, and possibly such other subjects as statistics or electrical engineering,, the proportions depending on the student's previous education and his planned research. Since computer science is becoming 'increasingly formal and abstract, we place considerable emphasis on the student's mathematical education and ability.

In any case there are the following requirements:

1) Completion as a graduate student of an approved coherent program of at least 60 units, including Computer Science courses 225, 231, 236a, b, 237a, b, 238, 239 (6 units), 382 (2 units of presenting papers), and either 224 or 245. An especially well written paper for course'239 is required.

2) A substantial reading knowledge of one of the languages: French, German, or Russian.

3) Passing qualifying examination before admission to candidacy.

The most important requirement for the Ph.D. degree is the dissertation. The Department is now prepared to supervise dissertations in the mathematical theory of computation, numerical analysis, programming languages, artificial intelligence, computer control of external devices, and in certain applications of computers.

TEACHING AND RESEARCH ASSISTANTSHIPS

There are graduate assistantships available in both the Computer Science Department and the Computation Center. Assistants will receive a tuition scholarship up to nine units of study per quarter during the academic year, and in addition will receive stipends for the nine-month academic year ranging approximately from $2300 to $2600. They will have desks in Polya Hall at the Computation Center. Some may work full time in the summer for $500 to $550 per month.

Duties in the academic year involve less than 20 hours of work per week. Part of this is in assisting Stanford people with their programs and methods for solving problems with computers, often in connection with formal or informal programming courses. Part of the time is spent in developing programs and systems for solving problems of general interest on computers, or in assisting senior staff members with research in computer science. Approximately two hours of the work week are spent in attendance at Computer Science Department colloquia and seminars.

Applicants for assistantships are expected to have a background in computing at least as deep as that achieved in course 136, together with some knowledge of a machine language. A deeper background is preferable. An applicant's major field may be computer science, mathematics, statistics, physics, psychology, electrical engineering, or other discipline in which there is significant research involving the use of automatic digital computers. Because of the great need for improved computing and programming systems as tools for research, preference will generally be given to students of computer science.

Further information may be obtained from the Executive Head of the Computer Science Department. Applications for assistantships should be made to the Financial Aids Office, together with an application for admission to graduate study in some department. Unless the applicant is also applying for admission to the Computer Science Department, he should at the same time write to the Executive Head of the Computer Science Department of his desires to have an assistantship in computing and stating his desired major department.

COURSES FOR UNDERGRADUATE STUDENTS

5. Introduction to Programming - This course is an introduction to ALGOL, a problem-oriented language for describing computational processes. There will, be practice in solving elementary problems on Stanford's automatic digital computers. The course is limited to freshman and sophomore students. Prerequisites: Mathematics B, or equivalent.

2 units, autumn, (————), WF 11
winter, (————), TTh 1:15
spring, (————), WF 11

6. Introduction to programming--Continuation of 5. - Courses 5 and 6 together include approximately the same material as course 136, with emphasis on scientific applications. This course is limited to undergraduate students.

2 units, winter, (————), TTh 1:15

COURSES FOR UNDERGRADUATE AND GRADUATE STUDENTS

126. Computing for Nonscientists - This course is directed to students of social science and the humanities, and is not open to students who have the prerequisites for course 136. The syllabus is roughly that of course 136, but the problems are selected more from nonnumeric applications. Prerequisites: Mathematics B, or equivalent.

3 units, autumn, (————), MWF 2:15

136. Introduction to Algorithmic Processes - Concept and properties of an algorithm; language and notation for describing algorithms; analysis of computational problems and development of algorithms for their solution; use of a specific procedure-oriented language to solve simple numericai'and nonnumerical problems using an automatic digital computer. Prerequisite: Mathematics 23 or 43.

3 units, autumn, (————), MWF 11; (————), MWF 1:15; (————) TTh 9:30-10:45
winter, (————), MWF 10; (————), MWF 1:15
spring, (————), MWF 11; (————), MWF 1:15

137. Numerical Analysis - This course and 138 are designed to acquaint seniors and graduate students of science and engineering with methods of solving mathematical problems on automatic digital computers. Problems discussed include numerical differentiation and integration, solution of linear and nonlinear equations, solution of differential equations, and approximation of functions. Introduction to the analysis of convergence and errors. Pitfalls in automatic computation and their remedies. Prerequisites: 136 and Mathematics 130, or equivalents.

3 units, winter, (————), MWF 11; (————), MWF 2:15

138. Numerical Analysis--Continuation of 137. - Also the numerical analysis of functions of several variables, including problems of linear algebra. Prerequisites: 137 and Mathematics 113, or equivalents.

3 units, spring, (————), MWF 2:15

139. Computers and Machine-Language Programming - Introduction to machine-code programming. Representation of numeric and nonnumeric data. Machine arithmetic. Discussion of various ways of organizing machines. Prerequisite: 136 or concurrent registration in 136.

3 units, autumn, (————), TTh 9:30-10:45
winter, (————), MWF 1:15

COURSES INTENDED PRIMARILY FOR GRADUATE STUDENTS

224. Computer Simulation of Cognitive Processes - Introduction to computer simulation techniques and information-processing models of thought processes. Survey of various computer simulation models. This research area lies at an interface between psychology and computer science, and the course is expressly designed for graduate students in both fields. Some knowledge of experimental and theoretical psychology is advisable but not mandatory. Prerequisite: 136 or equivalent.

3 units, autumn, (————), by arrangement

225. Artificial Intelligence - Introduction to problem solving and heuristic programming. Survey of chess- and checker-playing programs; theorem-proving programs; General Problem Solver; mathematical, linguistic, and industrial applications. Question-answering programs, and natural-language communication with machines. Advice-taker and Inquiring System concepts. Other topics as time allows.' The course is designed to dovetail with 224 with minimum overlap, but 224 is not a prerequisite. Prerequisite: 136 or equivalent.

3 units, winter, (————), by arrangement

231. Structure of Digital Computers - Boolean algebra; analysis and synthesis of combinatorial and sequential networks; electronic components used in logical gates. The design of a simple digital processor, arithmetic unit, program control, memories. Use of this processor and its simulation on another computer. Various existing forms of machine organization. Prerequisite: 139 or equivalent.

3 units, winter, (————), MWF lo:oo

233. Topics in Numerical Analysis - Selected topics in numerical analysis. Prerequisite: 138 or equivalent.

3 units, winter, (————), TTh 9:30-10:45

236a, b. Systems Programming and the Theory of Formal Languages - The technique of constructing systems programs: supervisory programs (monitors), input-output systems, interpreters and compilers for procedure-oriented languages, in particular ALGOL. Selected topics from the theory of formal languages: syntactic analysis and semantic interpretation. Prerequisite: 139 or equivalent.

236a. 3 units, winter, (————), TTh 9:30-10:45
236b. 3 units, spring, (————), TTh 9:30-10:45

237a, b, c. Advanced Numerical Analysis - Selected topics from the theory and practice of using automatic digital computers for approximating arithmetic operations, approximating functions, solving systems of linear and nonlinear equations, computing eigenvalues, and solving ordinary and partial differential equations. Testing of methods on a digital computer. Automation of methods. Prerequisites: 138 and Mathematics 114 and 115 or equivalents.

237a. 3 units, autumn, (————), MWF 3:15
237b. 3 units, winter, (————), MWF 3:15
237c. 3 units, spring, (————), MWF 3:15

238. Computing with Symbolic Expressions - The LISP programming language with applications to symbolic differentiation, integration, simplification of algebraic expressions and compiling. Design of list-processing systems. Prerequisite: 136 or substantial programming experience.

3 units, autumn, (————), TTh 11:00-12:15

239. Computer Laboratory - A substantial computational program is undertaken and well documented. Prerequisite: 138 or 139 or equivalent.

Any quarter, (Staff), by arrangement

243. Mathematical Theory of Computation - Semantics and syntax of programming languages; formal systems for proving equivalence of programs; computability and unsolvability; computer proof procedures; related topics in mathematical logic. Prerequisite: 238.

3 units, winter, (————), TTh 11:00-12:15

245. Advanced Topics in Artificial Intelligence - Analysis and discussion of selected frontier research problems in the field, e.g., Advice-taker, game-playing programs, pattern recognition, man-machine interaction, proof procedures. Term paper focusing on research problems will be required. Prerequisites: 225 and 238.

3 units, spring, (————), TTh 11:00-12:15

246. Data Reduction and Control Programming - Organization and programming of automatic data reduction systems: data collection, storage, and retrieval;

248. Computational Linguistics - Applications of computers to language problems; Formal models of language. Parsing algorithms; recognition programs for transformational grammars; mechanical translation. Prerequisite: 136 or consent of the instructor.

3 units, spring, (_____), by arrangement

machine-to-machine data transmission; control programs; interrupt processing; list-processing applications; decision processes.. Prerequisites: 137, 231, 236a, 238.

3 units, spring, (_____), MWF 9

360. Advanced Reading and Research.

Any quarter, (Staff), by arrangement

382. Computer Science Seminar - There are ordinarily two or more sections on different topics.

1 or 2 units, any quarter, (Staff), by arrangement

The following courses offered in other departments may be of especial interest to students of computer science:

Analog Computation - See Electrical Engineering 268.
Data Processing - See Industrial Engineering 156, 210, 257, 261, and 263.
Data Processing in Business Problems - See Business 367 and 368.
Mathematical Logic - See Philosophy 160a, b, 161, and 292a, b, c.
Mathematical Models in Behavioral Sciences - See Behavioral Sciences courses.
Mathematics, - See Mathematics courses.
Organizational Processes and Task Performance - See Psychology 221.
Probability and Statistics - See Statistics 116, 219, 220.
Recursion Theory - See Philosophy 293a, b, c.
Science in Management and Operations Research - See Business 366, and Industrial Engineering 152, 252, 253, and 257.
Statistical Inference in Economics - See Economics 272.
Theory of Automata - See Philosophy 162.
Theory and Design of Systems and Adaptive Systems - See Electrical Engineering 248, 249, 250a, b, 251a, b, and 286
Theory of Switching and Digital Computer Circuitry - See Electrical Engineering 261, 262, and 266.

STANFORD UNIVERSITY

COMPUTER SCIENCE DEPARTMENT

March 30, 1965

Syllabus for Ph.D. Examination in Artificial Intelligence Research, Non-numeric Applications, and Mathematical Theory of Computation

The following is a list of topics with which the student should be familiar:

1 List Processing, Symbol -manipulating languages for non -numeric applications.

1.1 Thorough familiarity with LISP or some other list processing language.

1.2 Some familiarity with the other languages of this type; similarities, differences, and issues in the construction of list processing languages.

NOTE: The languages under discussion here are LISP, IPLV, SLIP, COMIT, SNOBOL, plus others you may discover in your reading.

2 Artificial Intelligence

2.1 Heuristic Programming heavy emphasis.

Chess and Checker Playing Programs (various; Samuel)

SAINT (Slagle)

Logic Theorist (NSS)

Geometry Theorist (Gelernter)

Assembly Line Balancing (Tonge)

. General Problem Solver (NSS)

Geometric Analogies (Evans)

Graph Isomorphism Finder

Finding Group Transformation (Amorel)

Theorem Proving a la J . A, Robinson, H . Wong, etc.
(is any of this,. or is this not, heuristic programming?)

STUDENT (Bobrow)

Advice Taker Concepts (McCarthy)

Music Composition (Hiller and Issacson)

MH-1 Hand (Ernst)

2.2 Simulation of Human Cognition

General Methodological Considerations Regarding Information Processing Models in Psychology

EPAM (Feigenbaum)

Binary Choice Hypothesis Former (Feldman)

"Concept" Formation (Hunt)

Investment Decisions (Clarkson)

Belief Structures, Personality, Neurotic Behavior (Abelson, Loehlin, Colby)

2.3 Learning in Problem Solvers

2.4 Learning in Random and Structured Nets, and The General State of "neural" models.

2.5 Pattern Recognition Theory and Applications

2.6 Other Adaptive Systems

3 Other Non -numeric Applications .

3.1 Question-Answering Programs (BASEBALL, SIR, Black's Program)

3.2 Information Retrieval Research, more generally construed . . . broad outlines of the current state of art.

3.3 Automatic Indexing and Abstracting

3.4 Machine Translation ... broad outlines of current state of art.

4. Mathematical Theory of Computation.

4.1 Formalisms for Describing Computable Functions - Turing Machines, Post Canonical Systems, General Recursive Functions.

4.2 Universality and Undecidability . .

4.3 Functions Compatable in Terms of Base Functions

4.4 Formal Properties of Conditional Expressions - Proofs of Equivalence of Computations.

4.5 Recursion. Induction

4.6 Spaces Representable in Terms of Base Spaces

4.7 Abstract Syntax - Conditions for Correctness of Compilers

5. Mathematical Logic

5.1 Propositional Calculus - Canonical Forms Rules of Inference, Decision Procedures.

5.2 Predicate Calculus - Axiomatization of Theories in Predicate Calculus; Axioms and Rules of Inference, Models, Completeness, Proof Procedures,

5.3 Set Theory - Zermelo-Frankel or Von Neuman Axioms.

References

For Section 1 see the IPLV Manual, the LISP 1.5 Manual, the SLIP Manual (Comm. of ACM, Sept. 1963), COMIT Manual, SNOBOL article (JACM, in 1964). Also, Bobrow's and Raphael's survey article (Comm. of ACM, in 1964).

For Section 2.1 and 2.3 the prime reference is Computers and Thought. This book will lead you to the original sources (e.g., Slagle's SAINT was originally a Ph. D. thesis, as were many others), and also to many and various references as you read through the articles and scan through the Index to the Bibliography. Minsky's survey article is not to be missed! The host of Newell - Shaw - Simon papers, in RAND reports, in published literature, in the library, should be read. See Feigenbaum's short survey article in the IEEE Information Theory Transactions, plus other review articles in these Transactions. See the volumes Self -Organizing Systems (1960 and 1962). Also, the numerious JCC Proceedings and ACM Conference Proceedings. Bobrow's papers is Project Mac TR-1. Hiller and Isaacson wrote a book, Experimental Music.

For Section 2.2 prime reference again is Computers and Thought. See also Newell and Simon in the Handbook of Mathematical Psychology, chapter on Computers in Psychology. EPAM papers will be put in the library. Hunt has a book, Concept Formation: An Information Processing Problem. See also Tompkins and Messick, Computer Simulation of Personality.

For Section 2.4 literature abounds. Minsky's "Steps" article is a good start.

For Section 2.5, ditto. See, also, Sebestyen, Decision Making Processes in Pattern Recognition. Look at Selfridge's Pandemonium and Uhr's work (cf. Computers and Thought),

For Section 2.6 - purposely vague, For example, see Ashby, Design for A Brain(second edition); Automata Studies (Shannon and McCarthy).

For Section 3.1 see Computers and Thought. Raphael's SIR is Project Mac TR-2. Black's report is in Fran Thomson's office.

For Sections 3.2 and 3.3, scan recent issues of the NSF publication "Current Research and Development in- Scientific Documentation" as a means of ascertaining state of art and diving into literature.

For Section 3.4, ditto, perhaps, for this. Look at survey article by Dick See of NSF in the magazine, "Science", sometime in Spring of 1964, Use as springboard to literature. See also, Bar-Hillel's famous article, highly critical of MT work (Advances in Computers, Vol. 1)

On pages 521-523 of Computers and Thought there is a list of collections and special proceedings and symposia that are relatively dense in articles with which you should be somewhat familiar. Try to get a look at as many as you can.

With respect to Sections 4 and 5, the following references are recommended:

Davis - Computability and Unsolvability

Suppes - Introduction to Mathematical Logic;.: Axiomatic Set Theory

McCarthy - A Basis for a Mathematical Theory of Computation in Computers and Formal Systems

- Towards a Mathematical Science of Computation, ICIP, 1962.

- A Formal Description of a Subset of ALGOL, AI Memo.

- Problems in the Theory of Computation, ICIP 1965, AI Memo.

STANFORD UNIVERSITY

COMPUTER SCIENCE DEPARTMENT

March 10, 1965

Syllabus for the Ph.D Examination in Computer Systems, Programming Systems and Programming Languages

The student taking the examination should have a background in the following areas:

1. Computer Components and Memories

1.1 A general familiarity with the elementary physical properties of the following devices which make them useful in digital circuitry: tubes, transistors, diodes, tunnel diodes, integrated circuits, cryotrons, thin films, and magnetic cores.

1.2 An understanding of the principles of operation of mechanically scanned memories (tape, drum, disc, card) and electrically scanned memories (core planes).

2. Logical Design and the Structure of Digital Computers ,

2.1 Some familiarity with Boolean Algebra and the synthesis of Boolean Expressions (this assumes some knowledge of elementary minimization procedures; Karnaugh or Veitch diagrams).

2.2 A knowledge of the properties of elementary sequential circuits (flip flops, shift registers and counters).

2.3 An understanding of the common number systems used in present machines (sign magnitude, l's complement, 2's complement and binary coded decimal). The student should have some appreciation and familiarity with the fact that many algorithms exist for performing arithmetic in various number systems.

2.4. Computer Organization

An understanding of the basic organizational features of a Von Neuman Sequential Processor (the 7090 as an example) a Stack Machine (the B-5000 as an example) and an understanding of how data flows through a machine; and how I-O can be organized.

3. Programming Languages

3.1 A thorough familiarity with ALGOL 60 and B5500 ALGOL.

3.2 A general familiarity with some other programming languages such as FORTRAN, LISP, etc. and at least one machine code.

3.3 A knowledge of some of the more ustf'ul features of Iverson's notation.

4. Programming Systems

4.1 A general familiarity with the principles and organization of assemblers, interpreters and compilers and the knowledge of the problems connected with the implementation of various computer languages, in particular ALGOL (the mechanisms for compiling expressions, procedures etc., and the problems of storage allocation).

4.2 An understanding of the general principles and tasks of supervisory systems and the problems involved with time-sharing a computer and with parallel processing.

4.3 A thorough familiarity with at least 2 computers, and a *general* familiarity with the capabilities and organization of a machine of the "new generation",

5. Phrase Structure Languages

A knowledge of production grammars, the specification of a programming language by a syntax (BNF) parsing of sentences of phrase structure languages, and syntax directed compiling.

REFERENCES

A partial list of useful references is included below. Other references of a similar nature can be found in the library.

Components, Memories, Logical Design and Computer Structure

Braun, Digital Computer Design

Phister, Logical Design of Digital Computers

Buchholz, Planning a Computer System

Ledley, Digital Computer and Control Engineering

Beckman, . . . "Development in the Logical Organization of Computer Arithmetic • @ Control Units", Proceedings of the IRE, January 1961, pp 53.

Rajchman, "Computer Memories: A Survey of the State of the Art", Proceedings of the IRE, January 1961, pp 104.

Mac Sorely, 'High Speed Arithmetic in Binary Computers", Proceedings of the IRE, January 1961, pp 67.

Burks, A.W., Goldstint, H.H., and von Neumann, J. "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", Collected Works of Von Neuman.

(References, cont'd)

Critchlow, A.J., "Generalized Multiprocessing and Multiprogramming Systems", AFIPS Conference Proceedings (FJCC - 1963), pp 107-126.

Amdahl, . . /'Architecture of the IBM System 1360", IBM Journal of Research and Development, Vol. 8, No. 2, April 1964, pp 87 - 101.

"The Structure of System 1360", IBM Systems Journal, Vol. 3, N's 2,3, 1964.

Barton, "A New Approach to the Functional Design of a Digital Computer', Proc. WJCC 19, (1961) pp 393 - 396.

Programming Languages and Systems

Barton, R.S., "A Critical Review of the State of the Programming Art", AFIPS, Conference Proceedings (SJCC - 1963) pp 169 - 177.

Bobrow, Daniel and Raphael, Bertram. "A Comparison of List Processing Computer Languages", Vol 7, No, 4, April, 1964, Comm ACM pp 231 - 240.

Floyd, 'R.W., "The Syntax of Programming Languages - A Survey" IEEE Transactions on Electronic Computers, Vol, EC-13, no. 4, August 1964.

Iverson, A Programming Language, Wiley, 1962

Rosen, Saul, "Programming Systems and Languages, A Historical Survey", AFIPS Conference Proceedings (SJCC - 1964), pp l-15.

Survey of Programming Languages and Processors", Comm ACM 6,3 (March 1963), PP 93-99.

Survey Issue on Programming Languages: IEEE Transactions on Electronics Computers, Vol, EC-13, No, 4, August 1964.

B. Randell and B. Russell; "ALGOL 60" Implementation; AP 1964

STANFORD UNIVERSITY
COMPUTER SCIENCE DEPARTMENT

March 3, 1965

Syllabus for Ph.D. Examination in Numerical Analysis and Computational Mathematics

1. A general familiarity with the following computer problems: solving one or more linear or nonlinear equations; simple problems involving ordinary or partial differential equations; approximation of data by polynomials; approximating integrals and derivatives of functions by linear formulas; locating maxima of functions; computing eigenvalues of matrices. The subjects and the supporting Mathematics or Statistics should be known at a depth like that of courses 137 and 138 and the following books: Hamming, COMPUTING FOR SCIENTISTS AND ENGINEERS; Henrici, ELEMENTS OF NUMERICAL ANALYSIS; Anonymous, MODERN COMPUTING METHODS, 2nd ed.; Stiefel, ELEMENTS OF NUMERICAL ANALYSIS. For computational methods of linear algebra, see Forsythe, NOTES ON COMPUTATIONAL METHODS OF LINEAR ALGEBRA (dittoed notes for course 137, 1964, on reserve in Computer Science library), or L. Fox, AN INTRODUCTION TO NUMERICAL LINEAR ALGEBRA. For solving parabolic and elliptic partial differential equations, see D. Young's Chap. 11 of John Todd (editor), SURVEY OF NUMERICAL ANALYSIS, or "A survey of numerical methods for parabolic differential equations," by Jim Douglas Jr., in ADVANCES IN COMPUTERS, vol. 2 (1961).

2. A reasonable understanding of the pragmatics of scientific computation with automatic digital computers: the importance of fully automatic procedures for frequently used computations; pitfalls in using standard algorithms of mathematics on computers with (necessarily) limited precision; what constitutes a good and well documented algorithm, and where such algorithms can be found for various problems; the different types of errors in computation, and ways to estimate and (where possible) reduce the errors; the influence of the logical design of computer hardware and software on the accuracy and cost of computation in time, storage, and human effort.

3. A deeper knowledge of some selected area of numerical analysis, with a depth like that of courses 237a, b. Examples: discretization error and stability in solving ordinary differential equations (Henrici, DISCRETE VARIABLE METHODS IN ORDINARY DIFFERENTIAL EQUATIONS), round-off error (Wilkinson, ROUNDING ERRORS IN ALGEBRAIC PROCESSES), solution of partial differential equations (Varga, MATRIX ITERATIVE ANALYSIS, or Forsythe-Wasow, FINITE-DIFFERENCE METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS), approximation (Davis, INTERPOLATION AND APPROXIMATION, or J. R. Rice, THE APPROXIMATION OF FUNCTIONS), computation methods in linear algebra (Wilkinson, THE ALGEBRAIC EIGENVALUE PROBLEM), numerical integration (Krylov, APPROXIMATE CALCULATION OF INTEGRALS); Monte Carlo methods (Hammersley and Handscomb, MONTE CARLO METHODS).

4. Familiarity with the principal reference books for mathematical computation --bibliographies, tables, collections, formulas and algorithms, specialized monographs, etc.