RUTGERS UNIVERSITY
The State University of New Jersey
LIVINGSTON COLLEGE
DEPARTMENT OF COMPUTER SCIENCE

PROPOSAL FOR A PH.D. PROGRAM
in
COMPUTER SCIENCE

Saul Amarel, Chairman
Department of Computer Science

January 1970
TABLE OF CONTENTS

HIGHLIGHTS ........................................ iii - vi

I. THE FIELD OF COMPUTER SCIENCE
A. The global view .................................. 1
B. The local view ................................... 11
   (a) Representations in high level language of problems, data, and procedures in an application area ............ 13
   (b) Theory of computation, analysis of algorithms .......... 17
   (c) High level languages for different application areas; schemes for structuring data and procedures ............... 21
   (e) Machine level languages, storage schemes and programming mechanisms; machine organization schemes; executive and control schemes ............... 21
   (f) Theory of formal languages; Automata theory; Switching theory; Theory of electronic digital circuits ............... 23
   (g) Hardware systems; Operating Systems; Translators .......... 24
C. Types of activity and training in Computer Science. PhD programs in other universities ................ 25

II. COMPUTER SCIENCE AT RUTGERS
A. The undergraduate program in Computer Science ........ 28
B. The present M.S. program in Computer Science ........ 28
C. The proposed M.S. program in Computer Science (to start in the Fall of 1970) ..................... 29
   (a) Objectives. Rationale ......................... 29
   (b) Curriculum. Study plans ....................... 31
   (c) Requirements for admission, scholastic standing, and graduation .................. 34
      c1. Admission .................................. 34
      c2. Scholastic standing ......................... 35
      c3. Graduation requirements for M.S. .......... 36
(d) Transition from the present graduate program to the proposed M.S. program ........................................... 39

D. Activities relevant to Computer Science in other University Departments ................................................. 40

E. Center for Computer and Information Services (CCIS) ........................................................................... 41

F. Educational Computing Center (ECC) .................................................................................................... 43

III. PhD PROGRAM IN COMPUTER SCIENCE AT RUTGERS

A. Need for a PhD program in Computer Science at Rutgers ................................................................. 44

B. Goals and approaches of the proposed PhD program ........................................................................ 46

C. Emphasis of the proposed PhD program ............................................................................................. 47

D. The Faculty. Current research in the Computer Science Department .................................................. 49

E. Curriculum. Study plan ......................................................................................................................... 49

F. Thesis Research .................................................................................................................................... 52

G. Requirements for the PhD degree ......................................................................................................... 53

H. Requirements for Admission and Scholastic Standing ........................................................................ 54

I. Master of Philosophy. .............................................................................................................................. 55

J. Outline of typical doctoral students program .................................................................................... 55

IV. PEOPLE AND RESOURCES

A. Students ................................................................................................................................................ 58

B. Faculty .................................................................................................................................................. 58

C. Resources ............................................................................................................................................. 58

Appendix I. Undergraduate Courses in Computer Science 1969-70

Appendix II. Graduate Courses in Computer Science 1969-70

Appendix III. Graduate Faculty of the Computer Science Department

Appendix IV. Proposed modification of the current list of graduate courses in Computer Science (which is given in Appendix II).

Appendix V. Descriptions of the modified or new graduate courses that are listed in Appendix IV.
HIGHLIGHTS OF THE PROPOSED PROGRAM

The introduction of a doctoral program in computer science at Rutgers would mark the completion of an important first stage in the Department's development, it would complement and strengthen our present undergraduate and MS programs in the field, and it would enhance computer research and utilization in the University and in other State institutions.

There is a recognized need to educate people at the PhD level with sufficient depth of knowledge in computer science, as well as with a broad enough vision to explore applications of computers in significant scientific and social problems. Many leading institutions are offering PhD programs in computer science. Rutgers is in a good position to join them with a program that has its own distinctive character. In our program we propose to emphasize the relationship and mutual impact of computers and application areas. This means emphasis on problem types, algorithms and procedures; their relationships to classes of applications; their expression in high level languages; the design and processing of such languages; and the design of systems for man-machine interaction. We envision a close interaction between theoretical work and computer experimentation. We would encourage the treatment of computer science problems in the context of advanced applications in the behavioral sciences and in problems of decision and design that occur in medicine, education, urban planning and engineering.
We have redesigned our present graduate curriculum to provide a comprehensive and well structured initial basis for the proposed doctoral program. The new curriculum is based on a partition of the field into the following four main topics of study:

1. Hardware systems (including switching theory, automata theory, machine organization).

2. Software systems (including programming techniques, data structure and programming languages, translators, theory of formal languages, operating systems, system modeling).

3. Numerical applications (including numerical analysis, linear process, computational aspects of statistical analysis).

4. Non-numerical applications (including combinatorial algorithms, computers in simulation and in mathematical research, graphics, artificial intelligence).

We are proposing to add a seminar and a "selected problems" course that would provide opportunities to discuss and review current research in the field, and also to work on substantial computer projects.

PhD students are expected to complete successful course work for a minimum of 48 credits including one required course (3 credits) in each of the four topics of study and one seminar (for at least 3 credits). Except for the minimal base of required courses (15 credits) that is designed to provide perspective in the field and a view of current research, the student's program is flexible and it can be shaped to fit his individual interests. Students will be encouraged to take relevant
courses in other departments (especially in Mathematics, Statistics and Philosophy). We propose to stress independent study, writing review and research papers, doing active work on computers, and interacting with computer people in Rutgers and elsewhere, and also with people in other disciplines. Acceptance to PhD candidacy would require, in addition to course work, reading knowledge in one foreign language (Russian, French, German or Japanese) completion of a significant project in systems software or in an application area, and passing a two-part qualifying examination. The first part would be written and it would consist of questions covering all four topics of study in the field; this part would be identical with the final Masters examination, and it may be taken as soon as a student has completed 24 credits of course work. The second part of the qualifying examination would be oral, it would test the student's knowledge in his chosen area of concentration and it would be scheduled soon after the student has completed his course work. Thesis research would be a major part of the PhD program (24 credits). The thesis should cover original investigations of one or more problems in the student's area of concentration. Current research by the graduate faculty is expected to stimulate doctoral research in the areas of artificial intelligence, question answering, numerical analysis, computer-assisted mathematics, graphics, theory of programming languages, language processors, description languages, and computer-aided design.
We are proposing certain changes in our MS program, so that it would articulate well with the PhD program. MS students are expected to complete 24 credits of successful course work and a Masters Thesis (6 credits), or 30 credits of course work and a satisfactory essay. The student's course work must include one required course in each of the four topics of study in the field (a total of 12 required credits). The MS degree would be awarded after passing a final written Masters examination. The examination would be identical with the first part of the PhD qualifying examination. The courses taken for the MS degree count also toward the PhD degree. We propose that a M.Phil. degree be available for students who have completed with distinction all the requirements for PhD candidacy but they have not offered a PhD thesis.
I. THE FIELD OF COMPUTER SCIENCE

Computer Science is a young and rapidly expanding discipline. It began to assume an independent identity in American universities in the late fifties, as the digital computer (scarcely ten years old at the time) was reaching extraordinary rates of penetration in government, science, technology, and business, and as a body of experiences and concepts was emerging that pointed to the intellectual and social significance of a computer-oriented culture.

In order to clarify the nature of computer science, it is useful to consider two views of the field, a global view which attempts to capture broad characteristics of the field and its relationships to other fields, and a local view which focuses on the inner structure of the field. The global view is concerned with the kinds of objects, phenomena, and concepts that form the domain of discourse in computer science, and also with the pattern of relationships between this domain and other domains of study. The local view is concerned with the kinds of knowledge, problems, and activities that exist within the discipline, as well as the relations between them.

A. The global view

Computer science is concerned with information processes, with the procedures and information structures that enter in representations of such processes, and with their implementation in machines. It is also concerned with relationships between information processes and classes of problems that give rise to them, and with general methods for solving problems with the help
of information processing machines.

The computer and the phenomena surrounding it are the main objects of study in computer science. Work in the field is focused on the structure and operation of computers, on the principles that underlie their design and programming, on effective methods for their use in different classes of information processing tasks, and on theoretical characterizations of their properties and limitations. Also, a substantial effort is directed into explorations and experimentation with new computer systems, and with new domains of intellectual activity where computers can be applied.

The main concern in the field is with man-made information processes and with systems that are designed to achieve desired goals. An essential part of this activity is the understanding of the variety of possible elementary processes and possible schemes of combination that can be used in the design of information processes. The study of information processes that occur in nature can contribute substantially to such an understanding. To specify an information processing system that satisfies desired goals, we also need a rationale and a set of methods for choosing from the space of possibilities the one which is most appropriate for the attainment of desired goals. To physically implement a desired information processing activity, we need in addition convenient and flexible schemes for transforming the abstract specification of an information process into an operating machine. The stored program digital computer provides such a scheme, and this is the main reason for its centrality in the field.

There exist in nature important information processes that are of great interest to computer science; for example, percept-
ual processes in the nervous system, and cellular processes that
direct protein synthesis via interpretation of genetic informa-
tion. An understanding of these processes enriches the pool of
basic concepts and schemes that are available to computer science.
In recent years, research in information processing in the ner-
vous system has stimulated the study of new logical building
blocks for computers (e.g., threshold logic). In turn, applica-
tion of computer science models to neurophysiology has resulted
in theoretical insights into important neural structures (e.g.,
visual pathways, reticular formation). Similarly, automata
theory, a theoretical component of computer science, is beginning
to provide promising models for the study of processes that trans-
cribe, translate, and control genetic information in cells.

The central role of the digital computer in the field is due
to its near universality as an information processing machine.
With enough memory capacity, a digital computer provides the basis
for embodying any information processing machine, provided that
the task to be performed by the machine can be specified in the
form of a program that can be stored in the computer memory. In
addition to a memory, the other key component of a computer's
structure are a processing unit wherein a set of elementary informa-
tion processes are carried out, and a controller that directs
the pattern of operations in the processing unit in accordance with
the program which is stored in memory. The stored program digital
computer enables us to conveniently build any number of information
processing machines where operation and structure can be easily
separated from each other. A program in the computer memory re-
presents the operation of a specific machine, and the controller
together with the processing unit provide the structural elements
that carry out the desired operation.

In addition to the fundamental property of universality which is inherent in the digital computer scheme, there is another reason - of pragmatic and methodological significance - for the centrality of digital computers in a science of information processing and problem solving. Today's computers are concrete physical devices that carry out complex information processes in reasonable times (minutes and hours rather than days and years). This way they provide a tangible focus for ideas that can be developed and tested over a scale of time which matches well normal human attention spans. Furthermore, they stimulate the creation of new concepts, and they permit a sustained and critical examination of concepts and of models through physical experimentation. In many complex processes and models, where the number of variables is large and their interdependence is high, computer runs provide the only possible method for gaining reliable insight into the phenomena involved. In his 1968 ACM Turing lecture, R. W. Hamming speaks forcefully of the centrality of the computer in the field*: "At the heart of computer science lies a technological device, the computing machine. Without the machine almost all of what we do would become idle speculation, hardly different from that of the notorious Scholastics of the Middle Ages."

There are two major strands of activity in computer science, one oriented to synthesis, exploration, and innovation, and the other oriented to analysis, search for fundamental principles, and formulation of theories. A continuous interaction between these two strands is essential for a vigorous rate of progress in the field. The activities in the synthesis-oriented strand have a strong pragmatic flavor, and they have been largely responsible for major advances in computers and for the great diversity of areas in which computers are being applied. Much of the work in the analysis-oriented strand is directed to understanding the exploratory and constructive activities in the field, and to developing conceptual guidelines for more efficient and more powerful designs and uses of computers.

Experimental work in computer science requires extensive use of computers, and it often stimulates new developments in computer design and utilization. Typical experimental activities may involve the development and evaluation (via computer simulation) of a new memory configuration for computers, or the development of a language for convenient communication with the computer in problems of urban planning. In these activities, workers in computer science are heavy consumers of computer resources. A close coupling with a computer center is necessary for success in this type of experimental work. In turn, experimentation in computer science tends to strengthen the tools and facilities of the computer center, and to raise the general level of its capabilities.

Theoretical work in computer science relies on several branches of mathematics and logic. A typical theoretical problem may focus on the characterization of a specific class of computer procedures (for example, algorithms for syntactic analysis in a class of
computer languages), the analysis of their structure, and the establishment of bounds on the storage space and time that they require for execution. Another type of problem may seek to demonstrate that the utilization of a given restrictive rule in a computer procedure for proof finding (say in the first order predicate calculus) does not affect the completeness of the procedure. In these examples, the objects of discourse in the theory are computer procedures and their properties. Just as mathematics is used in physics to develop theories of certain physical processes, mathematics and logic are used in computer science to develop theories of information processes. Furthermore, mathematics and logic are likely to receive stimulation from their use in computer science, as Analysis was stimulated by Mechanics.

There is a closer bond between computer science and mathematics: a common concern with formalism, symbolic structures and their properties, and emphasis on a set of general methods and tools that can be used in a great variety of situations. These connections between computer science and mathematics, as well as with other closely related disciplines, are well described by A. Newell in a memorandum to the Committee On Support of Research In the Mathematical Sciences*:

Computer science shares with mathematics a concern with formalism and a concern with the manipulation of symbols. It also shares with mathematics the role of handmaiden to all of science and technology. It shares with electrical engineering the concern with the design and construction of information processing systems that accomplish ends. It shares with all of engineering a concern with the process of design, considered as an intellectual endeavor.

It shares with linguistics a concern with language and communication. It shares with psychology a special concern with forms of information processing that result in intelligent behavior, broadly viewed. It shares with the library sciences a concern with how to store and retrieve large amounts of information, either as documents or as facts."

The physical structure of a computer (the hardware) consists largely of electronic building blocks that implement a variety of switching, storage, and communication functions. In the logical design and system design of computers, the concern is with the choice of hardware building blocks, and with their local and global organization in the light of given operational requirements. These design areas are important subjects of study in computer science. They also have strong points of contact with work in electronics and in solid state physics which is concerned with the physical aspects of computer hardware. Frequently, new developments in computer electronics, and in manufacturing technology impose new requirements on logical design. Also, new concepts in logical design and new modes of computer utilization are likely to stimulate new developments in computer hardware. Thus computer hardware is a natural domain of cooperation between computer science and electrical engineering.

One of the important reasons for the concern of computer science with the process of design is due to the strong synthesis-oriented component in the discipline. Computer systems are highly complex man-made entities that are intended to operate effectively and efficiently over a wide range of problem environments. The planning and design of these systems can be guided only in part by systematic knowledge, and it involves many of the uncertainties and the methodological difficulties that people face in such areas as in city planning, the design of communication
or transportation networks, or the planning of educational systems. Thus advances in methodologies of design for complex systems are of great relevance to computer science.

There is a more fundamental reason for a close coupling between computer science and a science of design. It comes from the concern of computer science with the intellectual processes of problem solving and goal-oriented decision-making that are present in design processes. The concern here is not with understanding a given state of nature but with understanding a process that involves conceiving alternative states, evaluating the alternatives, and deciding on a course of action in the light of given goals. Complex symbolic representations of situations, of desired conditions and of relevant facts, processes of search and decision-making, procedures for deductive reasoning, and methods for evaluating merits of partial solutions, are all objects of study in computer science, and they are also key components of design processes*.

The development of computer science has been strongly stimulated by demands for the application of computers in a wide variety of new areas. There exists a "moving front" of computer applications that started its journey in the late forties, and has traversed since then a path which goes roughly as follows: numerical calculations, processing routine business data, assembly and compilation of programs, managing large data bases, modeling and simulation of complex systems, and performance of complex diagnostic and design tasks. In parallel with this movement of the applications front, there has been an evolution

* A case for the importance of a science of design in a world filled with complex man-made systems, and a discussion of the close kinship between such a science and computer science, are convincingly presented by H. Simon in his recent book, "The Sciences of the Artificial," MIT Press, 1969.
of machine designs, programming schemes, computer languages, types of algorithms, and approaches to problem solving. More fundamentally, the conception of the computer has evolved from that of a rapid calculator to that of a flexible symbol manipulator capable of performing fairly demanding intellectual tasks. The moving front of computer applications is an essential domain of activity for workers in computer science. The challenges presented by new applications, and the constructive attempts to meet them, are key factors in the growth and maturation of the field.

There is a missionary-exploratory attitude in computer science as it interacts with other disciplines at the moving front of computer applications. Much of it is motivated by the interest in identifying significant new classes of problems and phenomena in information processing. There is also a strong interest in testing the reach and limitations of current ideas and techniques in the field.

At present, new computer applications are being explored in almost every discipline: in mathematics (investigations of finite topological spaces, coset enumerations); in logic (decision procedures, methods of deduction); in statistics (properties of stochastic processes); in physics (control of experiments, bubble chamber analysis, organizing new bodies of experimental data, simulation of plasmas); in chemistry (organic synthesis, interpretation of spectrograms, formation of structural models for large molecules); in different branches of engineering (computer-aided design, simulation); in architecture (graphic representations, design); in urban and regional planning (transportation networks, design, simulation, optimization); in the social and behavioral sciences (management of data bases, modeling and simulation); in education (computer-aided instruction, resource
scheduling); in biological sciences (organizing experimental
data bases, study of models, morphological analysis of cells);
in medicine (managing data bases, diagnostic processes, psychiatric
interactions); in law (organizing and managing statutes); in
language and literature (phonology, grammatical structures,
studies in style); in the arts (music composition, pattern analysis).

An important application for computers, which is of special
interest to computer science, is the design of more powerful,
efficient, and easy-to-use computer systems. Developments in
system software (programs for translating between computer lan-
guages and for managing the operation of a computer) and in
time sharing systems are in this area. The use of computers in
the study of computers and in their improvement is a powerful means
for gaining the knowledge and insights that computer science
seeks, while at the same time the field is being bootstrapped.

In most of the activities at the frontier of computer ap-
lications, there is collaboration between workers in computer
science and specialists in the respective discipline. Through
this type of collaboration, the contact between computer science
and other "computer consuming" disciplines (essentially all
disciplines that involve any intellectual activity) is likely
to continue for some time.

One of the most significant effects of computer penetration
in a discipline is an increase in clarity of the concepts, pro-
blems, and methods in the discipline. In some cases, major
changes in point of view take place that affect in a fundamental
way the theoretical frameworks in the discipline; such is the
case in parts of psychology, where models of cognitive processes
are being developed in the image of computer processes.
From the experience of work with different computer applications, a body of knowledge is growing which includes properties of different classes of problems and solution methods, and characteristic patterns of information processes. Theoretical questions are beginning to emerge about methodologies of inquiry, and about possible modes of transforming knowledge into decisions during problem solving processes. The study of these questions is beginning to create points of contact between certain classical parts of Philosophy (nature of knowledge, methodology, epistemology) and Computer Science.

In bringing a computer to bear on a "real life" problem, the problem must be formulated in a form that can be accepted by the computer, and furthermore, computer methods must be available for approaching the problem. Bridging the gap between a real life problem and its representation within a system wherein the problem can be studied and solved, is a complex intellectual activity which is characteristic of much work in applied mathematics. This involves the formulation and study of models, the creative choice of viewpoints for approaching a problem, and the development of broad methodologies for problem solving. In these areas, there is strong overlap between applied mathematics and computer science.

B. The local view

In the following we present a view of the internal structure of the field which we have used as a conceptual framework for the design of our graduate curriculum. The concept of a procedure -- an algorithm -- is of primary significance to computer science. A computer program is a procedure in a computer language that expresses a method of carrying out a desired information processing function. The language used to formulate the procedure must be "understandable" by the computer, i.e., any statement in the language must be such that it can be appropriately interpreted by the computer.
A typical information processing function is to find solutions for a given class of problems. Examples of such problems are: invert a matrix, sort given data, translate between a pair of computer languages, find the optimal location for a regional highway, decide on a medical diagnosis, establish whether a given statement is a logical consequence from a given body of premises.

Communication of a procedure to computer hardware must take place in "machine level" language, but such a language is usually too cumbersome for expressing concepts that enter in the formulation of procedures for many problems of interest. At present, there are several application areas where a procedure can be formulated in a "high level" language which is designed for convenient communication of concepts and operations that are characteristic to the area. For each high level language, there are translators into machine level language that take a procedure from a form which is convenient to man to a form which can be directly handled by the machine.

The intellectual effort that goes into the formulation of a procedure in a given language, regardless of the level and conceptual power of the language, consists of finding a combination of steps that can be expressed in the language and that collectively represent a correct method of solution for a given class of problems. Usually, one looks for a combination of steps which is optimal in some well-defined sense. An important part of the procedure formulation effort involves analysis and validation of proposed procedures. A body of theory is slowly emerging for guiding these evaluation processes. A theoretical treatment of a procedure's validity is more likely to succeed when the procedure is formulated in a language where the essence
of the method expressed by the procedure can be clearly seen. The situation becomes much more difficult when the formulation language is close to machine language. The evaluation of a procedure must rely then on experimental tests on a computer. Considerable effort is now being directed to experimental methods for testing, validating, and "debugging" procedures.

The conception, formulation, computer implementation and evaluation of procedures for a broad variety of problems constitute a major part of the activity in computer science. Closely associated with these activities are efforts to develop schemes, means, and tools for building and executing programs—such as languages, major principles for structuring procedures, programming mechanisms, computer organizations, and design aids to facilitate these efforts. In addition, a significant amount of effort is directed to the design of advanced systems (software and hardware). All these activities have important connections with several theoretical efforts in the field—some in application areas, and others in the analysis of algorithms, in formal languages, automata theory, switching theory, and system analysis.

In Figure 1 we show diagrammatically the major areas of activity in computer science and certain relationships between them. Activities are shown as boxes in the diagram. A line connecting two activities stands for a relationship of close conceptual coupling between the activities. This means that such activities have closely related subject matter, and the state of knowledge of the one is highly relevant to the state of knowledge of the other.

(a) **Representations in high level language of problems, data, and procedures in an application area.**
Figure 1. Structure of major areas of activity in Computer Science.
The main problems in this area are to find solution methods for classes of problems in different domains of application, and to formulate them in a suitable high level computer language. As mentioned previously, it is difficult to see any limits to the possible extent of computer applications in any area of intellectual activity. Computers will be used in all disciplines, and for a great variety of tasks. From the point of view of computer science, it is useful to classify applications not by discipline or subject matter, but by the kinds of problem solving methods, information structures, and procedures that are characteristic of the problems in the application.

In Figure 2 we show a set of nine major application areas that have achieved a reasonable degree of distinct identity from the viewpoint of computer science. In the figure, we have organized the application areas into a connected structure. The connections are intended to denote conceptual closeness between the areas in terms of solution methods, types of data and procedures, and schemes of processing. For example, the area of Numerical Calculations is close to the areas of Statistical Methods, Mathematical Programming, Optimization Problems, and Modeling and Simulation; the area of Formula Manipulation, Combinatorial Processes, and Search is close to the areas of Text Processing, Information Systems, Artificial Intelligence, and Modeling and Simulation. In the area of Modeling and Simulation, concepts and techniques from the area of Numerical calculations meet with concepts and techniques from Combinatorial Processes and Search, as well as with concepts
Figure 2. Application Areas from the viewpoint of Computer Science
from other areas. In general, every area has some points of contact with the other areas; our connections in the diagram are intended to indicate the major couplings only.

Now a specific class of problems in a discipline may contain elements that are characteristic of several of our application areas. As the scope and complexity of "real life" computer tasks continue to grow, they will become less and less distinguishable in terms of content in one or more of our areas. In the seventies we can expect several major computer applications that require concepts and techniques from all the areas shown in Figure 2. For example, consider a medical application where the objective is to build and manage a data base about a class of diseases and about a group of patients, and furthermore the data base is to be used for medical education and for diagnosis. From the viewpoint of computer science, this type of application, which is considered to be in the research and development stage at present, presents problems in (at least) the areas of Information systems, Artificial Intelligence, and Statistical Methods. A complex task of design (say, in the context of regional planning) may pose important problems in all the nine application areas. Yet components of the large task may be strongly identified with specific application areas, and thus the knowledge in each area may be effectively utilized in the problems of each component. The dissection of a large task into parts, in a manner that would permit the best utilization of existing knowledge for the treatment of each part, requires deep knowledge of the concepts and techniques in each of the major application areas.
This also presupposes the ability of taking the "real life"
task, decomposing it in various ways, and representing it
in a manner which is appropriate for computer processing.
In the seventies, we will need an increasing number of
people with this kind of knowledge and skills.

Consider now a problem domain in a given application
area. Suppose that there exist high level languages in
this area, and also schemes for structuring data and proced-
ures. Suppose further that there exists a body of theory
which is relevant to the problem domain. Several types of
procedure writing activities are possible: The formulation
of some effective procedure in an appropriate high level
language where none existed previously; the formulation of
a procedure in a high level language which is appreciably
better than existing ones (in a sense of "better" which is
intrinsic to the logic of solution in the given problem
domain); and the development of a procedure that responds
to a class of problems which is much broader than one or
more classes for which specific procedures exist already.
In all these cases, the essence of the effort is to find a
method of solving problems in a given domain in terms of the
knowledge and concepts (the theory) in the domain. Most of
the contributions to method finding and procedure writing
in a problem domain are likely to come from specialists in
the domain (discipline) in cooperation with workers in com-
puter science. The relative contribution of computer scien-
tists is especially significant in the initial stages of
computer penetration in a problem domain.

A domain to which computer scientists have been con-
tributing most directly consists of problems in computer system design, both hardware and software. This includes problems in computer-aided design of computer hardware, problems in the design of language processors (such as translators from high level language into machine level language), and approaches to the formulation of procedures for controlling the operation of a computer and for managing its resources. At present, there is vigorous activity in this domain; the level of activity is likely to grow in the next decade.

(b) **Theory of computation; analysis of algorithms.**

There is a body of theory from mathematical logic which is concerned with computability and recursive functions. The emphasis in this area has been to establish whether there exist processes that can compute certain broad classes of functions. The concepts and the formalisms of this theory have contributed general insights and clarifications to problems of computation, and they have provided useful frameworks for formal approaches in computer science. However, the distance between the concerns of the theory and the main problems of computer science remains large.

There are beginnings of a movement towards the study of particular classes of algorithms and their properties; for example, characterizations of the complexity of classes of computations, and of relationships between the structure of a class of algorithms and a class of computational tasks (e.g., the work on Perceptrons by Minsky and Papert). There are also beginnings of significant research on whether an
algorithm ever terminates for inputs satisfying certain conditions, on the equivalence of two algorithms, on transformations of algorithms that preserve equivalence, and on the semantics of languages in which algorithms are formulated (e.g., the work of McCarthy and his colleagues at Stanford). These theoretical developments are of great importance for activities of procedure construction, debugging, and evaluation (discussed in (a) above).

As computer science matures, one of its important objectives is to develop models and theories of computation that are responsive to the major design problems in the field. This requires borrowing some fundamental ideas and approaches from work in mathematical logic, but also fashioning and extending the ideas in a manner which is most appropriate to the field.

(c) High level languages for different application areas; schemes for structuring data and procedures; Language descriptions; Translation schemes.

Given an application area, the major requirement for a high level language in which solution methods (procedures) are to be represented, is convenience and naturalness of expression for the concepts that are used by a specialist in the area in describing his methods and the knowledge which is relevant to them.

Many high level languages have been formulated to date in a variety of application areas. Many more are likely to emerge in the next few years. The development of a high level language usually involves contributions both from specialists in the area and from computer specialists. In
past developments, the major contributions have been made by computer people. Progress of current work on convenient methods for describing high-level languages may bring us to a point where the formulation of a high level language will be mostly done by specialists in the area. The development and study of schemes for language description are an active area of computer research. Innovations in this subject will contribute to the transfer of specific software design activities from the computer area to the different application areas. By facilitating this process, computer science will be making a strong contribution to the effective utilization of computer power in a broad range of problems.

The formulation of a high level language is central to the entry and strong penetration of computers in an application area. Even in well-developed areas, from the viewpoint of computer usage, a high level language provides a flexible means for writing and testing complex procedures within a reasonable period of time. This is especially relevant to the development of system software within the computer industry. At present, considerable effort is being directed to the development of a 'software implementation language' in which procedures that are major part of system software (e.g. compilers, file processors, schedulers) can be conveniently written.

Another kind of contribution which is also directed to strengthening means of expression for solution methods is the development of basic procedure schemes. While a high level language provides means for expressing in detail the individual steps of a method, the development of procedure schemes provides templates for expressing solution methods at the global level.
An important scheme of procedure construction consists of a structured data base and a control procedure that processes input data in accordance with the data base. Many pieces of software have already been fashioned according to this scheme. A prominent example is the syntax directed compiler, where a description of a high level language is stored in a data base, and a control procedure accepts statements in that language and translates them into machine language in accordance with the language definitions in the data base.

Other examples of procedures that are built in the form of a data base and an associated control procedure can be found in computer aided design, in medical diagnosis, and in computer assisted instruction. In the coming years we are likely to experience an enormous growth in the number of procedures of this general type, especially as management information systems and computer utilities will begin to emerge. In the development of such procedures, a major part of the effort will be directed to the formulation of languages in which information can be described for a data base, and to the preparation of the specific content of a data base in a given area. The development of languages for data bases is another major area of cooperation with other disciplines.

It is reasonable to expect that the major method of expansion of computer usage into new application areas will be via the development of convenient frameworks and forms that specialists can use to express their knowledge, work rules, and methods, in a manner that can be "understood" by computers. High level languages, and broad types of procedures that can easily accept and use new knowledge, are major contributions to these problems of flexible man-machine interface.
(d) **Representations in machine level language of problems, data, and procedures.**

The activity in this area is commonly identified with "programming a problem" for a machine. The task of the programmer is to find a procedure in a machine level language that expresses the essence of a given method of solution in a manner that best utilizes the hardware resources of the computer. Programming in machine language is becoming almost exclusively limited to systems software, where the machine environment is an important factor in the design. Human programming in machine language is rapidly becoming an extinct skill. However, it has strong educational value, as it permits a clear view of fundamental computational processes.

(e) **Machine level languages; storage schemes and programming mechanisms; Machine organization schemes; Executive and control schemes.**

A language at the machine level depends heavily on the physical mechanisms and the organization of the computer. It also has many features that are induced by developments in systems software. The specification of a machine-level language is an activity which calls for tight cooperation between hardware designers and software designers.

A contribution which is widely regarded as central to the art of programming is the development of key programming mechanisms. Examples of such mechanisms are pushdown stores, "hashing" in memory, subroutines, and macros. While a machine level programming language provides means for expressing in detail the individual steps of a program, the programming mechanisms provide means for organizing and structuring programs at a more global level.
Innovations in programming mechanisms have major impact on the development of hardware and software systems. At times they introduce completely new modes of computing, and frequently they provide ways to increase the efficiency of computer utilization. Experience has shown that the appropriate environment for such innovations is heavy experimental work in computer programming and a climate of exploration with little or no commitment to specific problems.

The overall organization of the digital computer has changed little over the years. It captures the essentials of the abstract model of computation - the universal machine - introduced by Turing in the thirties. There have been many variations and refinements of the basic scheme; each has provided solutions to the central pragmatic question of how to implement significant computations at reasonable speed, reliability, and cost. This is an area of vigorous activity in the advanced development and design divisions of the computer industry.

There has been a continuous effort to seek new overall organizational schemes for computers. Stimuli for this effort come from new technologies (e.g., integrated electronics induces decentralized designs), from new tasks (e.g., processing visual patterns and problems in meteorology induce parallel computers), and from new modes of computer utilization (e.g., time sharing induces new memory organizations). Work in this area is closely connected with exploratory work on executive and control schemes for managing the operation of computer systems. These efforts are often an integral part of large design activities in operating systems. At present, this general area is one of the most active in the computer field, with computer science departments, industrial laboratories, and various entrepreneurs participating. The emphasis is on innovation, constructive exploration, and promotion.
The introduction of new organizational schemes, and the growth in complexity of system designs, is underlining the importance of systematizing the processes of computer design. Efforts in design methodologies and in computer-aided design are increasing. Methods of systems analysis and simulation are being developed, and they are being used in many proposed designs. The importance of work on computer-aided design of computers is being widely recognized. In addition to its usefulness for developing specific designs, it contributes basic clarifications to processes of computation, and it provides an excellent vehicle for the study of design processes in general.

Efforts in computer-aided design are stimulating new developments in the area of Graphics. The convenient use of spatial information (charts, diagrams) in man-machine interaction is highly significant for the facilitation of computer usage in a variety of disciplines. The situation is similar to the introduction of high-level languages in the field. Work in this area is likely to increase in the coming decade.

(f) Theory of formal languages; Automata theory; Switching theory; Theory of electronic digital circuits.

Theoretical activities in these areas are concerned with properties of computer languages, computer mechanisms, and their realizations. Automata theory has a central position, with strong ties to the theory of computation, and to formal languages and switching theory. It is concerned with abstract models of machines and their possible behaviors. Recent work in formal languages has been strongly stimulated by development in structural linguistics (Chomsky), and is currently a very active area of research. Results in this area are relevant to the design of high-level languages, translators, and programming mechanisms.
Many of the models studied in automata theory and formal languages have little connection with the real situations that are faced by computer and language designers. As computer science is maturing, there is a growing recognition of the need to bring theoretical work closer to computer design and utilization. For such an enterprise to be successful, it is necessary for potential contributors to have sound knowledge of present theories (and of the relevant background in mathematics and logic), and also to be actively exposed to the body of experience that has accumulated in the field, as well as to the concerns of designers and users.

Work in switching theory is concerned with many of the problems that appear in the logical design of computer subsystems. Activity in this area is slowing down, with many of the contributors moving to problems in automata theory and formal languages. Logical design of computers relies on building blocks (the elementary logic gates and storage elements) that are implemented by electronic circuits. Work on the theory of pulse circuits and on the physical processes that are used for switching and storage (solid state, magnetic, and optical phenomena) is relevant to developments in logical design. Many of these activities take place in electrical engineering departments.

(g) **Hardware systems; Operating systems; Translators.**

The computer industry is one of the fastest growing in the U.S. Its main product is computer systems, hardware and software. The planning, design, manufacturing, and maintenance of these systems requires a large number of people with training in computer science. In these activities, many design principles, theories, schemes, and techniques that form the body of knowledge in computer science are being used to satisfy many specific objectives.
C. Types of activity and training in Computer Science. PhD programs in other universities.

In view of the previous discussion, it is useful to distinguish between three general types of activities in computer science: (1) activities concerned with problems, and procedures for solving them, (2) activities concerned with languages, schemes of processing, and design principles, and (3) activities concerned with specific systems, (softwares and hardware). In the diagram of Fig. 1, activities of the first type are roughly at the left of the page, activities of the second type occupy the middle, and activities of the third type are at the right. In each of these activities we can find efforts that range from advanced research to routine design within the state of the art.

An undergraduate major in computer science is expected to perform competently in activities of types (1) and (3) that are within the state of the art.

A graduate of an MS program in computer science is expected to have a broad understanding of the entire field, and to perform well in any of the three types of activity discussed above. Furthermore, he should be able to expand the boundaries of the field, by developing new system designs or new computer applications - mostly within the framework of existing schemes and design principles.

A graduate of a PhD program in computer science is expected to contribute substantially to the advancement of the field. He must acquire the knowledge, the skills, and attitudes that will enable him to do independent research in computer science, and to produce new concepts and designs. He is expected to have the breadth of understanding and the problem solving experience that will enable him to explore new and more
advanced uses of computers in various domains. Furthermore, he must develop an overall perspective of the field, and a sufficient exposure to theory and experience, so that he can contribute to the development of an intellectually coherent discipline. This is an important objective (especially for education in computer science), in view of the rapid growth of the field, which causes fragmentation and accumulation of large amounts of unstructured detail.

Training in computer science at the PhD level is needed currently for positions in universities, independent research laboratories (such as Stanford Research Institute, Rand), industrial laboratories and design groups (such as Bell labs, RCA labs, IBM research labs), government laboratories and research institutes (such as NIH), large computer centers and service organizations and also for planning and administrative positions in government and business.

The recognition of the intellectual and social significance of the computer, and the realization of the critical need for trained professionals, researchers, educators and administrators in the field, have resulted in the creation of tens of graduate programs in computer science during the last decade.* Among the universities having PhD programs in computer science are Carnegie-Mellon, Case Western Reserve, Chicago, Cornell, Illinois, Maryland, Michigan, Ohio State, Pennsylvania, Purdue, Stanford, Texas and Wisconsin. Consideration of these programs,

and of programs in other universities, shows a considerable variety in content and approach. However, the existing programs have a substantial common core of subject matter, orientation and method, and also a unity of thought and concern, that give to advanced graduate work in computer science a recognizable identity.
II COMPUTER SCIENCE AT RUTGERS

A computer science department was established at Rutgers in 1966. The department is in Livingston College, and it offers both undergraduate and graduate programs.

A. The undergraduate program in computer science

The undergraduate program is oriented to majors in computer science, as well as to students in other sciences and in the humanities. The list of undergraduate courses for the current academic year is given in Appendix I. We are working at present on modifications of the undergraduate curriculum with a view to (i) offer a well-structured overall program of studies for majors in computer science, (ii) provide a broad introduction to the computer culture and its social significance to students in the humanities and the social sciences, and (iii) present a working knowledge of computer concepts and techniques to students in the physical and natural sciences, as well as to students in mathematics and engineering. Courses that cover prerequisites for graduate study in computer science will be available in the undergraduate program. At present, about 450 individual students from Livingston, Rutgers and Douglas are enrolled in our undergraduate courses. There are strong indications that the undergraduate enrollment will grow steadily in the next few years.

B. The present M.S. program in computer science

The present graduate program is oriented to an M.S. in computer science. The list of graduate courses for the current academic year is given in Appendix II. About 100 students are presently enrolled in the program; most of them (about 80) are
on a part-time basis. To date, eight students have graduated from our program; seven more are expected to graduate by the summer of 1970.

During the fall of 1969, the graduate faculty of the department, together with graduate students in computer science, have studied in detail the Master's program with a view to (i) clarify the rationale for the program, and re-evaluate accordingly the structure of the curriculum, (ii) determine clear departmental requirements for admissions, scholastic standing, and graduation, and (iii) adjust the program so that it would articulate well with our proposed doctoral pro-
gram. As a result of this study, the graduate faculty of the Computer Science Department has reached several conclusions, and has made several recommendations that are summarized in the next section.

C. The proposed M.S. program in Computer Science
(to start in the fall of 1970)

(a) Objectives, Rationale.

The training of an M.S. graduate in computer science should provide him with the knowledge and skills to hold professional positions in the development and design of computer systems (hardware and software), and in the
design and implementation of new applications software; to hold administrative positions that require planning and evaluation of computer-based systems; to teach in computer science; and to prepare him for further study and research at the doctorate level.

Because of the rapid rate of change in the field, students must acquire the capability to follow the research and professional literature, and to adapt independently to changes in approaches, languages, and systems. Furthermore, he must develop sufficient confidence in his ability to think and work independently, via substantial experience with computer projects of realistic scale.

Another important implication of the dynamic character of computer science is the lack of completely well-structured "core material", and the possibility of sudden developments in the field that both faculty and graduate students may find interesting to follow. This calls for a flexible and adaptable program with a minimum of required courses - just enough to insure a sound perspective over the entire field.
(b) **Curriculum. Study plans.**

From the structure of major areas of activity in Computer Science (shown in Fig. 1), and from the structure of application areas (shown in Fig. 2), we arrive at a partition of the field into four main **topics of study:** (1) Hardware systems, (2) Software systems, (3) Numerical applications, and (4) Non-numerical applications. The first two topics can be identified as covering two somewhat overlapping "neighborhoods" of activities in the diagram of Fig. 1. The "Hardware systems" topic centers on the "Machine organization schemes" activity and the theoretical and design activities connected to it. The "Software systems" topic covers the activities in "Machine level languages," "Executive and control schemes," "Language descriptions and translations" and the activities that are connected to them. The two topics of study that are concerned with types of applications can be seen as covering two neighborhoods in the diagram of Fig. 2; the "Numerical Applications" topic covers the top five areas of application in the diagram, and the "Non-numerical applications" topic covers the bottom four. In addition, each of the applications-oriented topics covers activities that are relevant to it and that are shown at the top of the diagram of Fig. 1; namely, "High-level languages," "Representations in high level languages of problems, data, and procedures," and the theoretical activities that are connected to them.

In view of our dual objective of coverage and flexibility, we are recommending a modification of our current list of
graduate courses that will satisfy the needs of our M.S. program. The differences between the modified list (which is given in Appendix IV) and the current list are as follows:

(1) Five courses that are two-semester sequences at present, are broken into two distinct one-semester courses each, with separate titles. This increases the flexibility of the program.

(2) Thirteen courses have changed titles to better reflect their content and orientation.

(3) Two one-semester courses are added (CS x: "Machine Organization," and CS y: "Numerical Analysis") and a two-semester course which appears in the current list but has not been offered as yet (CS 501, 502 "Programming Systems and Languages") is dropped. The additions are needed in order to provide an essential coverage of required material in two of the four main topics of study. Note that the total number of course-credits remains the same after the modifications.

(4) The new list of courses is structured in accordance with our partition into main topics of study. Each course is associated with a topic of study.

(5) For each main topic of study there is a small set of courses (one or two) that are **required courses**.

We propose to achieve the breadth of coverage in the field by requiring each candidate for the M.S. degree to complete successfully one required 3-credit course in each of the four main topics of study (i.e., a total of 12 required credits).
This requirement appears to be a realistic approximation to a program with a single required course of the "Introduction" or "Orientation" type (of two or more semesters), in which the overall structure of the field, its methods, and current problems, are mapped in sufficient detail so that the student can see the field in reasonable perspective and he can then make informed decisions on his subsequent course of study. Such a course would have many advantages over a small group of independent courses that are intended to cover parts of the field (even well chosen parts, that collectively span most of the field). It is being tried at present, under the name of an "immigration course" at the Computer Science Department of Carnegie-Mellon University. We propose to follow the experience of Carnegie-Mellon in this area, and to attempt at a future time (within the next five years) the introduction of a similar course in our program.

Upon entering the M.S. program, each student will be assigned a faculty advisor, and with his help he will choose an overall study plan which will include (1) four required courses as explained above, (ii) other graduate courses in computer science that will permit the student to go more deeply into one or two main topics of study in the field, and (iii) courses in other departments that are relevant to his main topic(s) of study. The study plan could be revised, in consultation with the advisor, as the student proceeds through his studies.

The total number of credits required for the Master's degree is 30. The student has the option of fulfilling the requirement of 30 credits entirely through course work. The
alternative option is to write a Master's Thesis which covers a research project for which the student receives 6 credits (and the rest of the required 24 credits are obtained from course work). The department will encourage the thesis option in the future.

Normally, a student choosing the thesis option is recommended to take no more than one 3-credit course outside the computer science department; a student choosing the alternative option is recommended to take no more than 6 credits of course work outside the department. However, these are only guidelines, and each student should decide on courses outside the department in consultation with his advisor.

Courses that are relevant to the graduate program in computer science may be taken in the following departments:

Mathematics
Statistics
Philosophy
Electrical Engineering
Industrial Engineering
Library Service

A list of the relevant courses in these departments is given in Appendix VI.

(c) Requirements for admission, scholastic standing, and graduation.

cl. Admission.
In addition to the general admission criteria of the Graduate School, the following departmental criteria are proposed:

(i) A substantial background in mathematics, especially in calculus, linear algebra, and finite mathematics (the equivalent of at least one semester in each of these areas).

(ii) A knowledge of computers and programming equivalent to our undergraduate CS 301, 302.

Results of the aptitude part of the Graduate Record Examination are required, as well as letters of recommendation. The applicant's academic record ought to exhibit at least a B average and ought to show distinction in computer science, mathematics, and related fields.

Good students will be admitted without having had all the requirements in (i) and (ii) above, but they will have to make up the deficiencies by taking, without credit, appropriate undergraduate courses.

People who have been working professionally in computer science, and who do not satisfy the formal departmental prerequisites for admission, will be given special consideration. In many cases, work experience at a high professional level can be substituted for some of the academic prerequisites.

c2. Scholastic standing

Courses with grades of 3 or higher cannot be used in meeting the requirements for a master's degree. Only grades of 1 or 2 in course work (as well as a satisfactory thesis) can be used for the 30 credit requirement. A student is expected to maintain a reasonable
academic standing throughout his course of study. If a student obtains a grade other than 1 or 2 in nine or more of his credits, then his academic performance will be considered poor, and normally he will not be permitted to continue in the program. However, each case will be considered individually by a committee of the department.

**c3. Graduation requirements for MS**

A student may choose either the thesis option or the essay option during the formulation of his study plan. The thesis or the essay must be written in English (except that some portion may require a computer language, with English documentation), it must be the student's own work, and it must demonstrate on the part of the student a facility for expository writing.

**Thesis.** The Master's thesis must be a written account of a critical and scholarly investigation in an area of computer science. It may represent (i) a piece of independent research (extensions and improvements of work in a given part of the field are acceptable, at a level of novelty which is less than that required for a doctoral thesis), (ii) a work of synthesis that gives to old and known results of several workers in the field a new significance and insight, or (iii) an important constructive contribution to the development of a computer subsystem (a new machine organization, a new language) or a computer application (e.g., program for class scheduling, a CAI program). The thesis may not be a digest of known results from the literature, a summary of a published report, company classified or
government classified material, or dependent for its background on other non-available reports. The thesis topic should be chosen by mutual agreement between the student and a member of the graduate faculty in computer science who will be his thesis supervisor. The student's thesis supervisor may be his regular advisor or another member of the faculty. The thesis must be approved by the student's thesis committee. The committee consists of the thesis supervisor and two other faculty members who are appointed by the graduate director in consultation with the thesis supervisor.

**Essay.** The essay must be an expository paper in a field of computer science that was covered in the student's coursework. It may be a paper written as part of a course in computer science or it may be based on such a course. No extra credit is given for the preparation of the essay. The essay need not require the level or original research or scholarship necessary for the thesis, but it must demonstrate, at the very least, a sound understanding of the concepts in the subject area that it covers. The essay must be approved by the graduate faculty member on whose course the essay is based.

In addition to the general requirements for the MS degree that are established by the Graduate School, the department proposes the following requirements:

1. A student offering a thesis must complete successful coursework for a minimum of 24 credits; and he must submit a Master's thesis (for 6 credits) which is approved by his thesis committee. The first
draft of the thesis must be ready for the thesis supervi-
son one month before it is due in the Graduate Office.
The student is also required to give a seminar in the
department presenting his thesis research.

A student offering an essay must complete successful
course work for a minimum of 30 credits and he must sub-
mist an approved essay (in final form) at the end of the
semester before graduation.

2. A student's program must satisfy the required courses
and also the requirements of scholastic standing for the
MS. If a student can demonstrate, by record of previous
study or by examination, that he has mastery in the area
of a required course, then he will not be required to take
the course. This does not mean, however, that he will be
given credit for the course. The idea here is to minimize
work on unnecessary requirements, but to permit the stu-
dent to spend a reasonable period of time in effective
interaction with the graduate faculty over advanced aspects
of computer science.

3. A student must pass a final written examination (the
Master's examination) which will be given at specified
dates once in each semester. The examination is compre-
hensive, and it will cover material in all the four main
topics of study in our program. It is the purpose of the
examination to test the integration of the course work, and
not to be a re-examination of the individual courses. The
emphasis will be on breadth, and understanding of relationships between different areas in computer science. If a student does poorly in the final examination, then he may be advised to take more course work in relevant areas in order to increase his chances of passing the final examination a second time. Taking the final examination for another time, requires the consent of a departmental committee that will be in charge of the Masters examination. Normally, a final examination will not be taken more than twice.

(d) **Transition from the present graduate program to the proposed MS program**

We propose that the modified MS program be put into effect in the Fall of 1970. At present, we have about 100 Master's students in different stages of their work toward graduation. In view of this situation, and in order to allow an orderly transition between the programs, we propose to partition the currently enrolled students into two groups: Those who will have 15 or more credits by September 1970, and those who will have less than 15 credits by that time. Students in the first group will continue their studies under the requirements of the current (old) program. The second group will continue under the requirements of the modified (new) program. The students registering in the Fall of 1970, and subsequently, will start directly with the new program. These proposed arrangements have been discussed extensively with our graduate students during the Fall of 1969, both in working groups and in open faculty
D. Activities relevant to Computer Science in other University Departments

There are several departments in the University that offer courses in mathematical and engineering subjects that are relevant to graduate programs in computer science; these are: Mathematics, Statistics, Philosophy, Electrical Engineering, and Library Service.

Furthermore, there are many departments whose academic programs and research are highly relevant to current research in computer science. In some of them there are opportunities and interests in the exploration of advanced computer applications. In Mathematics, research is underway on computer explorations of the structure of finite groups (Sims), and there are interesting opportunities to develop computer-aided methods for instruction in elementary mathematics. In Electrical Engineering, there is relevant research in switching theory (Marshall), in machine organization (Molony), and in computer modeling for biological and physiological systems (Welkowitz). In Industrial Engineering there are interests in optimization problems, computer-aided design, and graphics (Nanni). In Library Service there is research in information retrieval theory (Artandi). In Physics there is a strong program of computer-based experimentation that poses interesting problems of pattern recognition (Plano); there is also research in computer-aided instruction (Ehrlich). In Urban Planning there are many opportunities for front line exploratory research in design
processes, simulation of complex systems and graphics (Mann, Krueckeberg). In Anthropology, there are challenging problems in the analysis of behavior patterns and in the study of belief structures (D'Andrade). In Psychology there is extensive research in linguistic communication (Rosenberg) and in models of cognitive processes (Neimark). In the School of Medicine there is active interest in computer-aided instruction for parts of medicine, and in automated diagnosis (Hyams).

To date, there have been several contacts between faculty members in Computer Science and most of the people involved in the above activities. However, there is no significant research coupling as yet.

Many of today's advanced problems in computer applications are of interest to people in several Rutgers departments. By strengthening the research activities at the Computer Science Department, and by making serious efforts to collaborate with people in other departments over significant problems, all the departments involved would benefit. The returns will be in terms of contributions to basic understanding (in Anthropology, Psychology and in the other sciences), solutions to important problems of today's society (urban planning, medicine, education), and raising the overall quality of instruction.

E. Center for Computer and Information Services (CCIS)

The CCIS provides computational services to all branches of the University; its equipment includes an IBM 360/67 computer and three IBM 1130 computers for student use. The 360/67
operates both on a batch and time sharing (remote) mode. Under agreement with Princeton University, their IBM 360/91 computer is available to Rutgers on a batch mode for parts of the day. The computational needs of the Computer Science Department, faculty and students, are fully covered at present by CCIS. The department has nine remote terminals (eight in Livingston and one in 5 Huntington Street) for access to the Time Sharing System in CCIS; however, most of the computations at present are in batch mode.

The CCIS offers the resources and the environment that are needed for graduate level experimental work in system software and in computer applications. There is a close working connection between the CCIS and the Computer Science Department; several members of the CCIS have appointments in the Computer Science Department (including the Director of the New Brunswick CCIS center, Dr. Eastwood). The CCIS with its problems of software design and development provides also an excellent workshop where graduate students can experiment with system ideas, and where they can work on specific projects under realistic circumstances. Through this kind of student activity, and also through the research and consulting activities of the computer science faculty, the software resources of CCIS are likely to improve, and thus the overall service capability of CCIS vis a vis the entire University community would be enhanced.
F. **Educational Computing Center (ECC)**

The ECC is a new organization which was created in the Fall of 1969 on the initiative of the Department of Higher Education in the State; its main objective is to plan, promote and coordinate the educational usage of computers in the Institutes of Higher Education in New Jersey. The ECC is planned to be an independent corporate entity sponsored by Rutgers, the State Colleges Community Colleges, and independent institutions. Its director, Dr. Carroll, is a member of the Computer Science faculty. The offices of ECC are currently located at 5 Huntington Street, where the Computer Science Department also occupies some space. In view of its long range goals, the ECC would offer an excellent environment for studies and exploratory developments in computer science -- especially in the design of large data bases and computer networks, and in computer-aided instruction. As in the case of CCIS, the ECC is expected to provide a valuable workshop for graduate students, and conversely the students may satisfy some of the manpower requirements of the ECC (as part-time system analysts and programmers).
III. PhD PROGRAM IN COMPUTER SCIENCE AT RUTGERS

A. Need for a PhD Program in Computer Science at Rutgers

As indicated previously (p.26), there is a clear recognition at the national level of the critical need for doctoral programs in computer science. It is also becoming apparent that in the seventies the interface of computer and society will become universal. As an implication of this it is clear that the responsibility of training leaders in the field becomes great. It is essential for universities to train people with sufficient depth of knowledge as well as with a broad enough vision to see the potential of computers for the solution of significant problems in society. Furthermore, the leaders in the field must be trained to collaborate with people in other fields in initiating, promoting, and working for change. To achieve this kind of training, there is a need for a doctoral program in computer science that combines excellence in scholarship with sensitivity to the "real world" of problems, and emphasis on doing and innovating. Rutgers is in an excellent position to make a significant contribution in the development of such a program - both directly, and possibly by influencing the design of doctoral programs in computer science elsewhere. Some of the reasons for this are: the Rutgers climate of sensitivity to current social problems; the location of the Computer Science Department in Livingston where there is a strong social sciences component; and the emphasis in Livingston on problems of change, on processes of informed social action, and on interdisciplinary approaches to problem solving. Thus, one of the important arguments for initiating now a doctoral program in computer science at Rutgers is that Rutgers is in a strong position to develop it
in a manner that will serve society well.

Rutgers is located in the center of a highly industrialized state, and it is relatively near several major research, development, and manufacturing centers in the computer field (such as Bell labs and RCA); it is also near several major computer installations (in chemical industries and in insurance), and near hundreds of lesser installations and software firms that are concerned with an enormous variety of computer applications. Princeton and Stevens are the only institutions in the general vicinity of Rutgers that have developed relatively small PhD. programs in computer science. These programs cannot satisfy the strong demand in the area for computer science education at the PhD. level. Rutgers University is in an excellent position to respond to such a demand, and to develop into a major center of computer education and research in New Jersey.

The response to our Master's program gives ample evidence of the interest in graduate computer science in the state. Student enrollment has doubled in our graduate program from about 50 last academic year to about 100 at present. Several of these students (about 10) have expressed interest in entering a doctoral program in our department. Tens of inquiries and expressions of interest have been received in the department during the last semester about such a program.

The creation of a strong research activity in computer science, as a part of a PhD. program in the field, would help Rutgers to attract outstanding faculty (not only in computer science but increasingly also in other disciplines, such as psychology, sociology, urban planning), and this would be reflected in an increased overall quality of both the undergraduate and graduate programs. Furthermore, the presence in Rutgers of a substantial group of faculty members and graduate students who
are doing computer research and are exploring advanced applications would accelerate the effective application of computers in the problems of other disciplines. Also, such a group would provide suitable manpower for development projects at CCIS and ECC; and this would lead to an overall enhancement of computer power in the university and in other institutions of higher education in the state.

B. Goals and approaches of the proposed PhD. program

Students enrolled in the PhD. program are expected to acquire a broad knowledge in all areas of computer science, and an overall perspective of the field, its structure, its problems, and its orientation. They are expected to study in depth one or more areas of the field, and to make substantial contributions to them through creative innovation, research, and constructive scholarship.

Students should be able to advance the basic understanding of information processes, and to contribute to the creation and consolidation of knowledge in computer science. In addition, they should be able to see and understand new problems at the interface between computer science and other fields, to find imaginative solutions for them, and to carry them through. They should be able to innovate in the construction and use of computers, and at the same time they should be conscious of the social and economic environment of their work.

As we have indicated in our discussion of the MS program, the rapid rate of change in computer science implies a flexible and adaptable program of course work with a minimum of required courses. This holds true also for the PhD. program. In addition, we must create ample opportunities for the doctoral student to go deeply into one or more directions in the field via advanced course work (if possible) or via independent study and project work.
The basic approach of our proposed PhD program is to emphasize research, project work, active participation in seminars and colloquia, interdisciplinary involvement in computer applications, and a flexible program of course work which is shaped to fit the interests of the student, and which is built on a minimal base of required courses.

We propose to stress independent study, writing review and research papers, doing active work on computers (both state of the art and exploratory work) and interacting with computer people (in Rutgers and elsewhere) and also with people in other disciplines. These are the kinds of activities that will occupy much of the time of a PhD graduate after he ends his formal education, and his experience with them during his stay in the Graduate School would be highly valuable to him and to society.

C. Emphasis of the proposed PhD program

We propose to emphasize in our program the region where applications and computers meet. This means emphasis on algorithms, procedures, and problems; their relationships to various application areas; their expression in high level languages; the design and processing of such languages; and the design of systems for man-machine interaction. With reference to Figure 1 (in Chapter I), our center of interest is located in the upper part of the diagram.

With reference to Figure 2, our interest covers the entire configuration of application areas, with a preference on areas that are strongly relevant to design and decision problems. As mentioned previously, the large "real life" problem domains that we are starting to face today require a merging of the methods and techniques of the relatively "conventional" application areas that are shown in Figure 2.
We are interested in exploring advanced computer applications in urban planning, medicine, education, and computer design. In many of the problems encountered in these areas, the concern is with interactive computing, graphics, large and flexible databases, convenient languages for describing data structures and procedures, languages for man-machine dialogue, and heuristic problem solving methods.

We also propose to establish active coupling with studies in psychology and anthropology that are closely related to current research in computer science. In psychology, for example, work on cognitive processes and memory, studies in language behavior (acquisition, performance) and in dialogue; in anthropology, studies of belief systems, schemes for representing human behavior, and analysis of narrative.

One of the exciting areas that we propose to explore is the use of computers in the development of computer science courses (mainly at the early undergraduate level, but also in advanced new areas where the recorded knowledge is spread over several papers and perhaps it is embodied in complex computer programs). This is a very fruitful area of research, and it has direct significance for our educational programs.

The climate of work that we envision in the department will encourage doctoral students to study and do research in the areas that we have just mentioned. Some of the work will be theoretical, and some constructive--experimental. In all cases, the work will be oriented to significant problems in the mainstream of computer science, and it will have to satisfy the highest standards in the field. However, the treatment of problems in the context of significant applications in the behavioral sciences and in the sciences of design, is likely to give to our program its distinctive character.
D. The Faculty. Current research in the Computer Science Department.

The graduate faculty of the Computer Science Department consists of six full members and eight associates. The director of graduate programs is Professor Saul Amarel. A list of the graduate faculty, including biographical sketches of the full members, is given in Appendix III.

The following is a summary of research areas in which the faculty at Rutgers has been active recently: Artificial Intelligence, theory of problem solving, question answering (Amarel); Numerical Analysis (Fender); Statistical Methods (Philpot); Graphics, Information Retrieval (Baxendale); Language Description, Translation, Switching Theory (Paull); Design of Programming Languages, Language Processors (Rabinowitz); Theory of Programming (Mott); Description Languages, Machine Organization, Computer-Aided Design (Srinivasan). The research of our part-time faculty has been in the following areas: Numerical Analysis (Henry); Simulation, Machine Organization (Kaplan); Formal Languages (Korenjak); Time Sharing, Operating Systems (Easton).

Areas of research in which we are planning future activities are Theory of Algorithms, Computer Aided Instruction, and advanced applications in medicine and urban planning.

At present we have a grant from OSR for research in Machine Problem Solving. This grant covers part of the work of Professors Amarel, Paull, and Srinivasan. We are planning to seek grants from other government agencies (NSF, NIH) in order to strengthen our research base, and also in order to create financial support for research assistants in the PhD program.

E. Curriculum. Study plan.

The redesign of our present graduate curriculum, which is explained in connection with the proposed modifications of our MS program was intended to provide a sound initial basis for the proposed PhD program. The new curriculum is shown in Appendix IV.
It contains 24 courses that are classified in 4 main topics of study, and also "Research in Computer Science" (CS 701, 702).

Our proposed PhD Curriculum for 1970-71 consists of the Curriculum in Appendix VI with one modification, and two added courses. The modification is as follows:

Change the entry in CS 701, 702 to read:

RESEARCH IN COMPUTER SCIENCE.
Variable credit, both terms.

The two courses to be added are as follows:

CS ---: SELECTED PROBLEMS IN COMPUTER SCIENCE. Variable credit, both terms; prerequisite: consent of the department. (Staff)
Design studies, work on computer projects in systems software and in advanced applications; advanced work in areas not provided for in formal courses; conferences, reading and laboratory work will be arranged with the professor in charge.

CS ---: SEMINAR IN COMPUTER SCIENCE. 3 credits, both terms; Prerequisite: consent of department. (Staff)
Lectures and special problems in current computer science research.

We suggest that the above two courses be given 600-level numbers. They form a separate category which we may call "Preparation for Research."

The proposed Curriculum will satisfy the major needs of our program; namely breadth, a sufficient number of advanced courses in the field, flexible arrangement for independent study and project work, an opportunity to discuss and review current research in the field, and a sufficient allowance of credits for research.

We have initiated in the Fall of 1969 a Colloquium in Computer Science where workers in the field are invited to present
their research. Attendance in the Colloquium will be strongly recommended for all PhD students.

We propose to introduce during the next three years the following courses:

1. Design of Programming Languages
2. Theory of Algorithms
3. Data Structures
4. Information Systems
5. Optimization Methods
6. Natural Language Processing
7. Computer Reliability

These courses, together with the courses that we propose for 1970/71, will give us an excellent coverage of the field, and sufficient strength in the areas that we wish to emphasize. Of course, the introduction of these courses will depend on the interests of the faculty that will be joining us in the next three years.

The required total number of credits for course work is 48. A PhD student should complete successfully one required 3-credit course in each of the four main topics of study in our curriculum. Also, he will be required to participate in the "Seminar on Computer Science" course for at least 3 credits. Outside these 15 required course credits, there are no other formal course requirements.

Upon entering the PhD program, each student will be assigned a faculty advisor, and with his help he will choose a study plan (which is updatable). The plan will include the five required courses, and other courses in the computer science department as well as (possibly) courses in other departments. There is considerable flexibility in the choice of a study plan. However,
the student should keep in mind the requirements for the PhD degree (discussed below) in organizing his studies.

In Appendix VI we have given a list of the courses in other departments that are relevant to our graduate program. Students should plan to take courses from this list (especially in Mathematics, Statistics and Philosophy) that provide mathematical background for the topics in which they are specializing. In addition, students will be encouraged to take courses in other departments that are relevant to their project work and the orientation of their research. Such courses may be found, for example, in Psychology and in Urban Planning. In each case, a student should decide on courses in other departments in consultation with his advisor.

In course work, an effort will be made to use term papers and design projects rather than examinations. The "Selected Problems in Computer Science" course will give a student an excellent opportunity to work on the computer on a project of realistic nature. During his participation in the "Seminar in Computer Science" the student is expected to present at least one lecture.

F. Thesis Research

A PhD student should devote at least one year to research (24 credits). During this time he should pursue, under faculty supervision, an original investigation of one or more problems in his area of concentration in computer science, and to present the results in a thesis. Traditionally, a PhD thesis represents a single piece of creative work. Often such work is extremely narrow. Also, it does not permit the student to work on several subjects quasi-simultaneously over a given period of time, with the option to move from one to another according to the progress
of the work, the inputs from the environment, and the flow of ideas—a situation which is common in the real world of research. In order to counteract narrowness, and to provide the valuable experience of going through several cycles of research activity—preferably of different types, e.g., an experimental piece and a theoretical piece—it would be desirable to encourage thesis research which is not limited to a single piece of creative work. We are proposing that the option of a thesis covering several pieces of research (usually 2 or 3) be available in our PhD program. An option which is based on a single piece of research should also be available.

Work on the thesis research should be carried out under the direction of a faculty supervisor. It is up to the student to find a member of the graduate faculty in his general area of interest who is willing to be his thesis supervisor. The thesis must be approved by a faculty committee of four members, headed by the thesis supervisor, which is formed according to the rules of the Graduate School.

G. Requirements for the PhD degree

In addition to the general requirements for the PhD degree established by the Graduate School, the department proposes the following requirements:

1. A student must complete successful course work for a minimum of 48 credits including the five required courses. If he can demonstrate, by record of previous study or by examination, that he has mastery in the area of a required course, other than "Seminar in Computer Science," then he will not be required to take the course.

2. A student must demonstrate that he has reading knowledge of one foreign language among Russian, French, German, or Japanese.
3. He must complete a significant project in systems software or in an application area. The project should involve problem formulation, program development and testing, and documentation. The student may develop the project as part of his work in a 500-level course; or in the course "Selected Problems in Computer Science" which is well suited for work on such a project; or in connection with a research assistantship; or in work with CCIS or ECC.

4. He must pass a two-part qualifying examination. The first part is written, and it is identical with the final Masters examination. The second part is oral and it is intended to test the depth of the student's knowledge in his chosen area of concentration. The first part of the qualifying examination should be taken as soon as a student has completed 30 credits of course work; it may optionally be taken by a student after he has completed 24 credits. The second part can be scheduled in a flexible way, but it should take place no later than 6 months after a student has completed his course work. The department will use a student's performance in the written part of the qualifying examination together with his course grades and faculty recommendations to determine whether he should be encouraged to continue in his PhD program.

If a student has fulfilled the above four requirements and if he has found the thesis supervisor who recommends his candidacy, then he becomes a candidate for the doctor's degree. He can then proceed to his thesis work, prepare a doctoral thesis, have it approved by his thesis committee, and defend it.

H. Requirements for Admission and Scholastic Standing

The minimal requirements for admission of a student with a baccalaureate are identical with the requirements of admission to our MS program.
In addition to the requirements of the Graduate School for transfer students, we require that they pass the written part of the qualifying examination.

Only courses with a grade of 1 or 2 can be used to meet the course requirements for the PhD degree. A PhD student is expected to maintain a sustained record of distinction during the course of his studies. If a student obtains a grade other than 1 or 2 in more than a quarter of the total number of credits that he has received towards his doctorate, then his performance will be considered unsatisfactory, and normally he will not be permitted to continue in the program. However, each case will be considered individually by a committee of the department.

I. Master of Philosophy

We propose that a Master of Philosophy degree be available for Computer Science students on the basis of the guidelines set forth by the Graduate School. The only difference in requirements with respect to the PhD degree is the lack of thesis research. Students receiving the M.Phil. in Computer Science would have achieved a comprehensive mastery of the field and a deep knowledge in one of its areas, and also they would have gained a broad working experience with software design and with advanced computer applications.

J. Outline of Typical Doctoral Student's Program

Let us consider a hypothetical program of a well-prepared, full time student, whose concentration of interest is in Artificial Intelligence, in related theoretical problems, and in possible applications in dialogue systems for computer-aided instruction.

First Semester

CS:503 : Advanced Programming Techniques (3 credits)
CS y : Numerical Analysis (3 credits)
CS 508 : Non Numerical Algorithms (3 credits).
Phil 532 : Mathematical Logic (3 credits)
Second semester

CS 507: Introduction to Sequential Machines (3 credits)
CS 509: Computers in Mathematical Research (3 credits)
CS 510: System Modeling (3 credits)
CS 512: Introduction to Operating Systems (3 credits)

Third semester

CS 505: Syntax and Semantics of Programming Languages (3 credits)
CS 519: Formal List Processing (3 credits)
Phil 533: Seminar in Decision and Inference (3 credits)
Math 563: Theory of Recursive Functions (3 credits)

Fourth semester

CS 598: Special Topics in Computer Science; Artificial Intelligence (3 credits)
CS ---: Seminar in Computer Science (3 credits)
CS ---: Selected Problems in Computer Science (6 credits)

Fifth and Sixth semester

CS 701, 702: Research in Computer Science (24 credits)

Defense of thesis. Graduation

The student completes his basic four course requirements in the first year of his studies (CS 503, CS 507, CS 508), and he also includes in his program a combination of systems oriented courses (CS 510, CS 512) and theory oriented courses (Phil 532, CS 501). He then passes the written part of this qualifying examination. In his third semester he concentrates on theoretical background and conceptual schemes of computing. The fourth semester is devoted to (i) a Special Topics course (CS 598) in his main area of interest, where he may contribute a Seminar lecture and a paper (say a review of machine theorem proving in elementary logic), (ii) a "Seminar in Computer Science" course
(which is required of him) where he studies recent research in question answering and data structures, and he contributes lectures to the group, and (iii) an extensive effort on a large systems project under the "Selected Problems in Computer Science" course. In this project, he may develop a dialogue system for an instructional database in an area of medicine. During this work he is likely to collaborate with people in the medical school and the CCIS. After passing the oral part of his qualifying examination, he devotes one year to research (perhaps he works on a theory of certain linguistic models of dialogue that were stimulated by his project work, and he also explores via considerable computer experimentation the properties of certain heuristic search procedures that are used in theorem provers) after which he defends his thesis and he graduates.
IV. PEOPLE AND RESOURCES

A. Students

It is our present intention to accept between 5 and 10 students for the inception of the doctoral programs in 1970-71. About half the initial intake of doctoral students are likely to come from students who have passed through the department's Master of Science program. The other half would be students from the computer industry who have taken the Master's degree in computer science at other universities. Already a sufficient number of high calibre students have indicated strong interest in the projected doctoral program to ensure a healthy start.

At present, student support in the department depends exclusively on teaching assistantships. We have now six teaching assistants (TA's), and our program will necessitate four additional TA's for the next academic year. We hope to create support for several research assistants (3-4) via research grants that we are planning to seek from NSF and NIH during the current year. We are also exploring the possibility of direct support from industry in the form of fellowships (RCA, Bell Telephone Laboratories, IBM). The prospects for research assistantships and fellowships will become much better once our doctoral program is under way, and a reasonable level of research activity is established in the department.

It is customary in industrial research laboratories (such as Bell Labs and RCA Labs in our vicinity) to provide full financial support for some of their employees that are working in a PhD program. This is another important source of possible support for some of the doctoral students in our program.
B. Faculty

Our present graduate faculty provides an excellent nucleus for the initial stage of our doctoral program. However, this nucleus must be augmented rapidly by a group of faculty members that can generate and sustain an adequate level of research activity. The research requirements of the doctoral program (and the associated teaching and advising functions) and the expanding needs of the undergraduate program and the MS program will necessitate the recruitment of four full-time faculty members for 1970-71.

C. Resources

The Computer Science Department is presently adequately but temporarily housed in a student dormitory, building 4138A, at Livingston College. Part of the new "Science Center" under development at University Heights is the Mathematics-Statistics-Computer Science building which will be completed in 1971 and will provide permanent quarters for the Department of Computer Science and the Center for Computer and Information Services.

The present computer resources of the CCIS are adequate for the requirements of our program in the next 2-3 years. To service fifteen faculty members and ten research students we will need five new computer terminals in 1970-71.

The present library holdings in Computer Science are adequate for the general requirement of our doctoral program. To permit faculty and students to keep up to date with the latest developments in the field, it will be necessary to develop a small local library of periodicals and books for immediate access within the department.
Appendix I. Undergraduate Courses in Computer Science 1969-70

BASIC COMPUTER PROGRAMMING (Cr. 1) 02:198:103
Droege (Course Coordinator)
The use of a compiler language, FORTRAN IV, in analyzing, formulating, and solving scientific problems on a computer. Algorithms, programs and computers. Debugging and verification of programs. Laboratory sessions will meet weekly.

THE COMPUTER IN SOCIETY (Cr. 4,4) 02:090:117-118
Staff. Lec. 75 minutes; Rec. 75 minutes; Lab. 50 minutes.
The history of computer development. Logical processes and problem solving. Programming languages and software systems. Computer hardware sub-systems. Data banks, communications and computer networks. Computers in the business community, in government, in the social sciences and the humanities, in science and technology, in law and medicine, and in the space age; successes and failures in these areas. Legal aspects of computer usage. Automation and leisure.

INDEPENDENT STUDY A (Cr. 4,4) 02:198:223-224
Staff
Assigned readings and discussions in Logic and Algorithms with applications to the Computer and Information Sciences. Students will be required to give at least one lecture on a chosen topic.

THEORY OF COMPUTER PROGRAMMING (Cr. 3,3) 02:198:301-302
Staff. Lec. 2 hours; Lab. 3 hours.
Prerequisite: One year of college mathematics.
The use of machine language, assemblers, compilers, and interpreters; and an investigation of their structure. Systems and utility programs, programming techniques, and recent developments in computing. Laboratory sessions in writing, testing, and using programs.

COMPUTER PROGRAMMING AND DATA PROCESSING (cr. 3,3) 02:198:373-374
Cox. Lec. 2 hours; Lab. 3 hours.
Prerequisites: Math 136 (or equivalent) and 12 credits in social science or business courses.
Designed for students in business and social science.
Data processing related to programming of problems selected from the fields of business and social science; logical structure of electronic computers; basic principles of programming in machine oriented, procedure oriented and problem oriented languages. Organization and processing of files, data structures, and simulation in control systems.
COMPUTER PROGRAMMING AND NUMERICAL METHODS (Cr. 3, 3) 02:198:377-378
Baxendale and Beaucage. Lec. 2 hours; Lab. 3 hours.
Prerequisite: Math. 242 or 314.
Designed for students in science and engineering.
Programming of numerical algorithms based on calculus including polynomial interpolation, direct and iterative methods for solving equations. Matrix iterative methods, numerical integration and solution of ordinary differential equations; efficient use of computers, principles of programming; machine language, assembly programs, compilers, subroutines, and library programs.

COMPUTER PROGRAMMING AND NUMERICAL METHODS FOR RESEARCH (Cr. 3) 02:198:417
Fender. Lec. 2 hours; Lab. 3 hours.
A course for students engaged in research projects requiring extensive computation. Programming techniques, including the use of machine language, interpretive and assembly programs, compilers, sub-routines, and program library materials. Numerical methods for inverting matrices, least squares methods, approximation and error analysis.

(NUMERICAL METHODS IN ORDINARY DIFFERENTIAL EQUATIONS) (Cr. 3) 02:198:479
Fender. Lec. 2 1/2 hours; Lab. 1 1/2 hours.
Not offered in 1969/70.

INDEPENDENT STUDY B (Cr. 4, 4) 02:198:495-496
Staff
Assigned reading and discussions in the Computer and Information Sciences.
Appendix II. Graduate Courses in Computer Science 1969-70

*Computer Science 501,502. PROGRAMMING SYSTEMS AND LANGUAGES

- Three credits each term. Staff
  The structure of algorithmic languages; list processing and string
  manipulation languages; batch processing systems; multiprogramming
  and multiprocessor systems.

Computer Science 503,504. INTRODUCTION TO SYSTEM PROGRAMMING

- Three credits each term. Mr. Rabinowitz
  Assemblers, interpretive routines, and compilers; macro assemblers;
  open and closed subroutines; calling sequences; recursive sub-
  routines; data access schemes, virtual memory, and paging; file
  design; data organization and management.

Computer Science 505. SYNTAX AND SEMANTICS OF PROGRAMMING LANGUAGES

- Fall term. Three credits. Prerequisite: consent of department.
  Lecture 2 1/2 hours, laboratory 1 1/2 hours. Mr. Paull
  Formal description of programming language translation; methods
  for describing syntax, data structures, semantics; methods for
  parsing and code generation; SNOBOL will be used to illustrate
  concepts.

Computer Science 506. DESIGN AND CONSTRUCTION OF COMPILERS

- Spring term. Three credits. Prerequisite: consent of department.
  Lecture 2 1/2 hours, laboratory 1 1/2 hours. Mr. Rabinowitz
  Algebraic compiling; bounded context translation; precedence and
  operator precedence languages; syntax-directed compiling; top-
  down and bottom-up parsing algorithms; reduction analysis. Stu-
  dents will be required to write a compiler for a simple algebraic
  language.

Computer Science 507. THEORY OF FINITE STATE MACHINES

- Fall term. Three credits. Prerequisite: consent of de-
  partment. Mr. Paull
  Boolean minimization. Analysis and synthesis of sequential
  machines. Regular expressions and finite-state grammars.

Computer Science 508-509. COMPUTERS IN MATHEMATICAL RESEARCH

- Three credits each term. Prerequisite: consent of de-
  partment. Mr. Beaucage
  Recent uses of computers in investigation of finite topological
  spaces. Dyer's commuting functions conjecture, coset enumera-
  tion, and other areas of pure mathematics.
Computer Science 510,511. SYSTEM MODELING AND SIMULATION
Three credits each term. Prerequisite: consent of
department. Mr. Kaplan
Probability theory and stochastic processes; design of experi-
ments for optimization; analysis and interpretation of simula-
tion experiments; general simulation languages; large scale
system simulation.

Computer Science 512,513. OPERATING SYSTEMS
Three credits each term. Prerequisite: consent of instructor.
Lecture 2 1/2 hours; laboratory 1 1/2 hours. Mr. Easton
Processes, inter-process communication, problems of multipro-
gramming and multiprocessing, organization and queuing of mass stor-
age, file structures, paging and segmentation, working-set models
for page-turning.

Computer Science 514,515. COMPUTER-ORIENTED NUMERICAL ANALYSIS
Three credits each term. Prerequisite: advanced calculus,
linear algebra, and facility with a higher level computer
language. Lecture 2 1/2 hours; laboratory 1 1/2 hours.
Mr. Fender
Functional approximation, least square and minimum-maximum error
approximations, linear equations and matrices, existence theorems
for integral and differential equations by iterative methods.
Numerical differentiation and integration, numerical solutions
of partial differential equations, initial value and boundary
value problems.

Computer Science 516. COMPUTER-ORIENTED STATISTICAL ANALYSES
Fall term. Three credits. Prerequisite: introductory
courses in applied statistics. Mr. Philpot
Analysis and interpretation of data, using statistical programs.
Means, variances, tests of hypotheses, analysis of variance and
covariance, regression analysis, and non-parametric tests; data
manipulation.

Computer Science 517. THEORY OF FORMAL LANGUAGES
Spring term. Three credits. Prerequisite: consent of
department. Mr. Paull
Chomsky's hierarchy of grammars and languages; regular sets;
pushdown automata and context-free languages; linear bounded
automata and context-sensitive languages; stack automata;
decision problems related to formal languages.

Computer Science 519. FORMAL LIST PROCESSING
Spring term. Three credits. Prerequisite: one year of
system programming. Lecture 2 1/2 hours; laboratory 1 1/2
hours. Mr. Mott
Algebra of categorial grammars and list parsing; list processing
languages; LISP; lambda calculus machines; applicative expressions;
combinatory logic machines.
Computer Science 520. LINEAR PROCESSES
Fall term. Three credits. Prerequisite: 198:515
Lecture 2 1/2 hours; laboratory 1 1/2 hours. Mr. Fender
Theory of matrices; computer solutions of linear equations; inversion under adverse conditions; characteristic roots and vectors. Applications to linear programming and game theory.

Computer Science 521. STABILITY ANALYSIS
Spring term. Three credits. Prerequisite: 198:515
Lecture 2 1/2 hours; laboratory 1 1/2 hours. Mr. Fender
Stability analysis and uniqueness conditions for the numerical solution of ordinary differential equations; consistency and convergence of solutions; round-off distribution; local and global errors.

Computer Science 522, 523. APPROXIMATION AND INTERPOLATION
Three credits each term. Prerequisite: 198:515. Mr. Fender
Pure mathematics is used to study the applied problem of trigonometric and polynomial interpolation and approximation. Vector spaces, normed vector spaces, inner product spaces, Hilbert space.

Computer Science 524, 525. GRAPHICAL INFORMATION PROCESSING
Three credits each term. Prerequisite: matrix methods. Staff
Computer graphics techniques for input, storage, manipulation, retrieval and display of spatially variable data. Interactive graphic design systems. Data structures and languages.

Computer Science 599. SPECIAL TOPICS IN COMPUTER SCIENCE: COMPUTING SYSTEMS AND MACHINE ORGANIZATION
Fall term. Three credits. Prerequisite: consent of department. Seminar 2 1/2 hours. Mr. Srinivasan
Selected readings with weekly conferences. Theme of seminar: Advances in machine organization and their relation to software, high level programming languages, special areas of computer applications, and hardware; introduction to system design methodology based on a computer description language.

Computer Science 598. SPECIAL TOPICS IN COMPUTER SCIENCE: DATA STRUCTURES AND ARTIFICIAL INTELLIGENCE
Spring term. Three credits. Prerequisite: consent of instructors. Seminar 2 1/2 hours. Mr. Amarel & Mr. Srinivasan
Selected readings with weekly conferences. Theme of seminar: Data structures, their description and manipulation, associated languages, and their relevance to problem solving and question answering by computer; current problems in artificial intelligence.
Computer Science 701,702. RESEARCH IN COMPUTER SCIENCE  
Three credits each term.  

† Indicates course not offered during 1969-70 academic year  
* This course number will be changed to 597 in the fall of 1970
Appendix III. Graduate faculty of the Computer Science Department

A. Members of the Graduate faculty; Vitae

SAUL AMAREL, Professor of Computer Science and Chairman of the Department. (LC/LA)

Dr. Amarel received his B.S. in 1948 and the degree of Ingenieur E.E. in 1949 from the Israel Institute of Technology, Haifa. He pursued graduate studies at Columbia University, New York City, receiving his M.S. in 1953 and his Doctor of Engineering Science in 1955.

For six years Dr. Amarel was associated with the Scientific Department of the Israeli Ministry of Defense where he led the development of control and communication systems and also conducted research on computer simulation methods related to operations research problems.

From 1953 to 1955 Dr. Amarel was associated with the Electronics Research Laboratory of Columbia University where he developed operational methods for the analytic study of linear and nonlinear dynamic systems. From 1957 to 1969 Dr. Amarel was a member of the technical staff at RCA Laboratories, David Sarnoff Research Center, in Princeton, New Jersey. In 1958 he organized the Computer Theory Group; he was head of the group until mid-May 1969, directing research and working on a broad spectrum of problems in computer design, programming and advanced applications. His recent work has been in artificial intelligence, computer linguistics, the theory of computational processes, and information systems. In 1969 Dr. Amarel has joined Rutgers University as Professor and Chairman of the Department of Computer Science.

Dr. Amarel has presented numerous talks in artificial intelligence and computer theory at various scientific conferences and research seminars. In the summer of 1964, he was a guest lecturer at the Moore School of Electrical Engineering, University of Pennsylvania. In the spring of 1966, he was a visiting professor at the Computer Science Department, Carnegie Mellon University. He is a Columbia University Associate in the Seminar on Relations between Research and Education and Computers.

Dr. Amarel is a member of the Editorial Board of the Journal "Artificial Intelligence," and he is a Director of the American Society for Cybernetics. He is a senior member of IEEE, and a member of SIAM, ACM, AAAS, Sigma Xi, and the New York Academy of Sciences. He contributes paper reviews to the Mathematical Reviews, Computing Reviews of ACM, and to the IEEE Transactions of the PGECA.
Publications


"On the Formation of the Concept of a Transformation by Computer" in Summaries of the 1964 International Congress for Logic Methodology and Philosophy of Science, the Hebrew University, Jerusalem, Israel.


FRED GEORGE FENDER, Professor of Computer Science and Mathematics (LC/IA)

Dr. Fender received his A.B., M.S., and Ph.D. (1936) from the University of Pennsylvania. His graduate training was in theoretical physics. In 1937 he joined Rutgers University. During the war he designed and directed the building of several special purpose analog computers, one of which was the "Dehmel Flight Trainer." Dr. Fender pioneered in computer education from its early days, and he was instrumental in introducing computers and computer courses in Rutgers. Since 1947 he taught numerical analysis for digital computers. The course has constantly been updated as better methods have been found and better computers developed. Dr. Fender's work has been mainly directed to differential equations, and recently to problems in partial differential equations. His current research interests are in computer methods for solution of hyperbolic and elliptic cases. Dr. Fender was the first chairman of the Computer Science Department, from 1966 to 1967.

His scholarly activities include a contract with the State of New Jersey for the Computer design of the performance of the reservoir system of the Raritan Valley; the consulting with a water purification program in New York City which involved the computer solution of 35 simultaneous differential equations; a contract with the College of Agriculture to use linear programming to obtain the least cost of animal feed that supplied the necessary amino acids for nutrition, and about fifty other
computer projects of lesser magnitude with the U.S. Weather Bureau and the College of Engineering.

Dr. Fender is a member of the American Physical Society, the American Mathematical Society, the Mathematical Association of America, the Society for Industrial and Applied Mathematics, the New York Academy of Sciences, the New Jersey Academy of Science, the Society of Sigma Xi and a fellow of the American Association of the Advancement of Science.

THOMAS HEZEKIAH MOTT, JR., Professor of Computer Science and Dean of the Graduate School of Library Service

Thomas H. Mott, Jr., was born in Houston, Texas on January 24, 1924. He received the B.A. degree from Rice University, Houston, in 1948 and the Ph.D. degree in philosophy from Yale University, New Haven, Connecticut, in 1956.

From 1943 to 1946 he served as a meteorologist with the U.S. Army Air Corps, having received training at New York University, New York, New York. He taught mathematical logic at Yale University for a year before joining the Mathematics Research Department of Remington Rand UNIVAC at St. Paul, Minnesota in 1956. He was a member of the technical staff of the RCA Laboratories, Princeton, New Jersey, from 1958 to 1961, doing research in information processing and artificial intelligence. He joined Rutgers University, New Brunswick, New Jersey in 1962 as an Associate Professor of Information Processing, and in 1963 was promoted to Professor. From 1966 to 1969 Dr. Mott was Director of the Center for Computer and Information Services, from 1967 to 1969 he was Chairman of the Department of Computer Science. In 1969 he was appointed Dean of the Graduate School of Library Service. He has lectured at summer programs at the University of Michigan, Ann Arbor, and the University of Pennsylvania, Philadelphia. In 1967, he was a Visiting Professor of Computer Science at the University of Pittsburgh. In addition to his teaching and research duties at Rutgers, he was Visiting Professor of Mathematics at Stevens Institute of Technology, Hoboken, New Jersey for several years, where he participated in the development of a graduate Ph.D. program in computer science.

Dr. Mott is a founder, vice-president, director and treasurer, of Applied Logic Corporation, Princeton, New Jersey which specializes in time-sharing computing services for scientific and Industrial users and conducts basic research in the applications of logic to computer and information technology.

Dr. Mott is currently teaching a course on Programming Theory for Information Handling in the Graduate School of Library Service. He plans to develop a text based on this course designed to meet
some of the needs of graduate students in library service in the area of information science.

Dr. Mott is a member of Phi Beta Kappa and Sigma XI. His professional affiliations include Association for Computing Machinery and American Society for Information Science. He contributes critical reviews to the Journal of Symbolic Logic and the IEEE Transactions of the PGEc.

Publications:


MARVIN C. PAULL, Professor of Computer Science (LC/IA)

Mr. Paull received his B.S.E.E. in 1952 from Clarkson College of Technology. After one year of work at Arma Corporation, he joined Bell Telephone Laboratories in 1953 where he was employed until 1966 at which time he joined the Computer Theory Group of RCA Laboratories in Princeton. In 1969 Mr. Paull joined the Department of Computer Science of Rutgers University as Professor. At Bell Telephone Laboratories his work included both research and development on magnetic logic, sequential machine and systems, switching networks, macro assemblers and compilers. In the computer theory group of RCA Laboratories, Mr. Paull has been concerned with the problems involved in finding general techniques for describing programming language translations and building compilers. From 1967 Mr. Paull has been teaching a graduate course in advanced programming at Columbia University.

Mr. Paull is a member of Tau Beta Pi, Eta Kappa Nu.

Publications.


"Minimizing the Number of States in Incompletely Specified Sequential Switching Functions," IRE Transactions on Electronic Computers, Sept. 1959; co-authored with S.H. Unger.


IRVING NATHANIEL RABINOWITZ, Professor of Computer Science (LC/LA)

Dr. Rabinowitz received his B.S. from City College, N.Y. (Physics and Mathematics), and his M.A. and Ph.D. from Princeton University. His graduate work was in Astronomy and Astrophysics.

From 1953 to 1959 he was associated with the Knolls Atomic Power Laboratory, General Electric Company, as a research assistant in numerical analysis working on the CPC, Univac I, and IBM 650 computers. From 1959 to 1957 he participated in the Electronic Computer Project at the Institute for Advanced Study, Princeton; he worked first on the Meteorology program and then on systems programming for the IAS computer. From 1957 to 1962 he was Head of the Computer section at the Plasma Physics Laboratory, Princeton University. His work was in numerical analysis problems originating from plasma physics and in systems programming. From 1962 to 1966 Dr. Rabinowitz was associate director of the Princeton University Computer Center. In 1969 he spent
one year on leave to the Indian Institute of Technology at Kanpur, India, as a Visiting Professor. There he organized a center using an IBM 1720 and he taught programming and numerical analysis. From 1966 to 1969 Dr. Rabinowitz was Director of the Computer Center and Associate Professor of Mathematics at Stevens Institute of Technology. His academic work was primarily concerned with the organization of the computer science program at the Master's and Doctor's level.

In 1969 Dr. Rabinowitz joined the Computer Science Department of Rutgers University as Professor. His current research interests are in the design of programming languages and their processors.

Publications:

"Transport Properties of a Gas at Low Temperatures" (with Leo F. Epstein & Nancy E. French) - KAPL Report, April 1, 1954.

"Inhomogeneous Stellar Models III" (with M. Schwarzschild and R. Harm) - Astrophysical J., 118, 326, 1953.


"The Velocity Distribution of Plasma Electrons in an External Electric Field" (with Ira B. Bernstein) - Proc. 4th Intl. Conf. on Ionization Phenomena in Gases, Uppsala, 1959.


"Another Floating Point Interpretive Routine"


Dr. C. V. Srinivasan received his B.S. in 1953 and the Diploma, B.M.I.T. (Elec.) of the Madras Institute of Technology, India, in 1956. Thereafter, for three years he worked at the Tata Institute of Fundamental Research, Bombay, India. During this period, he participated in the design and development of a general purpose, digital computer, and also conducted research on sequential machine theory. In 1959, Dr. Srinivasan came to Columbia University, New York, to pursue graduate studies in Electrical Engineering, and received his M.S. in February 1963 and Doctorate in Engineering Science in June 1963.

From 1960 to 1962 Dr. Srinivasan was working at the Department of Electrical Engineering, Columbia University, as a Research Assistant. During this period he developed methods of analysing the transition properties of Linear Sequential Machines. From 1962 to 1969 Dr. Srinivasan was a member of the technical staff of RCA, David Sarnoff Research Center in Princeton, New Jersey, conducting studies in the areas of switching theory, probabilistic logic and sequential machines, error correcting codes for memory systems, self-repairing systems, formal descriptive methods for computing systems, automatic design aid systems for computer systems design, construction of large design data-base, and development of macro systems for software implementation. In 1969, Dr. Srinivasan joined the Department of Computer Science of Rutgers University as Research Associate Professor.

Dr. Srinivasan has published several papers and presented talks in the areas of sequential machines, error correcting codes, descriptive language and design and systems. In September 1967 he presented a series of lectures on "A Computer Description Language" at the Tata Institute of Fundamental Research, Bombay, India; in July 1968 he presented a series of lectures on "Error Correction in Memory Systems" at the Northwestern University, Evanston, Illinois. Dr. Srinivasan is a member of IEEE, IT, EC and SSC, and ACM. He referees papers for IEEE, EC and CT, and for Computing Reviews.

Publications


"State Diagram of Linear Sequential Machines" Jour. of Franklin Inst., Vol. 273, No. 5, 1962

"State Diagram of Linear Sequential Machines (with applications to decoding of error correcting codes), Doctoral Dissertation, Department of E.E., Columbia University, New York, 1963."


B. Associates of the Graduate Faculty

STANLEY BAXENDALE, Associate Professor of Computer Science (LC/LA) B.S. (Leeds)

DAVID ROBERT BEAUCAGE, Assistant Professor of Computer Science (LC/LA) A.B., M.S. (Rutgers); Ph.D. (New York at Stony Brook)

WILLIAM JOHN CARROL, Professor of Computer Science and Director of the Educational Computing Center A.B. (Catholic); M.B.A. (Detroit); Ph.D. (New York University).

WILLIAM BIGELOW EASTON, Lecturer in Computer Science (LC/LA) B.S. (Cornell); Ph.D. (Princeton)

LAUCHLAND ALEXANDER HENRY, Lecturer in Computer Science (LC/LA) B.S., M.S., D.Sc. (Columbia)

KENNETH R. KAPLAN, Lecturer in Computer Science (LC/LA) B.E.E., M.E.E., Ph.D.E.E. (Polytechnic Institute of Brooklyn)

ALLEN JOHN KORENJAK, Lecturer in Computer Science (LC/LA) B.S. (Notre Dame); M.A., Ph.D. (Princeton)

JOHN WESLEY PHILPOT, Assistant Professor of Computer Science (LC/LA) B.A.E. (New York University); M.S., Ph.D. (Virginia Polytechnic)